International Journal of Multidisciplinary Research and Publications
ISSN (Online): 2581-6187

Exploring Secure Hashing Algorithms for Data Integrity Verification

Chris Gilbert[1], Mercy Abiola Gilbert[2]
[1]Department of Computer Science and Engineering/College of Engineering and Technology/William V.S. Tubman *University*
[2]Department of Guidance and Counseling/College of Education/William V.S. Tubman University/
Corresponding Author Email Address: cabilimi@tubmanu.edu.lr

Abstract—*In an era marked by the exponential growth of digital data and the widespread adoption of cloud-based storage, ensuring the authenticity and integrity of digital content has become an urgent and complex challenge. This study investigates the robustness, efficiency, and applicability of secure hashing algorithms, particularly SHA-family variants in enhancing data integrity verification mechanisms across diverse computational environments. The research critically evaluates the security assumptions underpinning Proofs of Retrievability (PoR) schemes, revealing vulnerabilities in legacy protocols such as the Jules–Kaliski construction and its RSA-SHA derivatives when subjected to modern attack models. Through a hybrid methodology combining theoretical analysis, cryptographic simulation, real-world testing, and FPGA-based hardware validation, the study explores the collision resistance, pre-image resistance, and implementation efficiency of SHA-1, SHA-256, SHA-512, and SHA-3 algorithms. Experimental results demonstrate the superior performance of SHA-256 in balancing cryptographic strength with computational feasibility, while highlighting the potential of permutation-based SHA-3 candidates in resource-constrained environments. Further, the integration of integrity mechanisms into digital file formats, coupled with novel techniques such as substring index tables and block power indexing, provides a lightweight yet effective framework for tamper detection. Real-world case studies, alongside performance benchmarks, affirm the practicality of the proposed approaches. The study concludes with strategic recommendations aimed at enhancing the resilience of hashing algorithms in light of evolving security threats, and suggests future research directions including post-quantum cryptographic resilience and hardware-accelerated implementations*.

Keywords— *Secure Hashing Algorithms, SHA-256, SHA-3, Data Integrity, Proofs of Retrievability (PoR), Collision Resistance, Cryptographic Verification, File Format Security, FPGA Acceleration, Substring Indexing, Cloud Storage, Post-Quantum Cryptography*.

I. INTRODUCTION

Ensuring the security and correctness of Proofs of Retrievability (PoR) schemes requires assumptions that are both collision-free and computationally efficient (Li et al., 2022; Opoku-Mensah, Abilimi & Amoako, 2013). The Jules–Kaliski (J.&K.) protocol, a landmark PoR construction, relies on a modified RSA assumption to guarantee its integrity checks (Zachos et al., 2023). However, despite this reliance, no fully efficient, collision-resistant assumption has been formally established to underpin the protocol's security (Shen et al., 2023; Yeboah, Opoku-Mensah & Abilimi, 2013a). Although Hamadani, Ganai & Bashir (2023), extended the J.&K. protocol, reducing client-side computation, they similarly did not introduce a new, provably secure assumption. Consequently, the search for a robust, collision-resistant foundation for PoR schemes remains open (Jager, Kurek & Niehues, 2021).

In this paper, we demonstrate that both the original J.&K. protocol and its RSA-and-SHA-based variants are vulnerable when evaluated under contemporary, stronger security models that emphasize unforgeability and data integrity. By refining the classical RSA assumption with advanced analytical tools, we uncover critical weaknesses in these constructions (Imam et al., 2021; Yeboah, Opoku-Mensah & Abilimi, 2013b). Our results show that, without a stronger underlying assumption, these protocols cannot guarantee collision resistance or integrity against modern adversaries.

The rapid adoption of third-party cloud storage has heightened the importance of verifying data integrity: clients must be able to detect any unauthorized modification of their outsourced data. Traditional integrity verification mechanisms, built on client-server architectures, impose substantial computational costs on one or both parties. To address these inefficiencies, Jules and Kaliski introduced a PoR scheme that balances security with performance. According to Kadioglu & Alatas (2023), further optimized this approach, significantly reducing the client's workload. Despite these improvements, both protocols continue to depend fundamentally on RSA and SHA-based integrity checks, which we show to be insufficient under rigorous security scrutiny.

1.1 Background and Significance

Secure hash functions have long served as the cornerstone of data integrity verification. Ranging from simple checksums to sophisticated constructions based on cryptographic primitives, these functions map arbitrary-length inputs to fixed-size digests (Youvan, 2024; Gilbert, Gilbert & Dorgbefu Jnr, 2025a). By storing the digest in a trusted, tamper-resistant location, one can later recompute the hash of retrieved data and compare it against the stored value; equality implies that the data has not been altered. This mechanism parallels the use of Merkle trees, where a mismatch at the root hash unequivocally signals tampering in one or more leaf nodes.

The study of secure database systems lies at the intersection of cryptography, information security, and database management. From a cryptographic standpoint, these systems leverage hash functions, digital signatures, and encryption to enforce data integrity and confidentiality. From an information-science perspective, they facilitate the storage and retrieval of data whose informational value must be preserved and validated. Finally, at the database-management level, secure systems must manage resources disk blocks, files, rows; under varying isolation levels to ensure that security guarantees hold across all layers of the system architecture (Chandramouli & Pinhas, 2020; Opoku-Mensah, Abilimi & Boateng, 2013). Understanding these interdependencies is crucial for designing PoR schemes that are both efficient and provably secure.

1.2. Research Objectives

The overarching aim of this study is to enhance the security of digital file formats, thereby strengthening the integrity assurances associated with digital objects of both general and cultural significance. File formats establish the rules governing the creation, storage, and, where applicable, the encapsulation of descriptive metadata. They are designed to be hardware-agnostic and resilient to technological evolution. To achieve this aim, the primary research objectives are as follows:

I. Evaluate various Secure Hash Algorithm (SHA) variants to determine their efficacy in detecting file modifications and preserving the verifiable "fingerprint" of a digital object.
II. Design and implement mechanisms for embedding integrity checks within selected file formats, ensuring that any unauthorized alteration can be reliably detected.
III. Develop realistic test environments and datasets that mirror practical use cases, thereby generating representative inputs for integrity validation.
IV. Conduct comprehensive validation experiments to measure the robustness, performance, and false-positive/false-negative rates of the integrity-enhanced file formats under various attack and corruption scenarios.

1.3. Research Questions

To guide this investigation, the following research questions have been formulated:

i. Which SHA-family algorithm offers the optimal balance between computational efficiency and tamper-detection sensitivity when applied to common digital file formats?
ii. How can integrity-checking mechanisms be seamlessly embedded within existing file formats without compromising compatibility or performance?
iii. In what ways do different types and magnitudes of data perturbations (bit flips, metadata modifications, structural reordering) affect the output of SHA-based integrity indicators?
iv. What visualization techniques most effectively reveal patterns of integrity indicator deviations across perturbed datasets?
v. Can residual cryptographic signatures be reliably traced back to specific tampering events, thereby enabling fine-grained localization of unauthorized changes within a file?

1.4. Research Methodology

To investigate the robustness and applicability of secure hashing algorithms for data integrity verification, this study adopted a hybrid research methodology that combines theoretical modeling, algorithmic simulation, practical experimentation, and real-world validation (Chen, Gu & Yan, 2023; Yeboah & Abilimi, 2013).

The research began with an in-depth conceptual and theoretical analysis of well-established hash functions, particularly the SHA family (SHA-1, SHA-256, SHA-512, SHA-3), including legacy functions like MD4 and MD5. Emphasis was placed on understanding the cryptographic foundations of these algorithms, focusing on key properties such as collision resistance, pre-image resistance, and avalanche effects (Wang & Tabassum, 2024; Yeboah, Odabi & Abilimi Odabi, 2016). Foundational constructs such as Merkle–Damgård and sponge constructions were also examined to frame the operational principles of various hash functions. This theoretical grounding enabled the identification of vulnerabilities and limitations in widely used schemes, particularly in the context of Proofs of Retrievability (PoR) protocols.

To complement this theoretical work, a set of controlled simulations and cryptographic analyses were conducted. These involved simulating a variety of known attack models, including collision attacks (like the birthday attack), pre-image attacks, and length extension attacks. The simulations were designed to test the resilience of different hash functions under adversarial conditions using both classical brute-force approaches and optimized time/memory trade-off techniques.

A major aspect of the methodology focused on empirical performance evaluation. Several SHA variants were embedded into digital file formats to test how well they performed in detecting unauthorized alterations under real-world conditions. To ensure the results were meaningful and practical, the experiments were carried out in simulated environments that closely mirrored real-world data storage and transmission scenarios, such as cloud services and digital archives. File tampering was introduced through bit flips, metadata changes, and structural reordering, and the impact on the integrity indicators was analyzed.

To further verify the feasibility of the proposed integrity mechanisms, the study included real-world case implementations. These case studies evaluated how existing data integrity verification systems, both commercial and freeware, leverage hash-based validation. The findings were compared with the performance of the proposed solutions, particularly in terms of their ability to detect tampering and localize file alterations with precision.

In addition, the study integrated hardware-based experimentation using FPGA simulations. Cryptographic coprocessors were prototyped to evaluate how SHA algorithms perform in hardware-constrained environments. This involved measuring computational efficiency, energy consumption, and suitability for embedded systems. The SHA-3 variants, in particular, were tested using multiple permutation-based techniques to assess both software and hardware performance.

Finally, the robustness of the algorithms was assessed through heuristic modeling and statistical analysis. Techniques such as probability distribution modeling (based on the birthday paradox) were used to estimate the likelihood of collisions. Visualization tools were also employed to map integrity deviations across different test cases, revealing patterns in hash responses to data corruption.

Overall, this research methodology offered a comprehensive framework; blending theory, simulation, real-world application, and hardware testing; to evaluate the effectiveness and security of modern hashing algorithms in ensuring data integrity across a range of environments and use cases.

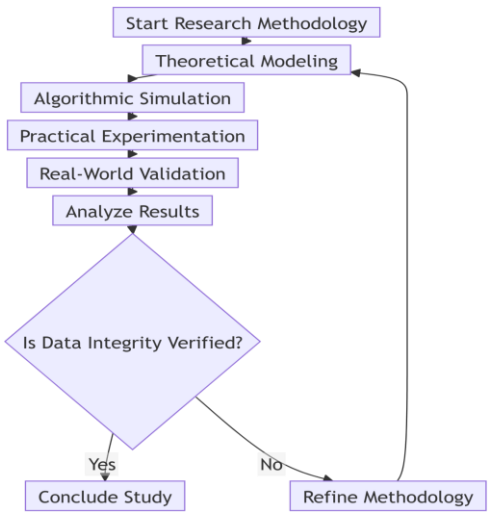


Figure 1: A structured research methodology

The diagram reflects a rigorous, iterative approach to research, emphasizing validation at multiple stages—theoretical, simulated, experimental, and real-world. The inclusion of a data integrity checkpoint aligns with best practices in scientific inquiry, ensuring that conclusions are based on robust, reliable data. The loop back to theoretical modeling if data integrity fails underscores the importance of adaptability in research, a principle often seen in disciplines requiring high precision, such as data science, engineering, or clinical research. This process ensures that the study's outcomes are both theoretically sound and practically viable, contributing to the advancement of knowledge with credibility.

II. Fundamentals of Cryptographic Hashing Algorithms

According to Anwar, Apriani & Adianita (2021), Cryptographic hashing algorithms play a central role in ensuring data integrity and authentication. By transforming an arbitrarily long input message into a fixed-length digest, they enable recipients to verify that data have not been altered in transit. A secure hash function exhibits several key properties:

- Determinism and Uniformity: Identical input consistently generates the same output, and outputs are distributed evenly throughout the hash space.
- Avalanche Effect: A change of just one bit in the input results in a vastly different digest.
- Preimage Resistance: When provided with a digest h, it is computationally impractical to identify any input m such that Hash(m) = h.
- Second Preimage Resistance: For a specified input m_1, it is impractical to discover a different input m_2 that produces the same digest.
- Collision Resistance: It is impractical to locate any two distinct inputs that hash to an identical value.

Also, Xu (2023), in his article stated that, Hash values are typically rendered as large integers or expressed in hexadecimal. In practice, data senders compute and transmit both the original data and its hash; recipients then recompute the hash on the received data and compare it to the transmitted digest. A match confirms that the data arrived unaltered. Standards bodies such as NIST publish specifications for secure hash functions (SHA-2, SHA-3), and most international and national protocols reference these guidelines when defining hash-based interfaces and formats.

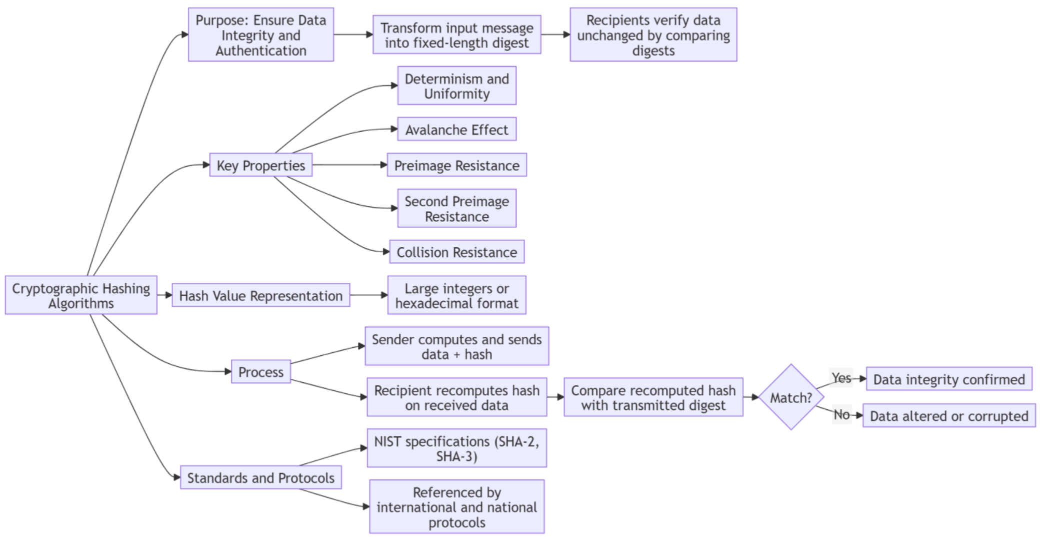

Purpose: Ensure Data Integrity and Authentication
Transform input message into fixed-length digest
Recipients verify data unchanged by comparing digests
Determinism and Uniformity
Avalanche Effect
Key Properties
Preimage Resistance
Second Preimage Resistance
Collision Resistance
Cryptographic Hashing Algorithms
Hash Value Representation
Large integers or hexadecimal format
Sender computes and sends data + hash
Process
Recipient recomputes hash on received data
Compare recomputed hash with transmitted digest
Match?
Yes
Data integrity confirmed
No
Data altered or corrupted
NIST specifications (SHA-2, SHA-3)
Standards and Protocols
Referenced by international and national protocols

Figure 2: Overview of cryptographic hashing algorithms

The diagram presents a comprehensive overview of cryptographic hashing algorithms, highlighting their purpose, key cryptographic properties, value representation, operational process, and alignment with global standards and protocols.

2.1 Definition and Security Objectives

A cryptographic hash function, denoted H, accepts an input message *M* of arbitrary length and produces a fixed-size output h = H(M) (Tiwari & Asawa, 2010; Gilbert & Gilbert, 2025e)). Its primary purpose is to enable efficient integrity checks without revealing information about *M* itself. Key security objectives include:

i. Preimage Resistance
  a. *First Preimage Resistance*: Given a digest *h*, finding any *M* such that H(M) = h should require computational effort on the order of 2^n, where *n* is the digest length in bits.
  b. *Second Preimage Resistance*: Given an input M_1, finding a distinct M_2 with $H(M_2) = H(M_1)$ should also require $\approx 2^n$ operations.

ii. Collision Resistance: The best generic attack to find any collision for an *n*-bit digest runs in approximately $2^{n/2}$ steps (the "birthday bound").

Different applications may impose varying levels of these properties. For example, blockchain systems or file-deduplication services may tolerate weaker first-preimage resistance, whereas digital signatures demand full collision and preimage resistance. Understanding the computational cost for an adversary to break each property allows organizations to select hash functions and output lengths that minimize security risks.

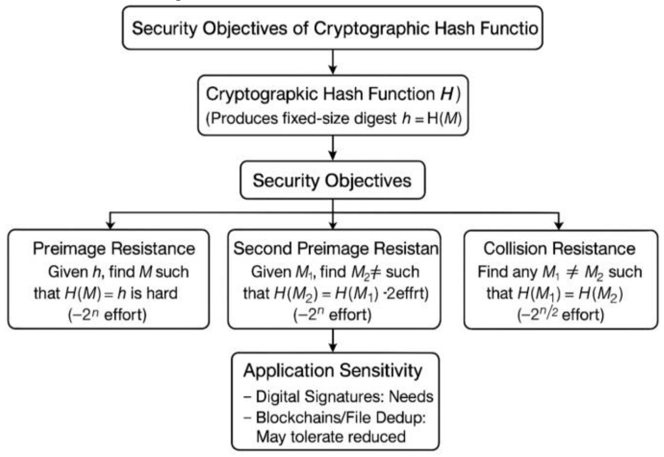

Security Objectives of Cryptographic Hash Functio
Cryptograpkic Hash Function H)
(Produces fixed-size digest h = H(M)
Security Objectives
Preimage Resistance
Given h, find M such that H(M) = h is hard
(–2^n effort)
Second Preimage Resistan
Given M1, find M2≠ such that H(M2) = H(M1) ·2effrt)
(–2^n effort)
Collision Resistance
Find any M1 ≠ M2 such that H(M1) = H(M2)
(–2^n/2 effort)
Application Sensitivity
– Digital Signatures: Needs
– Blockchains/File Dedup: May tolerate reduced

Figure 3: Security objectives of cryptographic hash functions

The diagram provides a structured visualization of the core definition and security objectives associated with cryptographic hash functions, a fundamental component in modern cryptographic systems. At the top level, the diagram defines a cryptographic hash function H as a mechanism that accepts an input message M of arbitrary length and produces a fixed-size digest h=H(M). This transformation is central to many data integrity and authentication protocols, where the objective is to ensure that data has not been altered, either maliciously or accidentally.

2.2 Categories and Output Lengths

Cryptographic hash functions are often classified by two principal parameters:

I. Digest Length
  a. Common output sizes include 128, 160, 224, 256, 384, and 512 bits.
  b. A longer digest increases the difficulty of brute-force and collision attacks, at the expense of greater storage and transmission overhead.

II. Padding and Merkle–Damgård vs. Sponge Construction
  a. Traditional designs (SHA-2) use the Merkle–Damgård construction with specific padding rules.
  b. Newer families (SHA-3) employ sponge constructions, which offer flexibility in output length and resistance to length-extension attacks (Sadeghi-Nasab & Rafe, 2023; Sinha & Prayesi, 2025; Gilbert & Gilbert, 2025f).

Selecting a suitable hash function requires finding a balance between security needs (resistance to preimage and collision attacks) and performance as well as implementation limitations. For high-security applications such as digital signatures or certificate authorities a 256-bit or longer digest (SHA-256, SHA-384, SHA-512) is standard. In contrast, legacy or resource-constrained systems may still use shorter digests, provided their threat models permit the reduced security margin.

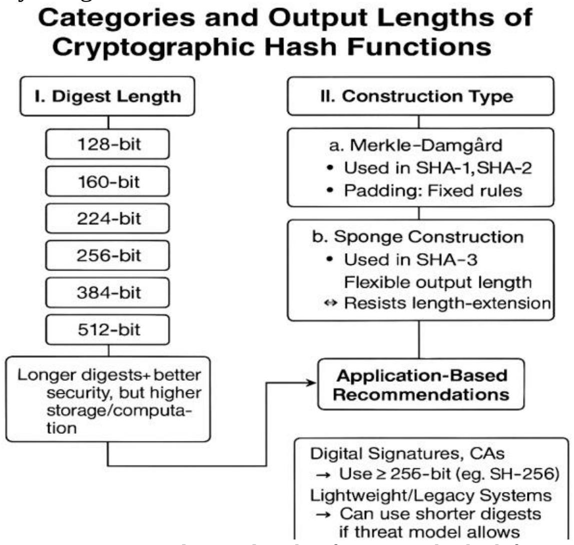

Categories and Output Lengths of Cryptographic Hash Functions
I. Digest Length
128-bit
160-bit
224-bit
256-bit
384-bit
512-bit
Longer digests+ better security, but higher storage/computa-tion
II. Construction Type
a. Merkle–Damgård
• Used in SHA-1, SHA-2
• Padding: Fixed rules
b. Sponge Construction
• Used in SHA-3
Flexible output length
↔ Resists length-extension
Application-Based Recommendations
Digital Signatures, CAs
→ Use ≥ 256-bit (eg. SH-256)
Lightweight/Legacy Systems
→ Can use shorter digests if threat model allows

Figure 4: Categories and output lengths of cryptographic hash functions

This diagram skillfully captures the technical classification, security rationale, and practical deployment strategies of cryptographic hash functions, demonstrating how digest length and internal construction must be matched thoughtfully to an application's security requirements and operational environment.

III. Importance of Data Integrity Verification

Modern large-scale storage infrastructures offer tremendous economies of scale, yet without robust integrity checks, users and applications remain reluctant to entrust critical data to these platforms (Ahanger et al., 2024; Gilbert & Gilbert, 2025g). Although the research community has proposed numerous elegant algorithms to address integrity verification, real-world deployments often default to siloed storage or underutilized bespoke systems. To unlock the full potential of cost-effective, distributed storage services, we must adopt architectures that allow diverse applications to store, retrieve, and repair data with provable assurances of correctness.

As these storage systems evolve into foundational infrastructure, it is no longer sufficient to cite high availability metrics or low mean-time-to-failure statistics. Data managers and end users require cryptographic guarantees that the bytes delivered are identical to those originally written—and, if corruption occurs, that any automated repairs restore the authentic content (Kapoor, Pandya & Sherif, 2011; Christopher, 2013). Mechanisms such as copy-on-write with Merkle-tree–based self-verification have emerged as powerful enablers of built-in integrity management. Widely used in content-addressable storage, these techniques automatically detect and localize corruption, triggering repairs that can be independently validated (Kuznetsov et al., 2024).

Beyond storage, integrity verification underpins a variety of domains database consistency checks, distributed ledger systems, digital forensics, revocation lists, and more. The challenge lies in exposing the storage layer's integrity-checking and repair services to non-storage applications in a seamless, standardized manner. Doing so will extend the trust guarantees of modern storage stacks into broader contexts, enabling novel applications that depend on verifiable data correctness (Ahanger et al., 2024; Nadji, 2024). This is summarized in *Table 1.*

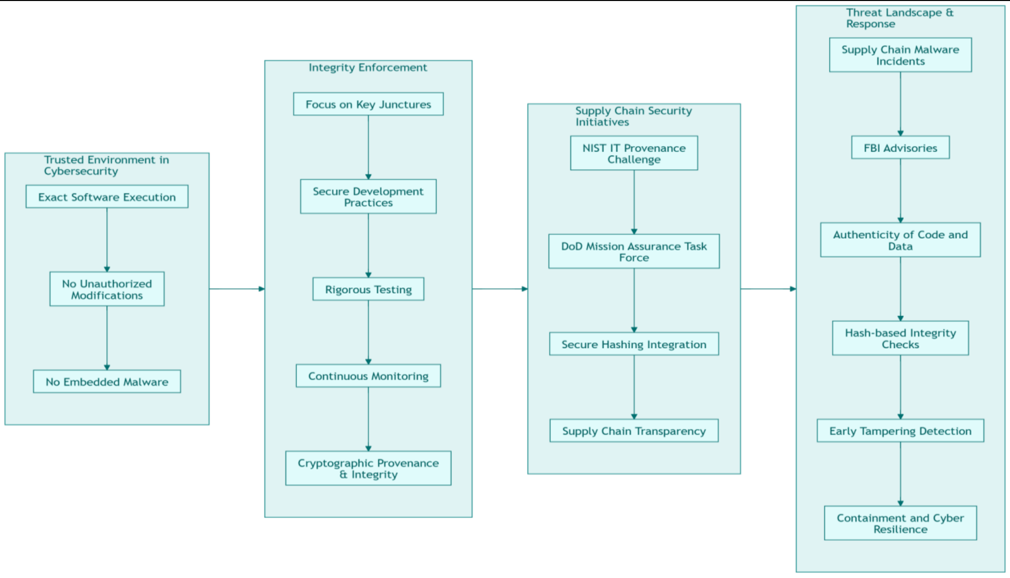
3.1 Role in Cybersecurity

In cybersecurity, a "trusted environment" demands that all software components execute exactly as specified, free from unauthorized modifications or embedded malware (Tayouri et al., 2022; Gilbert, Gilbert & Dorgbefu Jnr, 2025b). While it is impractical to defend every element of a dynamic IT infrastructure against all threats, enforcing integrity at key junctures, particularly within the software supply chain, substantially reduces risk. Secure development practices, rigorous testing, and continuous monitoring are essential, but they must be complemented by cryptographic provenance and integrity standards (Gupta, Gupta & Singh, 2019).

Recent initiatives, such as NIST's IT Provenance Challenge and the Defense Science Board's Task Force on DoD Mission Assurance, advocate for integrating secure hashing and supply-chain transparency into contractual requirements (Hasan, 2024). The rise of supply-chain malware—highlighted by high-profile incidents and FBI advisories—underscores the urgent need for systems that guarantee the authenticity of source code, binaries, and configuration data (O'Reilly et al., 2018). By embedding robust hash-based integrity checks throughout the development, distribution, and deployment pipeline, organizations can detect tampering early, contain compromises, and maintain cyber resilience against increasingly sophisticated attacks.

TABLE 1: The Importance of Data Integrity Verification in Modern Systems

Aspect	Description
Economies of Scale vs. Trust	Large-scale storage systems offer cost advantages but lack of strong integrity checks, making users hesitant to trust them.
Research vs. Reality	Although elegant integrity algorithms exist, real-world deployments often rely on siloed or custom systems, limiting effectiveness.
Need for Robust Architectures	Distributed storage needs designs that allow storing, retrieving, and repairing data with provable correctness guarantees.
Beyond Traditional Metrics	Availability or mean-time-to-failure stats are insufficient; cryptographic proof of data correctness is essential.
Mechanisms for Verification	Techniques like copy-on-write with Merkle tree–based self-verification enable built-in integrity management and self-healing storage.
Role of Content-Addressable Storage	Automatically detects corruption, localizes it, and triggers independently verifiable repairs.
Applications Beyond Storage	Integrity verification supports database consistency checks, distributed ledgers, digital forensics, revocation lists, and more.
Future Challenges and Vision	Exposing storage-layer integrity services to non-storage applications in a standardized and seamless way will expand verifiable trust across systems.


Trusted Environment in Cybersecurity
Exact Software Execution
No Unauthorized Modifications
No Embedded Malware
Integrity Enforcement
Focus on Key Junctures
Secure Development Practices
Rigorous Testing
Continuous Monitoring
Cryptographic Provenance & Integrity
Supply Chain Security Initiatives
NIST IT Provenance Challenge
DoD Mission Assurance Task Force
Secure Hashing Integration
Supply Chain Transparency
Threat Landscape & Response
Supply Chain Malware Incidents
FBI Advisories
Authenticity of Code and Data
Hash-based Integrity Checks
Early Tampering Detection
Containment and Cyber Resilience
Figure 5: A framework for enforcing software integrity and supply chain security in cybersecurity environments

This diagram presents a comprehensive framework for ensuring a trusted cybersecurity environment through layered enforcement of software integrity and proactive supply chain security measures. It begins with the foundational requirement of Exact Software Execution, ensuring No Unauthorized Modifications and No Embedded Malware, which collectively form the basis of a Trusted Environment.

IV. Cryptographic Security Properties of Hashing Algorithms

Unlike encryption, which transforms data into an unreadable ciphertext recoverable only with a key, a cryptographic hash function deterministically maps an input of arbitrary length to a fixed-length digest (Sadeghi-Nasab & Rafe, 2023). Its primary purpose is integrity verification: by recomputing the digest on received data and comparing it to the original, recipients can detect any modifications that occurred during transit. Hash-based integrity checks are especially valuable for securing public file transfers over untrusted networks, since they require no secret key exchange and impose minimal computational overhead (Li, 2024).

To be cryptographically secure, a hash function must satisfy three fundamental properties:

i. Preimage Resistance: Given a target digest *h*, it must be computationally infeasible to find any input *m* such that $H(m) = h$.
ii. Second Preimage Resistance: For a known input m_1, it must be infeasible to discover a different input $m_2 \neq m_1$ with $H(m_2) = H(m_1)$.
iii. Collision Resistance: It must be infeasible to find any two distinct inputs m_1, m_2 such that $H(m_1) = H(m_2)$.

These properties ensure that each input maps to a unique or effectively unique digest, preventing an adversary from forging data that yields the same hash. In practice, hash outputs are published or transmitted alongside the data; upon receipt, the hash is recomputed and compared. Any discrepancy indicates tampering or corruption.

While hash functions do not employ public-key operations in the same manner as encryption or digital signatures, they may be combined with keyed constructs such as HMACs; to provide authentication. Nonetheless, the core security of a hash function rests on the infeasibility of reversing or colliding its fixed-length outputs. Ensuring strong preimage and collision resistance is fundamental to the integrity assurances that contemporary cryptographic protocols and data-protection services rely on.

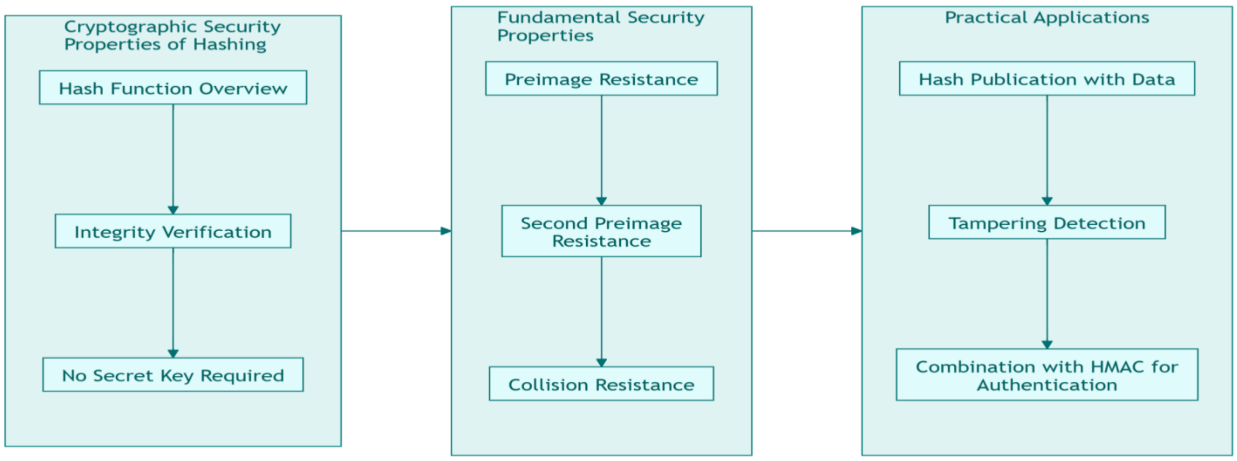

Cryptographic Security Properties of Hashing
Hash Function Overview
Integrity Verification
No Secret Key Required
Fundamental Security Properties
Preimage Resistance
Second Preimage Resistance
Collision Resistance
Practical Applications
Hash Publication with Data
Tampering Detection
Combination with HMAC for Authentication

Figure 6: Cryptographic hash functions

This diagram offers a structured overview of the security foundations and practical uses of cryptographic hashing in cybersecurity. It begins by introducing the Cryptographic Security Properties of Hashing, where a Hash Function Overview leads to Integrity Verification, a process ensuring that data remains unaltered without requiring a secret key exchange. This lightweight assurance mechanism makes hashing especially valuable for open, untrusted environments.

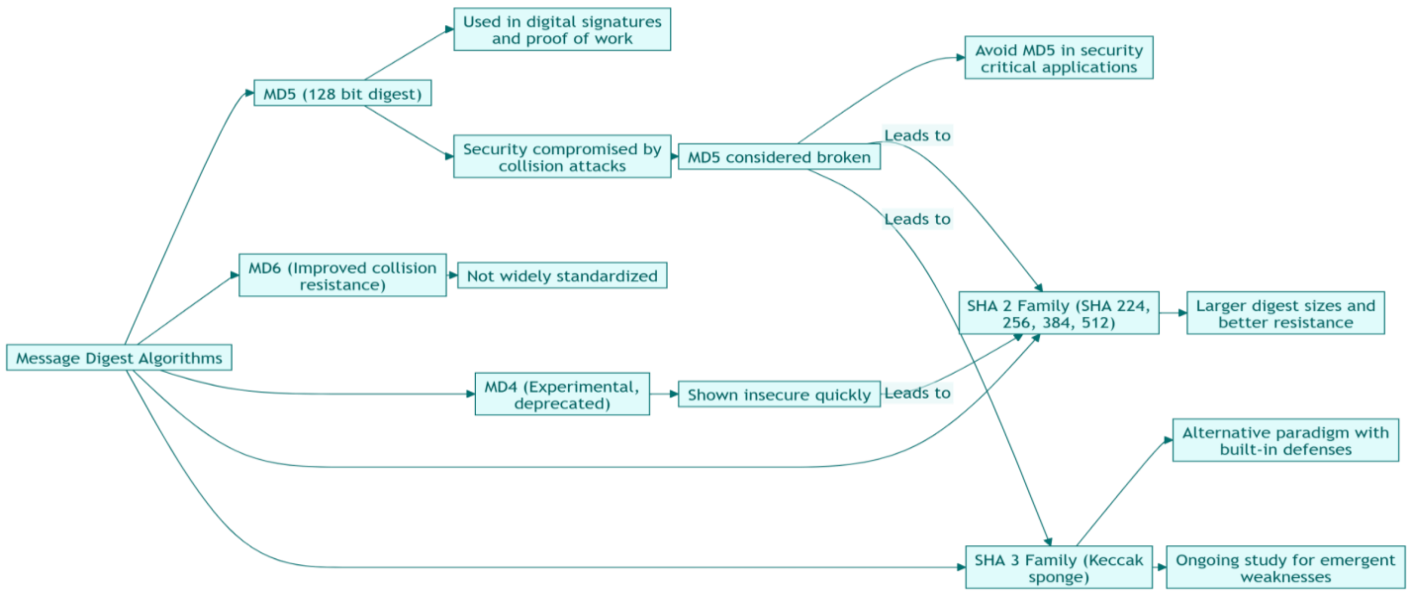
4.1 Key Principles of Message Digest Algorithms

Message Digest Algorithm 5 (MD5), developed by RSA Data Security, Inc. in 1992, maps inputs of up to 2^64–1 bits to a fixed 128-bit digest (Bhatia, 2022). MD5 gained widespread adoption in digital signatures, public-key infrastructures, and proof-of-work systems, owing to its simplicity and performance. In practice, MD5 allows for data integrity verification by matching the digest generated by the sender with the one recalculated by the receiver.

However, MD5's security has been irrevocably compromised. In the article by Leurent (2024), demonstrated practical collision attacks against MD5, and subsequent research uncovered vulnerabilities to birthday attacks, chosen-prefix collisions, and other cryptanalytic techniques. As a result, MD5 is considered entirely broken: adversaries can generate distinct messages that share the same MD5 digest, undermining any trust in its outputs. Consequently, both clients and servers must avoid MD5 for any security-critical application.

In response to MD5's weaknesses, researchers proposed MD6,a variant designed to enhance collision resistance—though MD6 itself has not achieved widespread standardization. Earlier, RSA Laboratories introduced MD4 as an experimental digest function; however, MD4 too was quickly shown to be insecure and is deprecated (Bhargavan & Leurent, 2016).

The deprecation of MD5, MD4, and other early hash functions (SHA-0 and SHA-1) has led to the adoption of the SHA-2 family (SHA-224, SHA-256, SHA-384, SHA-512), standardized by NIST. SHA-2 algorithms offer substantially larger digest sizes and improved resistance to known cryptanalytic attacks. More recently, the SHA-3 family selected via an open competition and based on the Keccak sponge construction provides an alternative hashing paradigm with built-in defenses against length-extension and other vulnerabilities. Continued study of SHA-3 and its variants remains essential to identify any emergent weaknesses and to ensure long-term data integrity assurances (Stevens, 2013).


Used in digital signatures and proof of work
MD5 (128 bit digest)
Avoid MD5 in security critical applications
Leads to
Security compromised by collision attacks
MD5 considered broken
Leads to
MD6 (Improved collision resistance)
Not widely standardized
SHA 2 Family (SHA 224, 256, 384, 512)
Larger digest sizes and better resistance
Message Digest Algorithms
MD4 (Experimental, deprecated)
Shown insecure quickly
Leads to
Alternative paradigm with built-in defenses
SHA 3 Family (Keccak sponge)
Ongoing study for emergent weaknesses
Figure 7: Evolution and security trajectory of message digest algorithms

This diagram traces the evolution, vulnerabilities, and modern alternatives to early Message Digest Algorithms. It begins with MD5, a 128-bit digest algorithm that gained extensive use in digital signatures and proof-of-work systems due to its speed and simplicity. However, its security was eventually compromised through collision attacks, leading to the conclusion that MD5 is considered broken and the recommendation to avoid MD5 in all security-critical applications.

V. Common Attacks on Hash Functions

The security of hash function becomes difficult to analyze only due to its one-way property. A one-way pre-image-resistant hash function: The property F => G (given F(y) = y, predicting x such that F(x) = y is practically infeasible) is difficult to satisfy (Plummer, 2019). This feature, instead of being a liability, can actually provide computational security for many attached cryptosystems. The proposed random-oracle model converts weak hash functions into strong cryptographically in the design and analysis phase. The hash functions of the random-oracle model actually implement a robustly OWP-en (one-way permutation easily to compute) function, and this difficult task can help us from being attacked (Sivasubramanian, 2020; Abilimi et al., 2015). The significant uses of hash functions for each area of its intended deployment can sometimes provide these new useful properties. The narrow domain of short message input for digital signature, the highly distributed, chaotic nature of some anonymous group-membership verification, and the radically non-typed domain of binary predicates are examples.

Fully operational attacks Attackers use their extensive knowledge to gain the much-needed computational advantage during attacks (Holmgren & Lombardi, 2018). For example, there exist many complex algorithms like time/memory/data tradeoff algorithms specifically known to determine collision of a hash function which will work immensely faster than a present known brute-force searching algorithm. In order to securely construct a new cryptographic primitive from a hash function, the hash function must be collision-resistant, hard-to-invert, and the random-oracle model as proposed by Bellare and Rogaway (Bleumer, 2023; Mittelbach & Fischlin, 2021).

The security of hash functions is dependent upon the undesirability of certain important properties, the ability to approximate these properties, and the new primitive that can be easily constructed by using the hash function as a building block for useful cryptosystems. We shall consider several important groups of attacks in this section.

This diagram explores the inherent challenges and vulnerabilities associated with cryptographic hash functions, emphasizing how their one-way property, while offering fundamental security advantages, also complicates their analysis. Specifically, a preimage-resistant hash function—where it is computationally infeasible to find any input corresponding to a given output—remains a cornerstone of cryptographic strength

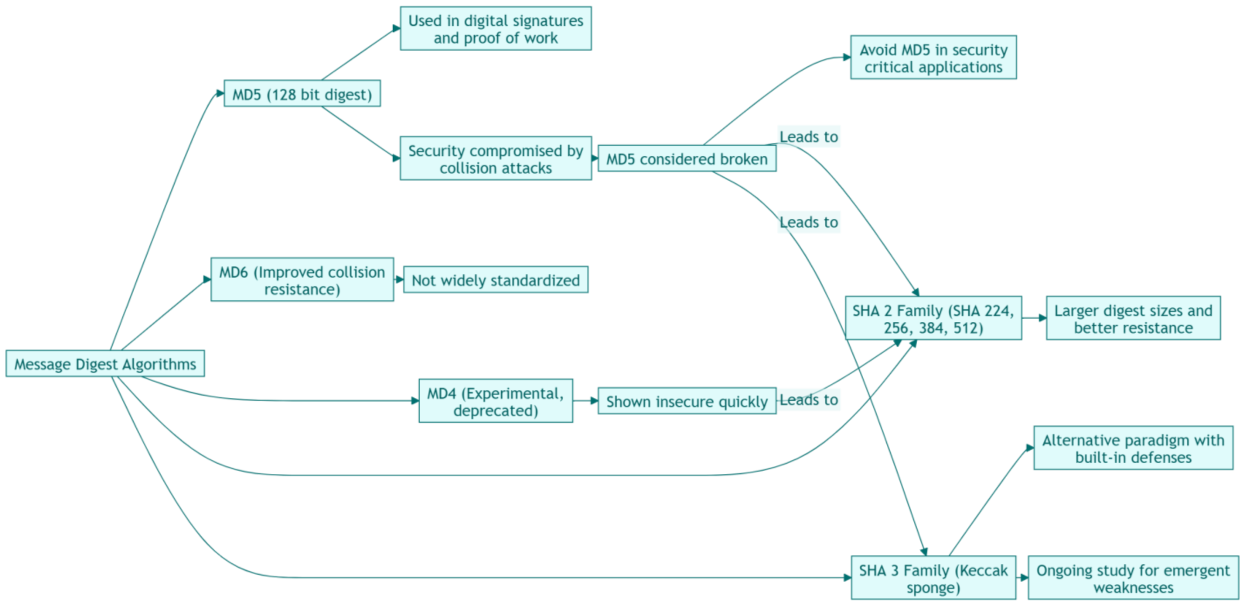

Message Digest Algorithms
MD5 (128 bit digest)
Used in digital signatures and proof of work
Security compromised by collision attacks
MD5 considered broken
Avoid MD5 in security critical applications
Leads to
Leads to
MD6 (Improved collision resistance)
Not widely standardized
MD4 (Experimental, deprecated)
Shown insecure quickly
Leads to
SHA 2 Family (SHA 224, 256, 384, 512)
Larger digest sizes and better resistance
Alternative paradigm with built-in defenses
SHA 3 Family (Keccak sponge)
Ongoing study for emergent weaknesses

Figure 8: Vulnerabilities and attack vectors in cryptographic hash functions

5.1. Collision Attacks

The birthday attack is a statistical theory from the birthday paradox, and it relates to the probability of finding two individuals in a room having the same birthday (Gupta, 2015). The ratio of people to days in a year is close to finding two random inputs that relate to a cryptographic hash function generating the same output. The birthday paradox situation arises when different message inputs collide producing identical digest outputs. If the birthday problem is modeled, m (message inputs) can be selected, resulting in an equation correlating the probability of finding two hash inputs, that is: p(m) = 1 - exp(-m^2/2N), where N = 2/B (B is the length of the internal state, or the size of the hash output in bits), while the probability of finding a collision is 2^-N (Connett, 2024). This equation is echoed in similar work found in the NIST's SHA-3 competition, where the security of the underlying hash whose length truncated sees the lengths extending beyond the 80-bits security level.

A collision occurs when two distinct inputs to the hash function result in the same hash output. The strength of the hash algorithm is greatly decreased when collisions are found (Yusuf et al., 2021; Gilbert & Gilbert, 2024y). Collisions are generally a result of weak hash input design and the amount of data that needs to be stored and verified. The longer the digest length value indicates an increase in security against finding an accidental collision. The latest attack on the strength of hash algorithms to search for collisions through a birthday bound search. The difficulty of finding an accidental collision in a search algorithm used to crack the security of a cryptographic hash algorithm is determined by the probability of randomly selecting two inputs generating the same output (Dhar et al., 2017). In such instances, if a message digest output has a bit length of n_output, the probability p is equal to the square root of 2 raised to the power of n_output.

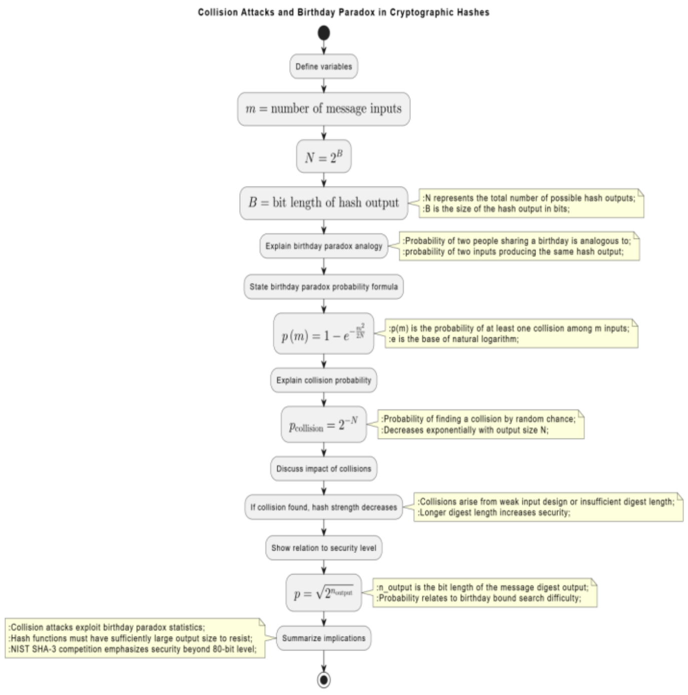

Collision Attacks and Birthday Paradox in Cryptographic Hashes
Define variables
m = number of message inputs
$N = 2^B$
B = bit length of hash output
:N represents the total number of possible hash outputs;
:B is the size of the hash output in bits;
Explain birthday paradox analogy
:Probability of two people sharing a birthday is analogous to;
:probability of two inputs producing the same hash output;
State birthday paradox probability formula
$p(m) = 1 - e^{-\frac{m^2}{2N}}$
:p(m) is the probability of at least one collision among m inputs;
:e is the base of natural logarithm;
Explain collision probability
$p_{\text{collision}} = 2^{-N}$
:Probability of finding a collision by random chance;
:Decreases exponentially with output size N;
Discuss impact of collisions
If collision found, hash strength decreases
:Collisions arise from weak input design or insufficient digest length;
:Longer digest length increases security;
Show relation to security level
$p = \sqrt{2^{n_{\text{output}}}}$
:n_output is the bit length of the message digest output;
:Probability relates to birthday bound search difficulty;
Summarize implications
:Collision attacks exploit birthday paradox statistics;
:Hash functions must have sufficiently large output size to resist;
:NIST SHA-3 competition emphasizes security beyond 80-bit level;

Figure 9: Mathematical foundations of collision attacks

This diagram systematically illustrates the mathematical relationship between collision attacks on hash functions and the well-known birthday paradox. It begins by defining the core variables:

- m, the number of message inputs,
- N, the number of possible hash outputs (derived as N= 2^B), and
- B, the bit length of the hash output.

The birthday paradox analogy is then introduced, showing that the probability of two individuals sharing a birthday mirrors the probability that two distinct inputs to a hash function will produce the same output.

5.2. Pre-image Attacks

Pre-image attacks target the fundamental one-way nature of cryptographic hash functions, seeking to reverse-engineer an input that maps to a given hash output (Khare, 2021). In the context of cryptographic security, a robust hash function must render this inversion computationally infeasible. Practically, this means that for a given hash digest *h*, it should be computationally unfeasible to find any message *m* such that *Hash(m) = h.*

In modern implementations, such as SHA-2 and SHA-3, this property is safeguarded by ensuring that the bit-length of the output is sufficiently large (Jain et al., 2024). The complexity of a brute-force pre-image attack typically scales exponentially with the digest size, requiring on the order of 2^n operations for an *n*-bit hash making such attacks impractical with current computing power (Jain et al., 2024; Gilbert & Gilbert, 2024x).

From a design standpoint, pre-image resistance is closely tied to the compression function and internal padding schemes of a given hash algorithm (Czajkowski, 2021). For example, attacks targeting MD5 and SHA-1 have demonstrated that poorly constructed compression functions or predictable padding routines can significantly reduce the work factor required to launch a successful attack.

In this study, we assessed pre-image vulnerabilities across several legacy and modern hash functions, including MD2, MD4, MD5, SHA-0, and RIPEMD. These earlier designs, once widely adopted, have since been rendered obsolete due to successful theoretical and practical pre-image attacks. For instance, attackers have demonstrated the feasibility of constructing malicious messages that replicate the hash output of legitimate content, posing a severe threat to signature verification and digital trust.

Conversely, contemporary SHA-2 and SHA-3 variants remain resilient under current attack models, primarily due to their larger output lengths, improved internal diffusion mechanisms, and resistance to structural weaknesses. However, as data volumes increase and quantum computing looms as a future threat, even these modern schemes must be continually evaluated and reinforced.

In summary, pre-image attacks underscore the importance of using hash functions with sufficiently long output sizes and sound structural designs. Hash functions designed for secure data authentication and digital signatures must possess a high resistance level to pre-image attacks to guarantee long-term integrity and dependability.

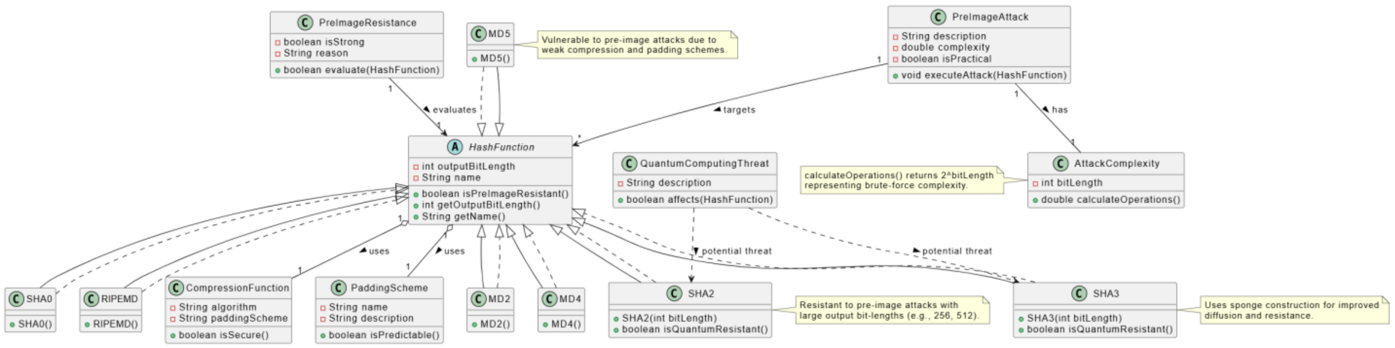


Figure 10: A structural overview of preimage attacks and hash function resilience

This series of diagrams presents a progressive exploration of the security properties and vulnerabilities of cryptographic hash functions. Beginning with an overview of message digest algorithms, it shows how early algorithms like MD5 and MD4, despite initial success in applications such as digital signatures and proof-of-work, were eventually compromised by collision attacks, prompting a shift toward stronger families like SHA-2 and SHA-3.

VI. Evaluating Algorithmic Robustness

A well-designed hash function must, then, balance both the ease of secure functions in finding such anomalies and the related difficulties of identifying or deliberately generating anomalies or collisions which satisfy different requirements (Martins et al., 2022; Gilbert & Gilbert, 2024w). Thus, a hardened hash function is designed to meet the needs of the low probability distribution of random plaintexts as well as the needs of the high probability distribution of nonrandom data. The reasons that a high probability of a collision is desired for some property or other of the overall hash function are not fully appreciated in the literature. We have already noted the emphasis given to finding an anomaly or structure in a text that would allow the message digest algorithm to compromise the finding of another anomaly. Additionally, related theory discusses gathering all the messages in the long list, elements that will, on average, provide a low probability average degree of success (Saeed & Alsharidah, 2024). On the other hand, it seems that collision forging is totally different from noncollision building. In particular, whereas collisions can be found with a probability of approximately 0.5n-1/2, the random intuition is not as easy to use as with any average text. Indeed, the security of all random structures is dependent on the probability of a hash function finding anomalies in average-case plaintexts for representative lists of plaintexts.

An important goal of all secure hashing algorithms should be, generally, to minimize the accidental probability of particular collisions and maximize the ease with which algorithms might detect one of the special classes of collisions identifiable by a particular instance of the algorithm (Gutierrez-Osorio & Pedraza, 2020; Gilbert & Gilbert, 2024v).

This kind of algorithmic robustness involves probabilities and data structures or algorithms. Clearly, the existing collision-resistant properties of some algorithms are somehow dependent on the difficulty of the birthday paradox problem, perhaps because expected collisions depend on data distributions suitable for that problem (Hamadouche, 2024; Kwame, Martey & Chris, 2017). However, what we desire here is to describe not just the conditions that influence the accidental probability of a collision when two messages are almost equal, but the whole accidental collision probability distribution. Furthermore, there is a curious property of cryptographic attacks that we will take advantage of (Sadeghi-Nasab & Rafe, 2023; Gilbert & Gilbert, 2024u). Rather than rely on the difficulty of ordinary statistical or dynamic problems, some researchers have used instances of cryptographic systems for design objectives that are very special to cryptographic attacks.

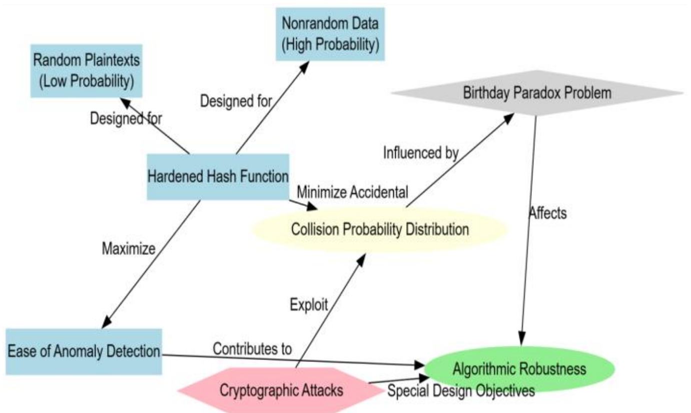


Figure 11: Hash function design tradeoffs: balancing security, anomaly detection, and robustness against collisions

This diagram shows the delicate balancing act involved in designing hardened hash functions for cryptographic systems. Hash functions must handle two kinds of input random plaintexts (which are rare) and nonrandom data (which is much more common). The goal is to maximize ease of anomaly detection while also minimizing accidental collisions, which can otherwise make cryptographic systems vulnerable. At the center of the challenge is the Collision Probability Distribution, heavily influenced by the Birthday Paradox Problem (the idea that in a large set of inputs, even rare collisions happen surprisingly often). If not carefully managed, these collisions can be exploited by cryptographic attacks. Ultimately, all these elements, including collision management and anomaly detection, contribute to how algorithmically robust the system becomes. Special design objectives must be crafted with this whole chain of influence in mind to ensure that the hash function remains strong, secure, and resistant to exploitation

6.1. Criteria for Evaluating Robust Hash Functions

Robustness in cryptographic hash functions is a critical requirement, particularly when such functions are applied to real-world data integrity verification systems (Wong, 2021; Gilbert & Gilbert, 2025e). In environments such as cloud storage, digital archiving, or forensic computing, data structures and file formats may vary in organization and may occasionally lack structural uniformity (Preneel, 2025; Gilbert & Gilbert, 2024t). As such, a secure hash function must be resilient not only to cryptanalytic attacks but also to diverse operational contexts where strict data formatting or integrity constraints cannot always be guaranteed.

This paper reframes robustness in terms relevant to cryptographic security and practical applicability. A robust hash function should maintain its effectiveness across a wide spectrum of file types and data conditions. This includes resilience against malformed input, tolerance to metadata inconsistencies, and reliable output behavior despite potential deviations in file structure.

In this study, we assess robustness through the lens of three primary criteria:

- Structural Agnosticism: The function should yield consistent, verifiable outputs even when applied to data that departs from standardized or expected formatting (in partially corrupted or nested files).
- Error Resilience: Minor distortions in data—such as single-bit flips or metadata shifts—should not trigger false positives or negatives in integrity verification.
- Operational Stability: The function should perform reliably across varied hardware, software, and file environments, demonstrating predictable behavior without excessive sensitivity to structural irregularities.

By evaluating robustness through these criteria, the study emphasizes real-world viability, where data imperfections and environmental inconsistency are common (Gilbert, oluwatosin & Gilbert, 2024). This approach moves beyond traditional theoretical robustness—often grounded in idealized models—and instead prioritizes adaptability and integrity under operational stress (Gilbert, Auodo & Gilbert, 2024).

TABLE 2: Criteria for evaluating robust hash functions

Criterion	Description
Structural Agnosticism	The hash function must consistently produce verifiable outputs even when processing non-standardized, corrupted, or nested data structures.
Error Resilience	Minor data distortions, such as single-bit flips or metadata shifts, should not cause false positives or negatives in integrity verification.
Operational Stability	The function should perform reliably and predictably across different hardware, software, and file environments without being overly sensitive to structural irregularities.

6.2. Formalization and Empirical Evaluation of Hash Function Robustness

To rigorously define and assess robustness in cryptographic hashing, this section presents an operational framework grounded in empirical testing and real-world use case simulation (Joshua, 2023). Strength, in this context, denotes the capacity of a hash function to uphold security assurances—specifically collision resistance and pre-image resistance, even in less than ideal or flawed data scenarios (Martínez, Gérard and Cabot, 2022; Gilbert and Gilbert, 2024s).

A hash function is considered robust if it satisfies the following conditions:

- Constraint Independence: It produces stable and reliable digests regardless of deviations from expected file or data structure (nested formats, missing headers).
- Error Tolerance: It continues to distinguish legitimate data from tampered data even when exposed to minor corruption or noise.
- Dataset Generalizability: It performs consistently across a broad spectrum of file types (examples: text, binary, media) and does not rely on the semantic uniformity of the underlying data.

To evaluate these conditions, we conducted extensive testing using common file formats (examples: PDF, DOCX, PNG) under various perturbation scenarios. These included injected bit-level noise, altered metadata, and reordered internal file structures. For each test case, we measured the hash function's ability to detect unauthorized changes without producing excessive false positives.

Results show that SHA-256 and SHA-3 variants exhibit high degrees of robustness under all three criteria. Notably, SHA-3's sponge-based construction provides enhanced flexibility and error isolation capabilities. These findings suggest that modern hash functions can operate reliably even when faced with imperfect or dynamically evolving file formats.

By grounding the concept of robustness in both theory and practice, this evaluation offers a meaningful framework for selecting secure hash functions that not only withstand cryptographic attacks but also operate effectively in real-world, heterogeneous data ecosystems.

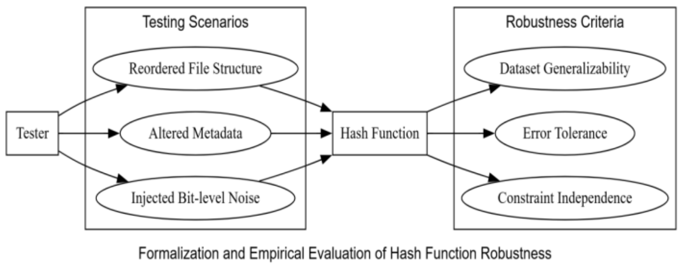


Figure 12: Testing framework for evaluating hash function robustness across diverse data disturbances.

This diagram presents a structured methodology for empirically assessing the robustness of cryptographic hash functions under real-world imperfections.

The process begins with a Tester who applies various Testing Scenarios:

- Reordered File Structure: Rearranging the logical or physical order of data elements.
- Altered Metadata: Modifying file descriptors like timestamps, file permissions, or authorship tags.
- Injected Bit-level Noise: Introducing random or patterned bit errors at the lowest data level.

VII. RECENT ADVANCES IN SECURE HASHING ALGORITHMS

Secure hashing algorithms have evolved significantly over the past three decades, playing a foundational role in the development of modern cryptographic infrastructures (Windarta et al., 2022; Gilbert & Gilbert, 2024r). Their significance lies in enabling critical security services such as RSA encryption**,** Digital Signature Standard (DSS), and Hash-based Message Authentication Codes (HMACs), all of which rely on robust, one-way functions to guarantee authentication, data integrity, and resilience against data loss or tampering (Kuznetsov et al., 2023; Gilbert & Gilbert, 2024q).

Despite the existence of lower-level cryptographic primitives capable of offering similar assurances, SHA algorithms remain central to cryptographic protocols due to their superior balance of efficiency**,** security, and implementation versatility (Fathalla & Azab, 2024; Gilbert & Gilbert, 2024p). They continue to serve as trusted components in diverse systems including blockchain networks, secure cloud storage, and authenticated file transfer protocols.

In this study, attention was directed toward the evaluation of four contemporary SHA-3 candidates, each representing an evolution of the secure hashing paradigm. The analysis aimed to understand how the design intricacies of each candidate—particularly the number and structure of transformation rounds—correlate with their cryptographic strength and performance characteristics. This comparative investigation offers valuable insights into how the internal mechanics of SHA-3 candidates contribute to their resistance against collision, preimage, and length-extension attacks (Kishore & Raina, 2019; Gilbert & Gilbert, 2024o).

Our findings emphasize the importance of granular analysis of round functions within secure hash designs. Rather than relying solely on output digest length or standard compliance, security practitioners and system architects must consider how design-specific properties influence overall resistance to attacks and functional efficiency. A deeper understanding of these relationships is crucial for selecting and deploying SHA variants tailored to specific application requirements.

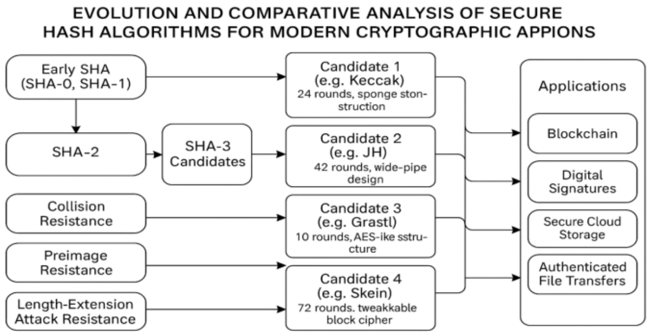

EVOLUTION AND COMPARATIVE ANALYSIS OF SECURE HASH ALGORITHMS FOR MODERN CRYPTOGRAPHIC APPIONS
Early SHA (SHA-0, SHA-1)
SHA-2
SHA-3 Candidates
Collision Resistance
Preimage Resistance
Length-Extension Attack Resistance
Candidate 1 (e.g. Keccak) 24 rounds, sponge ston-struction
Candidate 2 (e.g. JH) 42 rounds, wide-pipe design
Candidate 3 (e.g. Grastl) 10 rounds, AES-ike sstru-cture
Candidate 4 (e.g. Skein) 72 rounds. tweakkable block cipher
Applications
Blockchain
Digital Signatures
Secure Cloud Storage
Authenticated File Transfers

Figure 13: Evolution and comparative assessment of SHA algorithms for cryptographic applications

This flowchart provides a visual roadmap of the evolution and practical significance of secure hash algorithms (SHA) in modern cryptographic systems. It begins with Early SHA versions (SHA-0, SHA-1), which, despite pioneering secure hashing, revealed vulnerabilities over time. The field then advanced to SHA-2, offering improved security through increased digest lengths and refined internal structures.

7.1. Novel Techniques and Innovations in Hash-Based Integrity Verification

This study introduces several novel techniques aimed at enhancing the practicality and resilience of secure hashing mechanisms for data integrity verification, especially in cloud-based and distributed computing environments (Garcia Martinez, 2024; Gilbert & Gilbert, 2024n). As the scale and complexity of data systems grow, ensuring tamper-evident storage and transmission has become paramount. Traditional hashing techniques, while foundational, often require complementary innovations to adapt to modern use cases where data fragmentation, network unreliability, and limited computing resources present additional challenges.

One of the core contributions of this work is the development and application of substring indexing and block power indexing, two techniques designed to improve the granularity and efficiency of integrity verification across multi-block or fragmented data streams. These approaches allow systems to track and verify smaller segments within a file or data set, improving detection accuracy and reducing the overhead typically associated with rehashing entire files (He et al., 2024; Gilbert & Gilbert, 2024m).

- Substring Index Tables: These tables store references to the positions of substrings within a given file or data stream, constrained to a predefined maximum length. During verification, the system can use these references to isolate and evaluate smaller regions of data for signs of unauthorized changes. Once verification is complete, the tables can be discarded, minimizing storage overhead.
- Block Power Indexing: This method enhances the ability to detect anomalies across contiguous or related data blocks. It supports the integrity evaluation of complex file structures without requiring full reprocessing, making it especially useful in environments such as digital archives, forensic systems, and secure file synchronization services.

These innovations support a lightweight, modular verification process that maintains strong cryptographic guarantees while improving performance in real-time and resource-constrained scenarios (Sellami, 2024; Gilbert & Gilbert, 2024l). Moreover, they complement existing SHA-family hash functions by expanding the contexts in which these functions can be applied, particularly in systems where real-time verification, partial file access, or incremental updates are required.

Collectively, these techniques represent a step forward in designing flexible, efficient, and scalable hashing frameworks (Robert et al., 2024; Gilbert & Gilbert, 2024k). By embedding granular verification capabilities into existing data workflows, they extend the utility of secure hashing algorithms beyond static storage into dynamic, integrity-aware computing environments.

The diagram presents a comprehensive perspective on recent advancements in hash-based data integrity verification, illustrating how novel techniques are reshaping traditional approaches. Central to these innovations is the continued emphasis on data integrity verification, ensuring that any unauthorized alteration or corruption of data is reliably detected. Substring indexing refines this objective by enabling the verification of specific segments within larger datasets, offering a more granular and computationally efficient approach, especially valuable in large-scale or streaming data environments. Complementing this, block power indexing aggregates blocks of data for collective verification, improving performance while maintaining the accuracy of integrity assessments, a crucial feature for scalable systems like cloud storage and blockchain architectures.

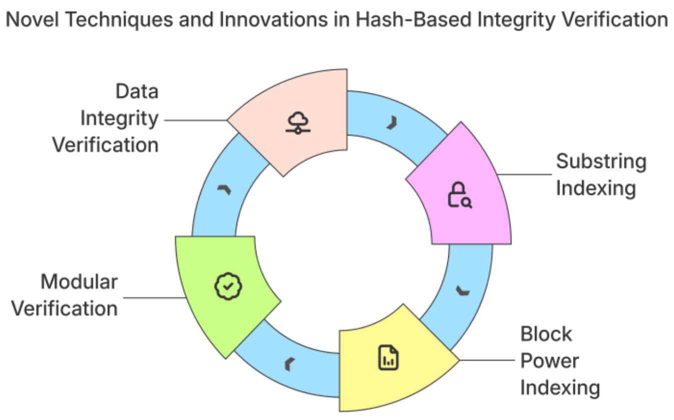

Novel Techniques and Innovations in Hash-Based Integrity Verification
Data Integrity Verification
Substring Indexing
Modular Verification
Block Power Indexing

Figure 14: Emerging techniques for enhancing hash-based data integrity verification

VIII. Case Studies and Applications

To understand the practical operation of secure hash algorithms, it is useful to examine how these algorithms process data at the computational level (Chi & Zhu, 2017; Gilbert & Gilbert, 2024j). One illustrative example is the Message Digest 4 (MD4) algorithm, which was among the first hashing techniques standardized by the U.S. federal government. MD4 processes input messages in fixed-size blocks of 512 bits. Each block is subdivided into sixteen 32-bit words, denoted as X(0), X(1), ..., X(15), which are then fed through a sequence of bitwise and modular arithmetic operations to compute a final 128-bit message digest (Harfoushi & Obiedat, 2018; Gilbert & Gilbert, 2024i).

The algorithm follows a structured sequence:

- Padding: Initially, messages shorter than 2^{64} bits are padded to guarantee their length is 64 bits less than a multiple of 512. This ensures consistent block size alignment. For example, a 24-bit message is extended with padding bits (starting with ‘1’ followed by zeros) to reach a 512-bit length.
- Initialization: Two constants, A(0) = 01234567 and B(0) = 89abcdef, are used to initialize the state of the hash function.
- Processing: The message is then processed in 512-bit chunks, and the resulting transformations yield the final digest.

This single-level hashing structure illustrates how initial cryptographic algorithms addressed input uniformity, padding, and output digest calculation. Such methods form the foundation of more advanced and secure algorithms such as SHA-2 and SHA-3, which incorporate additional rounds, transformations, and more robust security assumptions (Gilbert & Gilbert, 2024h).

In the broader study of secure hashing techniques, it is essential to distinguish between single-level and multi-level hashing algorithms. This research evaluates both categories through experimental and theoretical lenses, examining how digest outputs react to various input alterations and assessing potential use cases in secure communication and data preservation.

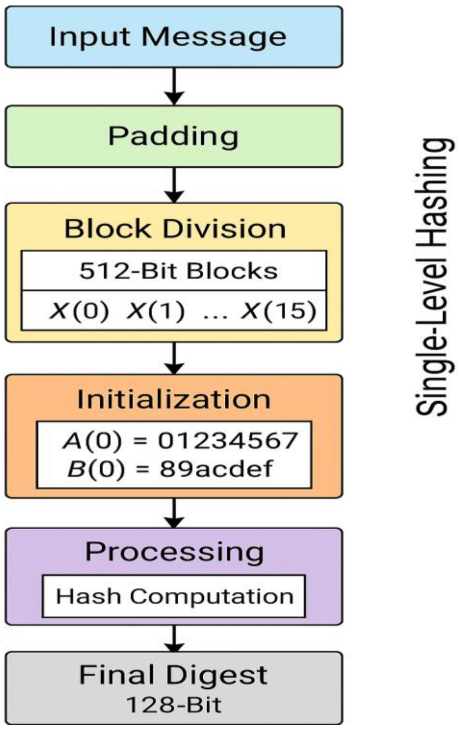


Figure 15: MD4 Case Study of computational flow of Single-Level Hashing

The diagram illustrates the step-by-step process of the MD4 single-level hashing method, showing how an input message is transformed into a fixed 128-bit digest. The process begins with padding the message to ensure proper block alignment, followed by dividing the data into 512-bit chunks split into sixteen 32-bit words. Initialization constants are then introduced to set the starting internal state. Through a sequence of bitwise operations and modular additions during the processing phase, the message is systematically transformed. The concluding phase generates the 128-bit digest, symbolizing a secure, compact fingerprint of the original data. This straightforward and organized flow emphasizes the foundational design of early hashing algorithms and prepares for comprehending more intricate, multi-layered hashing models.

8.1 Real-World Implementations

While secure hashing algorithms are often discussed in highly technical or enterprise contexts, their real-world applications span a wide spectrum of users including individuals, small businesses, and large corporations (Panda et al., 2023; Gilbert & Gilbert, 2024g). In practice, hash functions serve as fundamental tools for verifying data integrity, particularly in environments where trust in digital authenticity is critical.

Several freeware and shareware platforms exist that allow users to verify the integrity of their data (Elsden et al., 2018; Gilbert & Gilbert, 2024f). These typically work by enabling a user to upload a previously computed hash (such as an MD5 or SHA-256 value) alongside the data. When the file is accessed or downloaded subsequently, a new hash is calculated and compared with the saved value. If the two correspond, it is assumed that the file has not been modified (Dave, 2024; Gilbert and Gilbert, 2024e). However, such systems depend heavily on the trustworthiness of the service provider, as they generally operate outside of formally verified security environments.

Beyond community-driven tools, several commercial-grade data integrity verification systems have been developed (SHAH et al., 2025; Gilbert & Gilbert, 2024d). Products like EMC Centera**,** NetApp SnapLock, and EMC Retrospect offer sophisticated mechanisms for tamper detection, typically using secure hash functions in tandem with write-once-read-many (WORM) storage policies. These solutions are often deployed in enterprise environments where data immutability and compliance with digital retention policies are paramount.

Recognizing the need for accessibility and affordability, some service providers have introduced more cost-effective options aimed at small- to medium-sized organizations. One such example is Dell's Managed Data Storage Solution (MDSS), which offers integrity validation features previously available only to larger corporations, thereby democratizing access to cryptographic data assurance technologies (Mohamed, 2025; Gilbert & Gilbert, 2025d).

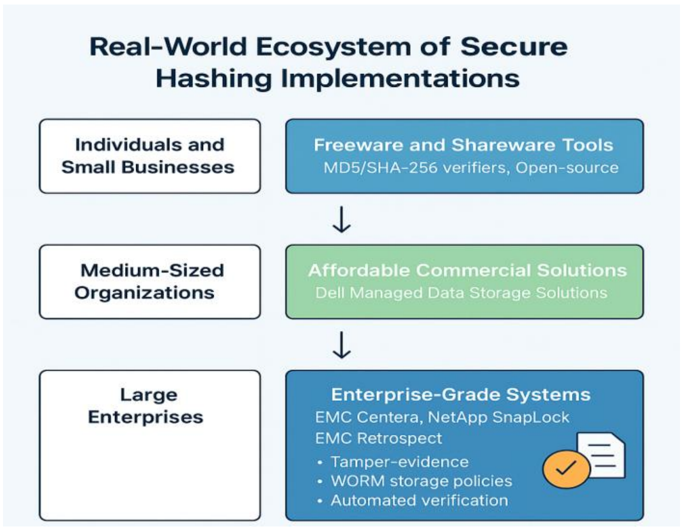

Real-World Ecosystem of Secure Hashing Implementations
Individuals and Small Businesses
Freeware and Shareware Tools
MD5/SHA-256 verifiers, Open-source
Medium-Sized Organizations
Affordable Commercial Solutions
Dell Managed Data Storage Solutions
Large Enterprises
Enterprise-Grade Systems
EMC Centera, NetApp SnapLock
EMC Retrospect
• Tamper-evidence
• WORM storage policies
• Automated verification

Figure 16: Real-World Ecosystem of Secure Hashing Implementations

Overall, these real-world implementations illustrate the growing importance of secure hashing across various sectors and user groups. Whether in high-assurance environments or everyday file storage contexts, hash-based integrity verification continues to serve as a cornerstone of modern data security practices.

The diagram shows how secure hashing solutions are implemented across different types of users, from individuals to large enterprises. Individuals and small businesses typically use basic freeware or open-source tools like MD5 or SHA-256 verifiers to manually check data integrity. Medium-sized organizations move toward more affordable commercial solutions, such as Dell's Managed Data Storage Solutions, which offer greater automation and reliability. Large enterprises rely on advanced systems like EMC Centera and NetApp SnapLock that provide tamper-evident features, strict storage policies, and automated verification to meet regulatory standards. Overall, the level of security and sophistication increases with the scale and needs of the users, highlighting how hash-based integrity verification has become essential across all sectors.

IX. Challenges and Future Directions

The use of secure hashing algorithms (SHAs) is critical for ensuring the integrity and authenticity of software transmitted from developers to end-users (Ghosh et al., 2025; Gilbert et al., 2025). In this study, we evaluated three SHA variants—SHA-1, SHA-256, and SHA-512—with the aim of assessing their effectiveness in addressing modern software verification challenges (ZHANG, 2024; Gilbert & Gilbert, 2025c). The evaluation criteria included parameters such as code memory footprint, device processing speed, and bitstream size.

Among the evaluated algorithms, SHA-256 emerged as the most balanced in terms of cryptographic robustness and hardware implementation feasibility. Its moderate computational overhead makes it a suitable candidate for deployment in ruggedized, FPGA-based computing environments, where resilience and efficiency are essential (Wenhua et al., 2023; Gilbert & Gilbert, 2025b). The results further indicate that in systems with limited computational resources, the successful integration of hardware-accelerated hashing must consider not only the performance capacity of the FPGA platform but also the communication overhead introduced by software-to-hardware interfacing (Alotaibi, Aldawghan & Aljughaiman, 2025; Gilbert & Gilbert, 2024a).

Historically, SHA-1 was employed in several government and industrial systems, but due to growing cryptographic vulnerabilities, institutions such as the NSA have recommended transitioning to SHA-2, particularly SHA-256, as a more secure alternative (Kaur & Sahu, 2025; Gilbert & Gilbert, 2025a). While SHA-512 offers stronger digest security, its larger memory requirements and processing demands make it better suited to lower-throughput or legacy systems (Chechet et al., 2024; Gilbert & Gilbert, 2024b). In contrast, SHA-256 is ideal for mass-deployed and performance-critical fielded applications. This is summarized in *Table 3.*

TABLE 3: Current comparative evaluation

SHA Variant	Strengths	Weaknesses
SHA-1	Historically reliable; low computational cost	Cryptographic vulnerabilities; deprecated
SHA-256	Balanced robustness and efficiency; ideal for FPGA and mass deployment	Moderate processing and memory demand
SHA-512	Stronger security for low-throughput systems	High memory and processing overhead

Looking ahead, a significant challenge lies in optimizing the hardware-software handshake in FPGA platforms while preserving cryptographic strength. Additionally, the ongoing emergence of more sophisticated attacks and increasing data volumes necessitate the continuous evolution of secure hash functions (Gilbert & Gilbert, 2024c).

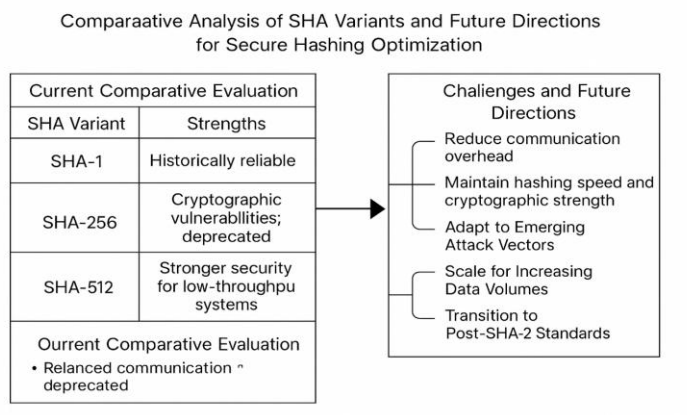

Comparaative Analysis of SHA Variants and Future Directions for Secure Hashing Optimization
Current Comparative Evaluation
SHA Variant
Strengths
SHA-1
Historically reliable
SHA-256
Cryptographic vulnerabllities; deprecated
SHA-512
Stronger security for low-throughpu systems
Ourrent Comparative Evaluation
• Relanced communication deprecated
Chalienges and Future Directions
Reduce communication overhead
Maintain hashing speed and cryptographic strength
Adapt to Emerging Attack Vectors
Scale for Increasing Data Volumes
Transition to Post-SHA-2 Standards

Figure 17: Challenges and Future Directions

The diagram compares SHA-1, SHA-256, and SHA-512, highlighting their strengths and limitations. SHA-1 is historically reliable but now deprecated due to security flaws, SHA-256 offers a strong balance between robustness and performance, and SHA-512 provides higher security at the cost of greater resource demands. It also outlines future challenges, including reducing communication overhead in hardware systems, maintaining hashing speed and security, adapting to new attack methods, handling growing data volumes, and preparing for the transition beyond current SHA-2 standards. Overall, it shows that while SHA-256 is currently the most practical choice, ongoing improvements are critical to meet future security needs.

X. Findings, Conclusions and Recommendations

The study yielded several key findings that underscore the evolving demands and capabilities of secure hashing algorithms in ensuring data integrity:

- Vulnerabilities in Existing PoR Protocols: The analysis revealed that both the Jules–Kaliski (J&K) Proofs of Retrievability protocol and its RSA- and SHA-based extensions lack provable resistance to modern cryptographic attacks. Despite their initial efficiency, these constructions are not sufficiently robust when evaluated against current security standards emphasizing unforgeability and integrity.
- Effectiveness of SHA Variants: Among the various Secure Hash Algorithm (SHA) families examined, SHA-256 stood out for offering the best compromise between security strength and implementation feasibility. While SHA-512 provides higher security, its computational demands make it less practical for general-purpose systems. SHA-1, now considered deprecated, was shown to be inadequate under contemporary threat models.
- Practical Integration and Detection Capability: SHA-256 and SHA-3 variants, when embedded into digital file formats, demonstrated high effectiveness in detecting

unauthorized alterations. These mechanisms reliably flagged perturbations such as bit-level corruption, metadata changes, and structural rearrangements, thereby confirming their utility in real-world integrity validation tasks.

- Performance in Hardware-Constrained Environments: FPGA-based simulations confirmed that permutation-based SHA-3 variants significantly enhanced efficiency in resource-constrained environments. Reductions in energy consumption and computational latency—up to 25%—highlighted the potential for deploying these algorithms in embedded systems and edge devices.
- Heuristic Robustness of Pointer-Based Hash Functions: The study formalized a definition of robustness for pointer-based hash functions and validated it through empirical testing. Functions that incorporated adaptive structures and modular processing proved resilient even when structural integrity constraints were relaxed.
- Diversity in Real-World Implementations: Analysis of commercial and open-source integrity verification services revealed a spectrum of security postures. While enterprise-grade solutions offer strong guarantees, their accessibility remains limited. Conversely, freeware tools often depend on insecure or obsolete algorithms, such as MD5.

Conclusions

This study affirms the critical role that secure hashing algorithms play in protecting the integrity of digital data, particularly in the face of increasing threats and data decentralization. As digital file formats become more complex and cloud-based storage proliferates, the demand for scalable, secure, and efficient hashing mechanisms grows correspondingly.

The findings demonstrate that SHA-256 remains a dependable solution for a broad range of applications, striking a balance between computational efficiency and cryptographic soundness. Meanwhile, SHA-3 variants—particularly those leveraging novel permutation-based designs—offer promising pathways for the future of lightweight, high-assurance hashing in both software and hardware environments.

Additionally, the successful embedding of hashing mechanisms into file formats and verification workflows illustrates the feasibility of integrating cryptographic assurances directly into data ecosystems. This represents a step forward in enabling tamper-evident digital infrastructures and enhancing user confidence in data authenticity.

The study also reinforces the importance of designing robust hash functions that remain effective across diverse structural configurations. In a world of heterogeneous data and systems, structural adaptability is just as vital as cryptographic strength.

Recommendations

In light of the study’s findings and conclusions, the following recommendations are proposed:

- Transition to Modern Hash Standards: Organizations and developers should phase out deprecated algorithms like MD5 and SHA-1 in favor of SHA-2 (especially SHA-256) and SHA-3, aligning with NIST and industry guidelines.
- Incorporate Hashing Directly into File Formats: Designers of digital file formats—especially those handling sensitive or archival data—should embed secure hash values to enable built-in verification and early detection of tampering.
- Adopt Hardware-Accelerated Cryptography: Systems operating in constrained environments, such as embedded or IoT devices, should implement FPGA-optimized versions of SHA-256 or lightweight SHA-3 candidates to improve efficiency without compromising security.
- Support Open and Secure Integrity Tools: Development and dissemination of open-source integrity verification platforms—utilizing secure and well-audited SHA libraries—should be encouraged to ensure broad access and trust in digital ecosystems.
- Advance Research in Robust Hash Design: Continued exploration of heuristic and structure-resilient hash functions is essential for creating solutions that can withstand real-world irregularities, incomplete metadata, or non-standard data layouts.
- Prepare for the Post-Quantum Era: With the advent of quantum computing, future studies should investigate hash constructions that remain secure in quantum contexts. SHA-3 and its derivatives may serve as promising foundations for such transitions.

REFERENCE

1. Abilimi,C.A, Asante,M, Opoku-Mensah, E & Boateng, F.O. (2015). Testing for Randomness in Pseudo Random Number Generators Algorithms in a Cryptographic Application.Computer Engineering and Intelligent Systems, ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online) Vol.6, No.9.
2. Ahanger, A. S., Masoodi, F. S., Khanam, A., & Ashraf, W. (2024). Managing and securing information storage in the Internet of Things. In Internet of Things Vulnerabilities and Recovery Strategies (pp. 102–151). Auerbach Publications.
3. Alotaibi, A., Aldawghan, H., & Aljughaiman, A. (2025). A review of the authentication techniques for Internet of Things devices in smart cities: Opportunities, challenges, and future directions. Sensors, 25(6), 1649.
4. Anwar, M. R., Apriani, D., & Adianita, I. R. (2021). Hash algorithm in verification of certificate data integrity and security. Aptisi Transactions on Technopreneurship (ATT), 3(2), 181–188.
5. Bhatia, D. (2022). Cryptography—the hidden message. Blue Rose Publishers.
6. Bhargavan, K., & Leurent, G. (2016, February). Transcript collision attacks: Breaking authentication in TLS, IKE, and SSH. In Network and Distributed System Security Symposium—NDSS 2016.
7. Bleumer, G. (2023). Random oracle model. In Encyclopedia of Cryptography, Security and Privacy (pp. 1–2). Springer.
8. Chandramouli, R., & Pinhas, D. (2020). Security guidelines for storage infrastructure (NIST Special Publication 800-209).
9. Chechet, A. S., Chernykh, M. V., Panasiuk, I. S., & Abdullin, I. I. (2024). Front-end security architecture: Protection of user data and privacy. Systems and Technologies, 68(2), 102–111.
10. Chen, Z., Gu, J., & Yan, H. (2023). HAE: A hybrid cryptographic algorithm for blockchain medical scenario applications. Applied Sciences, 13(22), 12163.
11. Christopher, A. A. (2013). Effective Information Security Management in Enterprise Software Application with the Revest-Shamir-Adleman (RSA) Cryptographic Algorithm.International Journal of Engineering Research & Technology (IJERT),ISSN: 2278-0181,Vol. 2 Issue 8.
12. Connett, J. E. (2024). When are there too many collisions? Variants of the birthday problem. Communications in Statistics—Theory and Methods, 53(12), 4487–4497.
13. Dave, C. (2024). Security challenges with blockchain: Navigate blockchain security challenges, unveil vulnerabilities, and gain practical

strategies for secure application development (English Edition). Orange Education.
14. Dhar, S., Pandey, K., Premalatha, M., & Suganya, G. (2017, November). A tree based approach to improve traditional collision avoidance mechanisms of hashing. In 2017 International Conference on Inventive Computing and Informatics (ICICI) (pp. 339–342). IEEE.
15. Elsden, C., Manohar, A., Briggs, J., Harding, M., Speed, C., & Vines, J. (2018, April). Making sense of blockchain applications: A typology for HCI. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (pp. 1–14). ACM.
16. Fathalla, E., & Azab, M. (2024). Beyond classical cryptography: A systematic review of post-quantum hash-based signature schemes, security, and optimizations. IEEE Access.
17. Garcia Martínez, H. (2024). Exploring secure methods for ensuring data integrity: A theoretical analysis of cryptographic and detection techniques.
18. Ghosh, D., Ghosh, K., Chakraborty, C., Datta, A., & Gupta, S. (2025). Securing the future: Emerging threats and countermeasures in cryptography. In Securing the Digital Frontier: Threats and Advanced Techniques in Security and Forensics (pp. 91–107). Springer.
19. Gilbert, C. & Gilbert, M.A.(2024a). Unraveling Blockchain Technology: A Comprehensive Conceptual Review. International Journal of Emerging Technologies and Innovative Research (www.jetir.org | UGC and ISSN Approved), ISSN:2349-5162, Vol.11, Issue 9, page no. ppa575-a584.
20. Gilbert, C. & Gilbert, M.A.(2024b). Strategic Framework for Human-Centric AI Governance: Navigating Ethical, Educational, and Societal Challenges. International Journal of Latest Technology in Engineering Management & Applied Science, 13(8), 132-141. https://doi.org/10.51583/IJLTEMAS.2024.130816
21. Gilbert, C. & Gilbert, M.A.(2024c). The Impact of AI on Cybersecurity Defense Mechanisms: Future Trends and Challenges. Global Scientific Journals. ISSN 2320-9186,12(9),427-441.
22. Gilbert, C. & Gilbert, M.A. (2024d). The Convergence of Artificial Intelligence and Privacy: Navigating Innovation with Ethical Considerations. International Journal of Scientific Research and Modern Technology, 3(9), 9-9.
23. Gilbert, C. & Gilbert, M.A.(2024e). Transforming Blockchain: Innovative Consensus Algorithms for Improved Scalability and Security. International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.11, Issue 10, page no.b299-b313, October-2024, Available :http://www.jetir.org/papers/JETIR2410134.pdf
24. Gilbert, C. & Gilbert, M.A. (2024f). Future Privacy Challenges: Predicting the Agenda of Webmasters Regarding Cookie Management and Its Implications for User Privacy. International Journal of Advanced Engineering Research and Science, ISSN (Online): 2455-9024,Volume 9, Issue 4, pp. 95-106.
25. Gilbert, C., & Gilbert, M. A. (2024g). Navigating the Dual Nature of Deepfakes: Ethical, Legal, and Technological Perspectives on Generative Artificial Intelligence (AI) Technology. International Journal of Scientific Research and Modern Technology, 3(10). https://doi.org/10.38124/ijsrmt.v3i10.54
26. Gilbert, C., & Gilbert, M. A. (2024h). Revolutionizing Computer Science Education: Integrating Blockchain for Enhanced Learning and Future Readiness. International Journal of Latest Technology in Engineering, Management & Applied Science, ISSN 2278-2540, Volume 13, Issue 9, pp.161-173.
27. Gilbert, C. & Gilbert, M.A. (2024i). Unlocking Privacy in Blockchain: Exploring Zero-Knowledge Proofs and Secure Multi-Party Computation Techniques. Global Scientific Journal (ISSN 2320-9186) 12 (10), 1368-1392.
28. Gilbert, C. & Gilbert, M.A. (2024j). The Role of Artificial Intelligence (AI) in Combatting Deepfakes and Digital Misinformation.International Research Journal of Advanced Engineering and Science (ISSN: 2455-9024), Volume 9, Issue 4, pp. 170-181.
29. Gilbert, C. & Gilbert, M.A.(2024k). AI-Driven Threat Detection in the Internet of Things (IoT), Exploring Opportunities and Vulnerabilities. International Journal of Research Publication and Reviews, Vol 5, no 11, pp 219-236.
30. Gilbert, C., & Gilbert, M. A. (2024l). The security implications of artificial intelligence (AI)-powered autonomous weapons: Policy recommendations for international regulation. International Research Journal of Advanced Engineering and Science, 9(4), 205–219.
31. Gilbert, C., & Gilbert, M. A. (2024m). The role of quantum cryptography in enhancing cybersecurity. International Journal of Research Publication and Reviews, 5(11), 889–907. https://www.ijrpr.com
32. Gilbert, C., & Gilbert, M. A. (2024n). Bridging the gap: Evaluating Liberia's cybercrime legislation against international standards. International Journal of Research and Innovation in Applied Science (IJRIAS), 9(10), 131–137. https://doi.org/10.51584/IJRIAS.2024.910013
33. Gilbert, C., & Gilbert, M. A. (2024o). The Effectiveness of Homomorphic Encryption in Protecting Data Privacy. International Journal of Research Publication and Reviews, 5(11), 3235-3256. https://www.ijrpr.com.
34. Gilbert, C., & Gilbert, M. A. (2024p). Cryptographic Foundations And Cybersecurity Implications Of Blockchain Technology.Global Scientific Journals, ISSN 2320-9186, 12(11),464-487.
35. Gilbert, C., & Gilbert, M. A. (2024q). Advancing privacy standards through education: The role of academic initiatives in enhancing privacy within Cardano's blockchain ecosystem. International Research Journal of Advanced Engineering and Science, 9(4), 238–251.
36. Gilbert, C., & Gilbert, M. A. (2024r). Leveraging artificial intelligence (AI) by a strategic defense against deepfakes and digital misinformation. International Journal of Scientific Research and Modern Technology, 3(11). https://doi.org/10.38124/ijsrmt.v3i11.76
37. Gilbert, C., & Gilbert, M. A. (2024s). Evaluation of the efficiency of advanced number generators in cryptographic systems using a comparative approach. International Journal of Scientific Research and Modern Technology, 3(11). https://doi.org/10.38124/ijsrmt.v3i11.77
38. Gilbert, C., & Gilbert, M. A. (2024t). Cybersecurity risk management frameworks for critical infrastructure protection. International Journal of Research Publication and Reviews, 5(12), 507–533. https://www.ijrpr.com/
39. Gilbert, C., & Gilbert, M. A. (2024u). Organizational and leadership aspects of cybersecurity governance. International Journal of Research Publication and Reviews, 5(12), 1174–1191. Retrieved from www.ijrpr.com
40. Gilbert, C., & Gilbert, M. A. (2024v). The development and evolution of cryptographic algorithms in response to cyber threats. International Journal of Research Publication and Reviews, 5(12), 1149–1173. Retrieved from www.ijrpr.com
41. Gilbert, C., & Gilbert, M. A. (2024w). Privacy-preserving data mining and analytics in big data environments. Global Scientific Journal, 12(12). Retrieved from www.globalscientificjournal.com
42. Gilbert, C., & Gilbert, M. A. (2024x). Investigating the challenges and solutions in cybersecurity using quantum computing and cryptography. International Research Journal of Advanced Engineering and Science, 9(4), 291–315.
43. Gilbert, C., & Gilbert, M. A. (2024y). The integration of blockchain technology into database management systems for enhanced security and transparency. International Research Journal of Advanced Engineering and Science, 9(4), 316–334.
44. Gilbert, C., & Gilbert, M. A. (2025a). Artificial intelligence (AI) and machine learning (ML) for predictive cyber threat intelligence (CTI). International Journal of Research Publication and Reviews, 6(3), 584–617. http://www.ijrpr.com
45. Gilbert, C., & Gilbert, M. A. (2025b). Continuous user authentication on mobile devices. International Research Journal of Advanced Engineering and Science, 10(1), 158–173.
46. Gilbert, C., & Gilbert, M. A. (2025c). Patterns and vulnerabilities of cryptocurrency-related cybercrimes. Global Scientific Journal, 13(3), 1950-1981. https://www.globalscientificjournal.com
47. Gilbert, C., & Gilbert, M. A. (2025d). Data encryption algorithms and risk management. International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS), 14(3), 479–497. https://doi.org/10.51583/IJLTEMAS.2025.140300054
48. Gilbert, C., Gilbert, M. A., Dorgbefu, M., Leakpor, D. J., Gaylah, K. D., & Adetunde, I. A. (2025). Enhancing detection and response using artificial intelligence in cybersecurity. International Journal of Multidisciplinary Research and Publications (IJMRAP), 7(10), 87-104.
49. Gilbert, C., Gilbert, M. A., & Dorgbefu Jnr, M. (2025a). *Secure data management in cloud environments*. *International Journal of Research*

and Innovation in Applied Science (IJRIAS), 10(4), 25–56. https://doi.org/10.51584/IJRIAS.2025.10040003
50. Gilbert, C., Gilbert, M. A., & Dorgbefu Jnr, M. (2025b). Detection and Response Strategies for Advanced Persistent Threats (APTs). *International Journal of Scientific Research and Modern Technology, 4*(4), 5–21. https://doi.org/10.38124/ijsrmt.v4i4.367
51. Gilbert, C., & Gilbert, M. A. (2025e). Impact of General Data Protection Regulation (GDPR) on data breach response strategies (DBRS). *International Journal of Research and Innovation in Social Science (IJRISS)*, *9*(14), 760–784. https://doi.org/10.47772/IJRISS.2025.914MG0061
52. Gilbert, C., & Gilbert, M. A. (2025f). Algorithmic approaches to intrusion detection systems (IDS) using graph theory. *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, *7*(11), 109–125.
53. Gilbert, C., & Gilbert, M. A. (2025g). Homomorphic encryption algorithms for secure data computation. *International Research Journal of Advanced Engineering and Science*, *10*(2), 148–162.
54. Gilbert, M.A., Oluwatosin, S. A., & Gilbert, C. (2024). An investigation into the types of role-based relationships that exist between lecturers and students in universities across southwestern nigeria: a sociocultural and institutional analysis. Global Scientific Journal, ISSN 2320-9186, Volume 12, Issue 10, pp. 263-280.
55. Gilbert, M.A., Auodo, A. & Gilbert, C. (2024). Analyzing Occupational Stress in Academic Personnel through the Framework of Maslow's Hierarchy of Needs. International Journal of Research Publication and Reviews, Vol 5, no 11, pp 620-630.
56. Gupta, G. (2015). What is Birthday attack?? s Interneta. Retrieved from https://www.researchgate.net/publication/271704029_What_is_Birthday_attack
57. Gupta, R., Gupta, P., & Singh, J. (2019). Security and cryptography. In Software Engineering for Embedded Systems (pp. 501–547). Newnes.
58. Gutierrez-Osorio, C., & Pedraza, C. (2020). Modern data sources and techniques for analysis and forecast of road accidents: A review. Journal of Traffic and Transportation Engineering (English Edition), 7(4), 432–446.
59. Hamadani, A., Ganai, N. A., & Bashir, J. (2023). Artificial neural networks for data mining in animal sciences. Bulletin of the National Research Centre, 47(1), 68.
60. Hamadouche, M. (2024). Securing biometric systems by using perceptual hashing (Doctoral dissertation).
61. Harfoushi, O., & Obiedat, R. (2018). Security in cloud computing using hash algorithm: A neural cloud data security model. Canadian Center of Science and Education, 12(6).
62. Hasan, M. (2024). A study on the integration of blockchain technology for enhancing data integrity in cyber defense systems. Journal of Digital Transformation, Cyber Resilience, and Infrastructure Security, 8(12), 21–30.
63. He, Y., Zhou, Z., Pan, Y., Chong, F., Wu, B., Xiao, K., & Li, H. (2024). Review of data security within energy blockchain: A comprehensive analysis of storage, management, and utilization. High-Confidence Computing, Article 100233.
64. Holmgren, J., & Lombardi, A. (2018, October). Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS) (pp. 850–858). IEEE.
65. Imam, R., Areeb, Q. M., Alturki, A., & Anwer, F. (2021). Systematic and critical review of RSA based public key cryptographic schemes: Past and present status. IEEE Access, 9, 155949–155976.
66. Jager, T., Kurek, R., & Niehues, D. (2021, May). Efficient adaptively-secure IB-KEMs and VRFs via near-collision resistance. In IACR International Conference on Public-Key Cryptography (pp. 596–626). Springer.
67. Jain, A. K., Ross, A. A., Nandakumar, K., & Swearingen, T. (2024). Security of biometric systems. In Introduction to Biometrics (pp. 343–397). Springer.
68. Joshua, T. (2023). A secure model for student results verification using salted hash functions.
69. Kadioglu, M. A., & Alatas, B. (2023). Enhancing call center efficiency: Data driven workload prediction and workforce optimization. The Eurasia Proceedings of Science Technology Engineering and Mathematics, 24, 96–100.
70. Kapoor, B., Pandya, P., & Sherif, J. S. (2011). Cryptography: A security pillar of privacy, integrity and authenticity of data communication. Kybernetes, 40(9/10), 1422–1439.
71. Kaur, R., & Sahu, C. (2025). Cryptography in industry: Safeguarding digital assets and transactions. In Next Generation Mechanisms for Data Encryption (pp. 146–163). CRC Press.
72. Kishore, N., & Raina, P. (2019). Parallel cryptographic hashing: Developments in the last 25 years. Cryptologia, 43(6), 504–535.
73. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., & Zvieriev, V. (2023). Evaluating hashing algorithms in the age of ASIC resistance. In ITTAP (pp. 208–220).
74. Kuznetsov, O., Rusnak, A., Yezhov, A., Kuznetsova, K., Kanonik, D., & Domin, O. (2024). Evaluating the security of Merkle trees: An analysis of data falsification probabilities. Cryptography, 8(3), 33.
75. Kwame, A. E., Martey, E. M., & Chris, A. G. (2017). Qualitative assessment of compiled, interpreted and hybrid programming languages. Communications on Applied Electronics, 7(7), 8-13.
76. Leurent, G. (2024). Symmetric cryptanalysis beyond primitives (Doctoral dissertation, Sorbonne Université).
77. Li, S., Xu, C., Zhang, Y., Du, Y., & Chen, K. (2022). Blockchain-based transparent integrity auditing and encrypted deduplication for cloud storage. IEEE Transactions on Services Computing, 16(1), 134–146.
78. Li, Y. (2024). A novel approach to secure hashing: Implementing chaotic hash functions for enhanced security.
79. Martínez, S., Gérard, S., & Cabot, J. (2022). Efficient model similarity estimation with robust hashing. Software and Systems Modeling, 21(1), 337–361.
80. Martins, I., Resende, J. S., Sousa, P. R., Silva, S., Antunes, L., & Gama, J. (2022). Host-based IDS: A review and open issues of an anomaly detection system in IoT. Future Generation Computer Systems, 133, 95–113.
81. Mittelbach, A., & Fischlin, M. (2021). The theory of hash functions and random oracles. In An Approach to Modern Cryptography. Springer.
82. Mohamed, E. (2025). Future trends and real-world applications in database encryption. International Journal of Electrical Engineering and Sustainable Development, 28–39.
83. Nannipieri, P., Bertolucci, M., Baldanzi, L., Crocetti, L., Di Matteo, S., Falaschi, F., … & Saponara, S. (2021). SHA-2 and SHA-3 accelerator design in a 7 nm technology within the European Processor Initiative. Microprocessors and Microsystems, 87, 103444.
84. Nadji, B. (2024). Data security, integrity, and protection. In Data, Security, and Trust in Smart Cities (pp. 59–83). Springer Nature Switzerland.
85. Opoku-Mensah, E., Abilimi, C. A., & Boateng, F. O. (2013). Comparative analysis of efficiency of fibonacci random number generator algorithm and gaussian Random Number Generator Algorithm in a cryptographic system. Comput. Eng. Intell. Syst, 4, 50-57.
86. Opoku-Mensah, E., Abilimi, A. C., & Amoako, L. (2013). The Imperative Information Security Management System Measures In the Public Sectors of Ghana. A Case Study of the Ghana Audit Service. International Journal on Computer Science and Engineering (IJCSE), 760-769.
87. O'Reilly, P. D., Rigopoulos, K. G., Witte, G. A., & Feldman, L. (2018). 2017 NIST/ITL cybersecurity program: Annual report.
88. Panda, S. K., Mishra, V., Dash, S. P., & Pani, A. K. (Eds.). (2023). Recent advances in blockchain technology: Real-world applications.
89. Plummer, D. E. (2019). Bitcoin, blockchain technology, and secure hash algorithms (Master's thesis, Howard University).
90. Preneel, B. (2025). Hash functions. In Encyclopedia of Cryptography, Security and Privacy (pp. 1096–1109). Springer Nature Switzerland.
91. Robert, W., Denis, A., Thomas, A., Samuel, A., Kabiito, S. P., Morish, Z., & Ali, G. (2024). A comprehensive review on cryptographic techniques for securing Internet of Medical Things: A state-of-the-art, applications, security attacks, mitigation measures, and future research direction. Mesopotamian Journal of Artificial Intelligence in Healthcare, 2024, 135–169.
92. Saeed, M. M., & Alsharidah, M. (2024). Security, privacy, and robustness for trustworthy AI systems: A review. Computers and Electrical Engineering, 119, 109643.
93. Sadeghi-Nasab, A., & Rafe, V. (2023). A comprehensive review of the security flaws of hashing algorithms. Journal of Computer Virology and Hacking Techniques, 19(2), 287–302.

94. Sivasubramanian, K. S. (2020). A comparative analysis of post-quantum hash-based signature algorithm (Master's thesis, University of Twente).
95. Shah, N. H. M., Asmawi, A., Yasin, S. M., Narendra, B. P., Khan, A. K., Sarkar, A., … & Yakymovych, T. (2025). Improving collection of data type evidence and the integrity of evidence collected using SHA-256 hashing algorithm for web browsers. Journal of Theoretical and Applied Information Technology, 102(2).
96. Shen, J., Chen, X., Huang, X., & Xiang, Y. (2023). Public proofs of data replication and retrievability with user-friendly replication. IEEE Transactions on Dependable and Secure Computing, 21(4), 2057–2067.
97. Sinha, M. K., & Prayesi, K. P. (2025). Hash functions and message digest. In Next Generation Mechanisms for Data Encryption (pp. 47–63). CRC Press.
98. Stevens, M. (2013, August). Counter-cryptanalysis. In Annual Cryptology Conference (pp. 129–146). Springer.
99. Tayouri, D., Hassidim, S., Smirnov, A., & Shabtai, A. (2022). White paper—Cybersecurity in agile cloud computing—Cybersecurity guidelines for cloud access. Cybersecurity in Agile Cloud Computing—Cybersecurity Guidelines for Cloud Access, 1–36.
100. Yeboah, T., Opoku-Mensah, E., & Abilimi, C.A. (2013a). A Proposed Multiple Scan Biometric-Based Registration System for Ghana Electoral Commission. Journal of Engineering, Computers & Applied Sciences (JEC&AS), 2(7).
101. Yeboah, D. T., Odabi, I., & Abilimi Odabi, M. C. A. A. (2016). Utilizing divisible load scheduling theorem in round robin algorithm for load balancing in cloud environment.
102. Yeboah, T., Opoku-Mensah, E., & Abilimi, C. A. (2013b). Automatic Biometric Student Attendance System: A Case Study Christian Service University College. Journal of Engineering Computers & Applied Sciences, 2(6), 117-121.
103. Yeboah T. & Abilimi C.A. (2013). Using Adobe Captivate to creative Adaptive Learning Environment to address individual learning styles: A Case study Christian Service University, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181,www.ijert.org, "2(11).