International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(8s) 2026 pp. 810-825
https://doi.org/10.70917/ijcisim-2026-3313
Article

Cryptographic Hash Functions: A Comparative Study of Algorithmic Efficiency and Security Solutions

Urvashi Chaudhary[1], Amit Bhatnagar[2]

[1]Department of Computer Science and Engineering, IFTM University, Moradabad, Uttar Pradesh, India.
[2]School of Computer Science and Applications, IFTM University, Moradabad, Uttar Pradesh, India.

Abstract: — Cryptographic hash algorithms are regarded as tools that validate the data integrity in the file. In this globalised world, it is important to note the strengths and weaknesses of these functions. This research presents a comparative classification of cryptographic hash functions. Moreover, it assesses their performance, security and efficiency. This research paper analyzes the different usefulness of these algorithms in essential fields like blockchain, authentication, data management, digital forensics, etc. We perform a detailed performance analysis on the cost of computational speed, robustness to attack, and adaptability to emerging threats such as quantum computing. In addition, this study fills important gaps in the existing literature through the establishment of a systematic taxonomy based on building type, standardization, interoperability and sensitivity to inputs. According to our study, a framework that can be useful for the researchers and practitioners in choosing the right hash algorithms for specific operations. This research throws light on real world situation of security, efficiency and future-proofing activities as such any revealed or uncovered issue can contribute to the development of digital

Keywords: Cryptographic Hashing; Chaos-Based Hash Functions; Data Security; Algorithm Evaluation; Digital Cryp-tography.

1. INTRODUCTION

Cryptographic hashing is an essential building block of digital security to maintain data integrity and authenticity. It is widely used in secure communications, data storage, and blockchain technology [1]. The security and efficient operation of classical approaches, including SHA-256 and MD5, which have become industry standards [3], are being challenged by increasing computing power and new threats.

Chaotic hash functions have become one of the most researched areas as a solution. Through chaos theory functions, these functions are sensitive to initial conditions in addition to being unpredictable, which is unlike conventional cryptography techniques [2]. Incorporating chaos theory into a hashing algorithm makes the algorithm more complex, and helps to build a more secure protocol for a chaotic environment [4].

The existing literature provides limited discussion on a comprehensive evaluation and classification of traditional and chaos-based functions across applications despite a considerable amount of research on conventional hashing. This paper fills this gap by offering a systematic categorization and performance evaluation of these hashing techniques.

The rest of the article is structured as follows: section I reviews the literature, while section II provides an overview of relevant hash functions and their application areas. Section IV presents the assessment and classification of different hashing algorithms, including chaos-based algorithms, their performance and applicability. Section V provides a categorization based on parameters, while Section VI mentions the latest NIST recommendations and analyzes future scenarios. In the last section (Section VII), we summarises our contributions and suggests future research directions in the field of chaos-based cryptographic hashing.

2. LITERATURE REVIEW

The work of Diffie, Hellman [5] and Merkle [6] in 1976 paved the way for public-key cryptography. The subsequent suggestion Rabin made was to first hash the message, then sign it. This has since become the standard in which performance, security and efficiency are well catered to [7]. Matyas et al. [8], which described constructions based on block ciphers, and the construction of MDC-2 at IBM in 1987 [9]. MD2, MD4, and MD5 was founded by Rivest between 1988 and 1991, which powered the US. The Secure Hash Standard (SHA) was first developed by the National Institute of Standards and Technology (NIST) in 1993. Later on, SHA-1 was released.

Nonetheless, a setback struck the cryptographic community when vulnerabilities arose. The cryptanalysis of MD4, MD5 and SHA-0 was successfully accomplished in the 1990s and early 2000s whose breakthrough was the work Xiaoyun Wang who in 2005 initiated collision attacks on SHA-1 at [18–25]. As a result, NIST re-evaluated its existing standards, resulting in the development of the SHA-2 family and the launch of the public competition resulting in SHA-3 [26, 27]. The creation of RIPEMD, HAVAL and others, as well as the block cipher-based Whirlpool, added to the list of cryptographic algorithms [21, 28–30].

Research in recent years has focused more on optimizing and designing for specific applications. A remarkable instance is Poseidon and its successor Poseidon2, which were designed for ZK (zero-knowledge) proofs systems [31, 34]. The field of innovation has extended into new quantum-resistant schemes [32], IoT-centric security [33, 39], and specialized structures, such as Tip5 [36] and Anemoi permutations [37].

Continuing advances are closing the gap between theory and practice. Research conducted by Koroglu et al. [40] and Shi et al. [42] has been aimed at network efficiency and quantum-speed enhancements while the other ones have introduced password handling [43], key exchange [44], and tamper detection [45]. In addition, hash functions are being incorporated into more complex structures such as blockchain-based systems [46], big data clustering [47], and biometrics [48, 49]. The significance of these algorithms in safeguarding the expanding digital ecosystem is highlighted by the development of face recognition [50] and image tampering detection [51].

3. OVERVIEW OF HASH FUNCTION

The landscape of hash functions is broad, covering legacy cryptographic standards to high-performance non cryptographic algorithms and new chaos based. The functions have been listed according to category.

The goal of these functions is to ensure the collision resistance and preimage resistance.

Cryptographic Hash Functions.

The MD2, MD4, MD5 were early cryptographic hash functions that were used in many applications. However, they have been known to have significant collision vulnerabilities, and are no longer recommended for further use.

SHA Group

SHA 1 was widely used earlier but no longer considered secure against collision attacks [54].

In application to perform SSL/TLS and digital signature work the SHA suite as it is also known has become the de-facto industry standard for strong security.

SHA-3 is the most recent standard of NIST; it has various hash length with security against quantum computing [57].

Some Other Cryptographic Hashes.

The RIPEMD family RIPEMD-160 is used in Bitcoin.

Whirlpool is a 512-bit hash that is both secure and fast [58].

Blake is SHA 3 finalist which is ultra secure and fast.

Gost/Gost-Crypto: Standardized for use mainly in Russian government systems [61].

The cryptographic functions for flexibility by Tiger and HAVAL are less popular than SHA [29, 59].

Snefru was an early algorithm that became secure but was superseded [60].

Hashes that cater to a purpose and include a password.

SipHash is used for short message crypto hashing. It is also optimized for speed.

Password hashing, such as bcrypt and scrypt, are designed to be either computationally or memory-intensive. They are very strong defenses against attacks that try to guess the stored password [70, 71].

Non-cryptographic hash functions

Designed for hyper-fast performance and uniform distributions, they work well for hash tables, checksums or checking for data errors, but definitely not for security.

Adler32, CRC32/64, and CRC32C are industry standards for detecting errors during data transmission.

High-performance table hashes are Murmur3, FNV, Joaat, Jenkins, CityHash, FarmHash, SpookyHash, and the xxHash family (XXH32, XXH64, XXH3, XXH128), which combine speed with low collision rates and are primarily for databases, Bloom filters, and high-speed data structures [64, 65, 66, 67, 68].

Chaos Controlled Hash Functions

Emerging frontiers in cryptography, these functions exploit chaos theory specifically sensitivity to initial conditions to generate unpredictable and highly complex outcomes [4, 74, 75]. Though not as widely adapted as conventional algorithms, they are under active development to facilitate high-security data protection. Furthermore, their ability to withstand conventional cryptanalysis makes them a fertile area for future study.

4. APPLICATIONS AREAS OF HASH FUNCTIONS

The classification of hash functions based on their operational domains is essential for understanding their diverse roles in modern digital infrastructure. These categories include:

Cryptographic Security: This domain encompasses hash functions explicitly designed to maintain the confidentiality and integrity of cryptographic systems. These algorithms are fundamental to encryption protocols, the generation of digital signatures, and the facilitation of secure communications [76].

Data Management and Integrity: Functions in this category prioritize the optimization of data handling. They are widely utilized in database management, distributed file systems, and high-volume data processing to ensure efficient storage, rapid retrieval, and ongoing verification of data consistency [77].

Authentication and Verification: This category highlights functions critical for identity assurance and data validation. They are primarily applied to secure password storage, the issuance and management of digital certificates, and the generation of Message Authentication Codes (MACs) to confirm sender authenticity [78].

Blockchain and Distributed Systems: Specifically tailored for decentralized environments, these hash functions maintain the immutability and security of distributed ledgers. They are indispensable for core blockchain operations, including transaction validation, consensus mechanisms, and cryptocurrency mining [79].

Forensics and Data Analysis: This classification focuses on the application of hashing in forensic investigations and analytical workflows. By generating unique digital fingerprints, these functions support malware detection, file integrity verification, and the tracking of sensitive data during forensic inquiries [80].

5. EVALUATION AND CATEGORIZATION OF HASHING ALGORITHMS

This section will elaborate on the exhaustive framework used for classifying and evaluating various hashing algorithms, which is discussed in the tables and figures of the study.

Measurement of performance and analysis of scaling

The study examines the response of hashing algorithms to scaling data demands. Table I illustrates the execution time of the different algorithms when the size of the files is varied from 1MB to 100 MB to 1 GB. The finding points to a pattern common to all file sizes. Nevertheless, the research finds serious performance gaps.

Older algorithms such as MD2 show a rapid and non-linear increase in processing time with gigabyte-size files.

Many algorithms exhibit an overall small processing time increase when using up to 100MBs, but they show a sharp and marked performance degradation when using 1GB.

The horizontal bar chart of hash rates (million hashes a second) is seen through the Throughput graph (Figure 3). Arranging these from least to most resource-demanding gives one a quick idea of the best algorithm for high-load points such as cryptocurrency mining or massive data analytics.

Creating methodologies and taxonomies

Table II provides a classification system for hash functions based on their internal structure.

MD5 and SHA-1 fall under Merkle-Damgard construction. Devices that operate on streams in this manner are known as stream processors. They treat an input stream as a sequence of blocks of data.

The SHA-3 family employs sponge construction. Thus, this method takes in all the data into a fixed state then squeezes out only the hash, thus allowing us to accept inputs of variable length, and outputs of variable length.

Wide Pipe Design implicates designs for which the internal state is significantly larger than the final one. The main advantage of this wide pipe design is that it adds security against internal collision attacks.

Tools like *Adler32, CRC32, CityHash and Murmur3* are a different category. Such algorithms achieve speed improvements in non-security contexts, such as hash tables and error detection. However, they do not offer the collision resistance necessary for security.

Bcrypt and Scrypt are two functions designed with a unique purpose. Both can be configured to be very expensive, which makes brute-force and hardware-accelerated attacks inefficient.

Regulations, Compatibility, and Sensitivity

The examination appraises the "readiness" of the algorithms through two lenses.

Table III categorizes functions according to their industry status. It distinguishes between Widely Used (e.g. SHA-2) from Obsolete functions and provides a qualitative Low to High metric for cross-platform compatibility.

The Avalanche Effect (Figure 2 & Table IV) is the important property of S-Boxes in which a small change in input, i.e. one input bit change, causes drastic and unpredictable change in output not just one bit change but radical change. This property is being tested with specific inputs CURaj and CURAJ. This assesses whether the algorithm is sufficiently sensitive so that a cryptanalyst may not exploit patterns.

Domain-Specific Suitability Similarity Results (table V and figure 4).

The hashing algorithms are classified into five different domains.

The SHA variants and Whirlpool are used for encryption and digital signatures.

High-speed and low-security series such as MD and RIPEMD are widely used for data integrity checks and checksums. Important functions like SHA-224, Scrypt etc have been given priority that will help in authenticating the credentials.

TABLE I

HASHING TIME IN SECONDS FOR DIFFERENT ALGORITHMS AT FILE SIZES OF 1MB, 10MB, 100MB, AND 1GB.

Algorithm	Time for 1MB	Time for 10MB	Time for 100MB	Time for 1GB
md2	0.14	1.26	12.30	122.92
md4	0.0017	0.0191	0.16	1.59
md5	0.0057	0.0263	0.24	2.53
sha1	0.0028	0.0259	0.21	2.34
sha224	0.0061	0.0572	0.52	5.53
sha256	0.0061	0.0529	0.53	5.91
sha512/224	0.0037	0.0327	0.35	3.50

sha512	0.0034	0.0375	0.33	3.36
sha3-224	0.0032	0.0368	0.33	3.38
sha3-256	0.0033	0.0347	0.35	3.65
sha3-384	0.0043	0.0446	0.47	4.49
sha3-512	0.0059	0.0622	0.62	6.32
ripemd160	0.0050	0.0525	0.53	5.46
ripemd256	0.0038	0.0392	0.39	4.21
whirlpool	0.0103	0.1047	1.04	10.87
tiger160,3	0.0018	0.0189	0.18	1.90
snefru	0.0310	0.3339	3.37	33.93
snefru256	0.0311	0.3232	3.29	33.68
gost	0.0153	0.1564	1.62	16.55
gost-crypto	0.0152	0.1608	1.56	16.25
adler32	0.0008	0.0090	0.08	0.89
crc32	0.0003	0.0034	0.03	0.33
fnv132	0.0012	0.0159	0.13	1.41
joaat	0.0021	0.0249	0.22	2.33
murmur3a	0.0005	0.0065	0.06	0.62
murmur3f	0.0003	0.0044	0.04	0.46
xxh32	0.0003	0.0042	0.04	0.43
xxh64	0.0003	0.0035	0.03	0.34
xxh3	0.0003	0.0044	0.04	0.39
xxh128	0.0003	0.0039	0.03	0.38
haval128,3	0.0030	0.0373	0.32	3.22
haval160,3	0.0030	0.0370	0.32	3.21

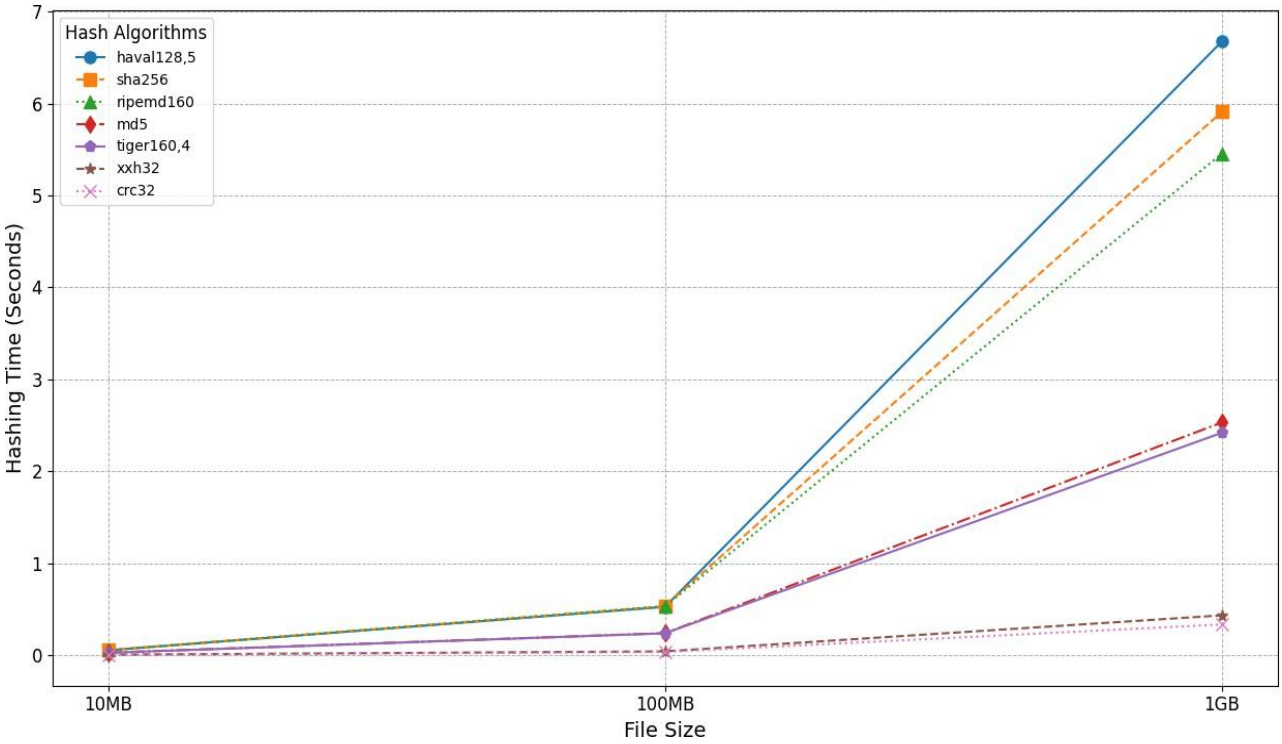

Fig. 1. Hashing Time for Different Algorithms at File Sizes of 10MB, 100MB, and 1GB.

The ideal candidates for mining and ledger immutability are SHA-256 and BLAKE due to their structural robustness.

SHA-3 offers better quantum attack resistance and guarantees long-term data integrity.

Statistical Randomness and NIST Validation

Theoretical integrity was verified by performing 10,000 samples using the NIST statistical test suites on these functions.

Table VI confirms that the bits generated are uniformly distributed without any bias either towards 0s or 1s.

Table VII provides the p-values associated with the frequency and block-frequency tests to confirm random behavior.

Reliability: Summary Table of Success Rates VIII The results show that SHA-256 and Whirlpool seem the most reliable ones. They pass the tests with the perfect proportion so they are the industry benchmarks for secure applications.

In the end, the research concludes that no algorithm fits all. As shown in Figure 4, Data Management and Cryptographic Security are the areas where most hashes are used. However, specialized domains like Blockchain and Forensics have a more focused use of particular hashes. This selective approach helps them meet specific security and performance requirements.

6. NIST RECOMMENDATIONS AND NEW USE CASES

The National Institute of Standards and Technology (NIST) serves as the primary authority in establishing cryptographic standards designed to protect data integrity and security. By emphasizing performance, resilience, and adaptability, NIST provides a structured framework for the deployment of hash functions across diverse digital environments.

NIST Standards and Algorithm Recommendations

Preferred Standards: For general cryptographic applications, NIST highly recommends the **SHA-2 family** (SHA-224, SHA-256, SHA-384, and SHA-512) due to their robust security profiles and seamless integration with modern systems [81].

Future-Proofing: For environments requiring enhanced resistance against emerging threats—specifically advances in quantum computing—NIST identifies the **SHA-3 family** as a superior alternative [82]. Its unique "sponge" architecture provides exceptional durability against theoretical cryptanalytic attacks [82].

Deprecation Warnings: NIST explicitly advises against the use of legacy algorithms such as **MD5 and SHA-1**. These functions are considered insecure for modern applications like digital signatures and authentication due to well-documented collision and pre-image vulnerabilities [83].

Best Practices for Implementation

Tailored Security: NIST emphasizes matching the output length of a hash function to the specific security requirements of the task. For instance, SHA-512 is recommended for high-security environments, while SHA-2 remains the standard for HMAC and key derivation functions (KDFs) [81, 84].

Password Storage: NIST distinguishes between generic hash functions and specialized password-hardening systems; it advises using memory-hard functions like **bcrypt or scrypt** for password storage rather than standard cryptographic hashes.

Regulatory Compliance: Adherence to established standards, such as **FIPS PUB 180-4** and **NIST Special Publication 800-57**, is critical for ensuring that systems meet industry best practices and legal security mandates [81, 84].

Practical Innovation: Hash-Based Web Crawling

Beyond core security, NIST principles extend to resource optimization, such as in web indexing. Implementing a **hash-based sitemap**—which stores the URL alongside a hash of the HTML content—significantly enhances efficiency for web crawlers:

Resource Optimization: Instead of retrieving entire webpages to check for updates, crawlers simply compare the current content hash with the stored version.

Efficiency Gains: This method drastically reduces bandwidth usage and server load, as only pages that have been modified are re-indexed.

Scalability: For large-scale sites (like Wikipedia), this approach replaces the need for resource-intensive, iterative queries with a single, streamlined request. By verifying content changes through cryptographic hashes, websites can maintain better content control while drastically improving the speed and intelligence of search engine indexing.

TABLE IV

HASH VALUES SENSITIVITY COMPARISON FOR INPUTS ”CURAJ” AND ”CURAJ”, AND SIZE IN BITS.

Hash Function	Hash Value Size (bits)	Hash for ”CURaj”	Hash for ”CURAJ”
md2	128	1a87aaf9f7abf093afa999a7d3215756	8c1a5280c24dbd350caa1b763cf8aa8f
md4	128	933b0eba42a59a36ab35a81bdd9cac56	872b09aaeebdcd51e0603932904de11c
md5	128	e824288128d77b113c96cd487d5e1e52	1caa07459600797e1f123f55637acc3f
sha1	160	72fb20f47e4d4df2f2961e9468d727c937d6c639	63ca79292c3d4154424a135558aba57c93b48808
sha256	256	068b7d496eff1717aa22346b90961ac3da5d8c33b859e9da71927d19dedc2953	0f76630016ee1099564d837f1e6d949979a67ad2184cc0795df7f94ac3110fe8
sha384	384	a6d20eca21eda60cfa01f00e29ffae348aa035d1b7a7cbfa734961c39f6c71e2d7c70aa8ace1fef8c4de97dad8e8b0de	59bcde11a94f338d506792cdd54bbbc4f8e0684b8e0c61c5be13dca4e656e442828b79da46fab66253b5adb35f1b0086
sha512/224	224	e76d74d8dd29e08ad5d621d1f4d633838	503f71d3388b72b795d2209ee1476ea7

		bcd4780e5926644953596e9	075d77d66153c5805 eaf380f
sha512/256	256	6fb92b3e1f54005969d7ed504a6a5426 66027a7e4809e0049b223da55f5c62cd	d1d0ba758b46478cad8e6c113bc6894c 4fd3eb993e15d790337e2eaac47db913
sha512	512	d737601875f7bd941f4075ed9d49d6632d8 fca11de45850a017abff30177ad1865dbbf 4b210bf3729673ecd4f22cde9b850d9fd51 cb52639dbb7671ba6e93dbe	aedc36ba662b1477d7aa3067d637483f92d 4f5cae594459b729aa10b5f3a6d2183a7e0 758aafb4f69041ac5b809e87e89c11ff89c 76b457c2299c4fa13918583
sha3-224	224	889c950592400ef349d85024bfe9489ab1 0d711895b66b5ebea9cc52	3c83478b9964f7de78f4ec5ab5ad1ec898 be1fe5e1e8f349ef4bd43
sha3-256	256	2b467b43626f5687201c7704e36b88c99 f219d4e15a57a1deda8bafc7067096a	4430290c737a74a018fa47f0361bfb71 824fba2faece7aba19835d47ba8d4169
sha3-384	384	89d63f07a86cdd96a8d8739638b467e1 188ec42602b710dc08eb6cae13f358c 9a8b294f300d0727668df16454786e6c2	845dbd7e0da06f286af177917c08fe9e cb99a3b392e53bc16c1d9d6da07452a5 864c37e4226091904b809d4b898689b1
sha3-512	512	834ef2fef178ac2ed33967109834b97c 16e608bbfd90559fc5d61004a414cf6 0f1e3784927dcfc12b84bced5adc2c39 f3ed8c46b88fb2d2bb489b9aef34eeba3	3a95ff26071bc33f1c17902ef64fd62 3ff162013286108d3e8632bba056e1c 5fcca1d42248345a34d4a87cc3ba59b 73f39eeca602cdbce75435fd3ce24a54f0b
ripemd128	128	3cbc4444bcd64898646a69f8 2c11ba0d	498f480b763714390c2ccc1e 2e8ffdf3
ripemd160	160	49bf12c6f402a526ac8dddb0067b8b6df4e3b ec0	d314c22851cfad4f5ae7f57e632de8e0ed6efc e7
ripemd256	256	8513a06bbec678df86cb4e6c0b58320 bd3f366c16e3aeae9c62aefcd0b1a5ef8	c6904a0805d43edbae829cd354968f8 6c163439bf80bad48f1cd867e7851fc42
ripemd320	320	913bd4ad53d32fb58c66250de4cfc429 c5dba0ed50d661e98c7263a67c10668 cc8d016d36e56c401	564b6ea0f70d11d550c0f95d8922334 ccc9df9a87103e57ae446f1bc3d71b8 9c95612db04554bf83
whirlpool	512	6117df878175f70c0805766be72bf6a fa4659e3575c1fb6d5a51114e483926 8f78367dc5b03748cf5b2686ac54cc7 1c86aaeb2e12031c320fba61732b240b0b4	17186bb917c7ffdc7552c9937381ad7 7eb0af408233532b4c033ecb896eeef 48da5d6916976641827b11a7e675f2 a96963d9ec30d1436756936f8161cec2b14d
tiger128,3	128	4ce20b0483cbbeec6c530edf551d0304	60ef2abd088c6c2f51bd2db12623b809
tiger160,3	160	4ce20b0483cbbeec6c530edf551d030418510 dd0	60ef2abd088c6c2f51bd2db12623b809fb918 68a
tiger160,4	160	c43ae734675c6c6f583930f502d674de6aaef 275	45a8018698dbd6fa219b027719079e6ae9f89 05a
tiger192,4	192	c43ae734675c6c6f583930f502d674de 6aaef2750990a54e	45a8018698dbd6fa219b027719079e6a e9f8905a9a7fc983
snefru	256	757cedccec30f7cf4964a55ca5027f5 c7e18a401c4837c 3ee5fe9e8689a90344	fd858aa5a6e189efc956f6d23c17e497f 03e156a8b10d4b97fe0b5417d0a9b4b
adler32	32	04ca01b6	046a0176
crc32	32	a7bf6ebb	613fee03
crc32c	32	bf07708b	e0311d57
fnv132	32	7b4180dc	5b414e1c
fnv1a64	64	aef57130d1cf2b62	af625130d22bc6e2
joaat	32	5ec5542b	d4946253
murmur3a	32	1e4d677f	dfe3d04e
murmur3f	128	cb80db78abcce80192d56f08 af1bcb8d	8de502b0d44ff8a3b254f5cd5 b4b1e7c
xxh128	128	54b92a69cdb8746a1d4aec3a 82e10675	d8836a7fc9b1aaff94d91b48 d6e30720
haval128,3	128	77e87d8a90ff5863814edacd4 1b219a4	ab42e886d7c0909d558f2a03 daeb9d14
haval160,3	160	7db36c2704f972388fb64ea63b333da87bd2 0776	1ffec659e65e91a63f6f302aa19cc999e9e6ae 87
haval192,3	192	da683dd25ab130471a8ce8fc7fe5e3d1b b3e1ab60b7a78a6	90fa535008acf10a073c512acca4df9c 14861a283866183b
haval160,4	160	683774205ae9cf80436a7116 701b9b9de3d05aed	69afd4d4ce0a1f535d9365e5 6c346940b6aa8a2a
haval192,4	192	f428cea675ed696ffa9407826 02cec8c0196df136b6ba818	e143d50baa55f18e0070c8e6 e6a658dfd777cae3286b4ecc
haval224,4	224	f1f6dc8ebcfcf3cb8d3d2a958b1ac76122d9a3 2bc d957ebc54da969f	a202d380acce8ccc847b747f6b7e4adb6af5d 992 43ece266b555ef8f

haval256,4	256	e30a845a64826bf66e4cb948972a5b02694c4d d666240a8ce6bb312fb29dba73	32315e9dfc0067052d6f8388b180e3541aa2d cf98acee9c54f379ad7c9017434

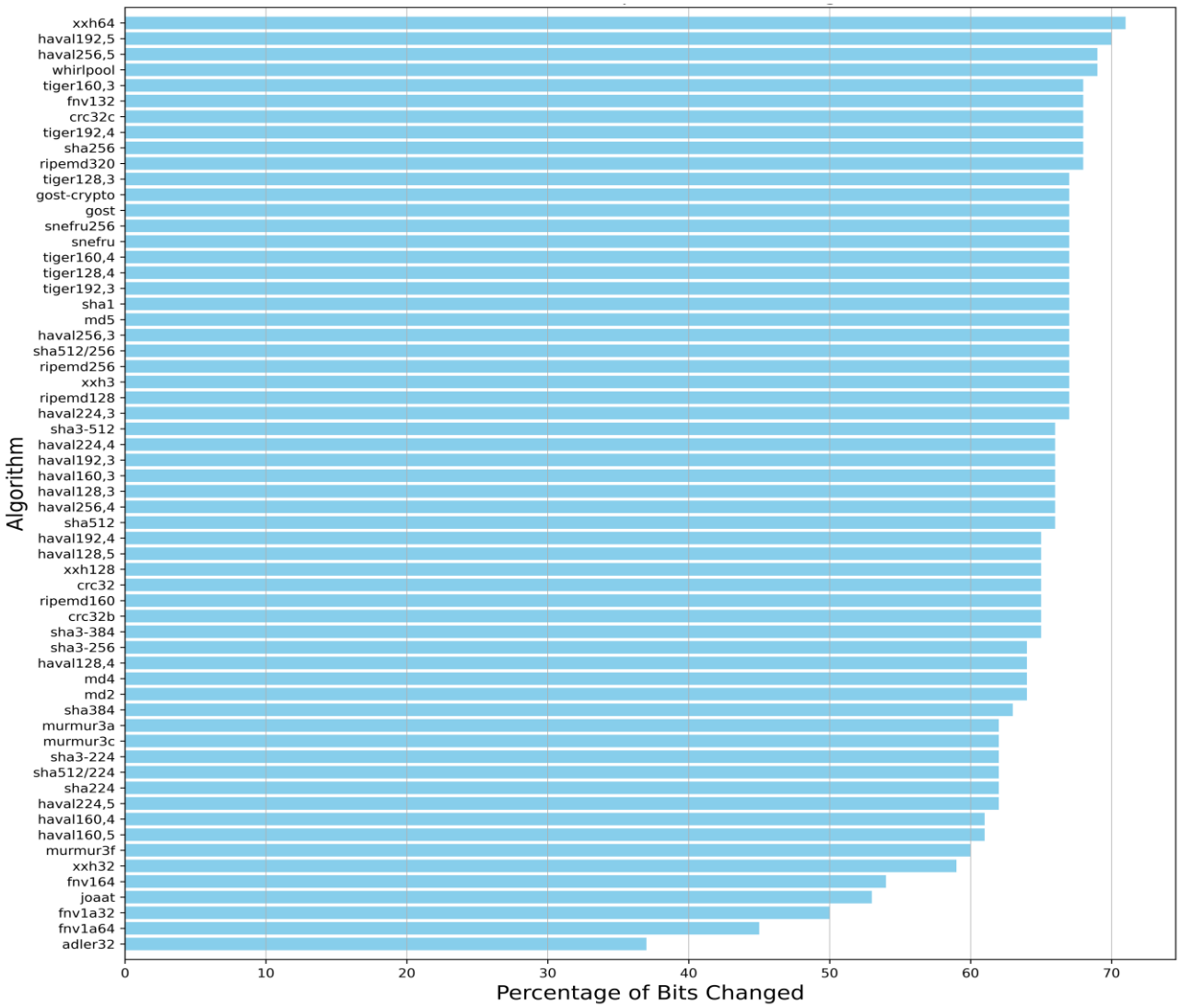

Fig. 2. Comparative Analysis of Avalanche Effect in Hash Functions

TABLE V

SUITABILITY AND PERFORMANCE CHARACTERISTICS OF HASH ALGORITHMS ACROSS DIFFERENT APPLICATION DOMAINS

Application Domain	**Hash Algorithms**	**Suitability for High-Security Applications**	**Speed**	**Resistance to Quantum Attacks**
Cryptographic Security	SHA224, SHA256, SHA384, SHA512/224, SHA512/256, SHA512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, Whirlpool, Tiger series, Gost, Gost-Crypto, Haval series, BLAKE variants, Chaos based hash functions	Very Suitable	Moderate to High	Widely used in encryp-tion, digital signatures, se-cure data transmission

Data Management and Integrity	MD2, MD4, MD5, SHA1, RIPEMD series, Adler32, CRC32, CRC32B, CRC32C, FNV series, CityHash, FarmHash, XXH32, XXH64, XXH3, XXH128	Less Suitable	High	Data storage, checksums, error-checking in networks and storage devices
Authentication and Verification	SHA224, SHA256, SHA384, SHA512/224, SHA512/256, SHA512, SHA3 series, Bcrypt, Scrypt, SipHash, RIPEMD160 (notably used in Bitcoin)	Suitable	Moderate to High	User authentication, data authenticity verification
Blockchain and Distributed Systems	SHA256 (widely used in blockchain technologies), RIPEMD160 (as in Bitcoin), BLAKE variants, Chaos based hash functions (potential application due to their unique properties)	Suitable	Moderate to High	Cryptocurrency mining, transaction validation, distributed ledgers
Forensics and Data Analysis	SHA3 series (due to their resistance to quantum attacks), Whirlpool, Haval series, Gost, Gost-Crypto	Suitable	Moderate to High	Digital forensics, data integrity verification, mal-ware detection

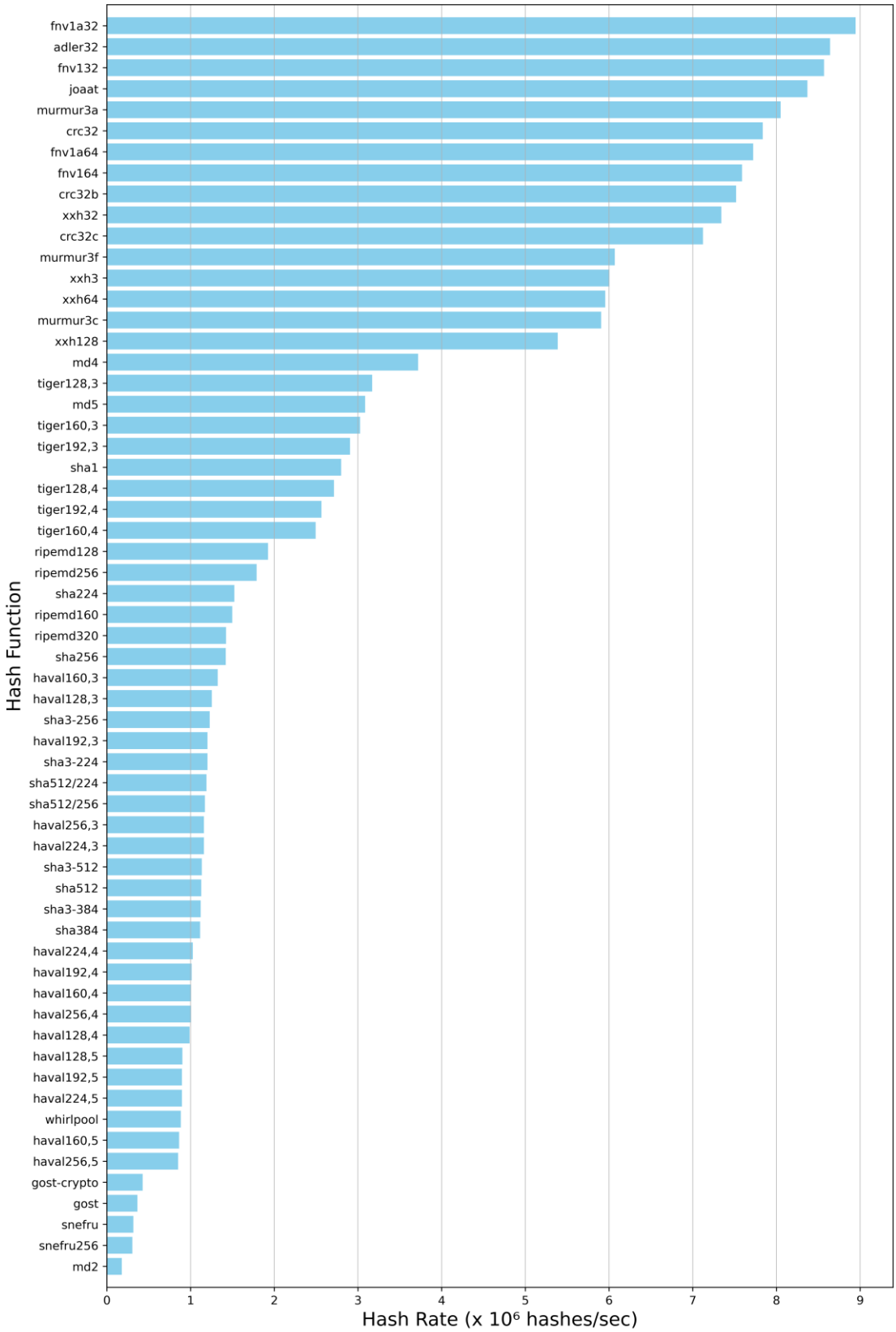

Fig. 3. Performance Comparison of Hash Functions by Hash Rate

TABLE VI

BIT POSITION VALUES FOR VARIOUS ALGORITHMS.

Bit_pos. / Algo.	crc32	haval256,5	md5	ripemd160	sha1	sha256	tiger192,3	whirlpool
1	5004:4996	5053:4947	4912:5088	4925:5075	5003:4997	4936:5064	4978:5022	5013:4987
2	4990:5010	5046:4954	4913:5087	4925:5075	4956:5044	5060:4940	4976:5024	5046:4954
3	5022:4978	4981:5019	4953:5047	5070:4930	5012:4988	4926:5074	4954:5046	5078:4922
4	4964:5036	4977:5023	5024:4976	4956:5044	5034:4966	4938:5062	5034:4966	4942:5058
5	4977:5023	4996:5004	4958:5042	4930:5070	4936:5064	4982:5018	4953:5047	4944:5056
6	5025:4975	4984:5016	4932:5068	4914:5086	4996:5004	4974:5026	5002:4998	4952:5048
7	5011:4989	5001:4999	5005:4995	4951:5049	4987:5013	5067:4933	5011:4989	5001:4999
8	4945:5055	5063:4937	5003:4997	5012:4988	4934:5066	4936:5064	5043:4957	4938:5062
9	5039:4961	5034:4966	4951:5049	4947:5053	5061:4939	5050:4950	5028:4972	4998:5002
10	4985:5015	4949:5051	5023:4977	5032:4968	5022:4978	5025:4975	5015:4985	5044:4956

TABLE VII

P-VALUES FOR STATISTICAL TESTS ACROSS DIFFERENT ALGORITHMS.

Test/p-value	crc32	haval256,5	md5	ripemd160	sha1	sha256	tiger192,3	whirlpool
Frequency	0.350485	0.911413	0.739918	0.122325	0.534146	0.122325	0.122325	0.911413
BlockFrequency	0.000199	0.066882	0.911413	0.066882	0.122325	0.534146	0.911413	0.534146
CumulativeSums	0.350485	0.534146	0.911413	0.534146	0.991468	0.350485	0.350485	0.350485
CumulativeSums	0.350485	0.534146	0.911413	0.534146	0.991468	0.350485	0.350485	0.350485
Runs	0.534146	0.213309	0.213309	0.534146	0.739918	0.350485	0.213309	0.534146
LongestRun	0.534146	0.739918	0.350485	0.534146	0.350485	0.534146	0.911413	0.122325
Rank	0.000000	0.017912	0.017912	0.739918	0.911413	0.000199	0.350485	0.739918
FFT	0.000000	0.991468	0.122325	0.017912	0.008879	0.534146	0.911413	0.911413
OverlappingTemplate	0.213309	0.911413	0.911413	0.534146	0.534146	0.911413	0.350485	0.739918

TABLE VIII

PROPORTION OF SUCCESSFUL OUTCOMES FOR STATISTICAL TESTS ACROSS DIFFERENT ALGORITHMS.

Test/algorithm proportion	crc32	haval256,5	md5	ripemd160	sha1	sha256	tiger192,3	whirlpool
Frequency	10/10	9/10	10/10	9/10	10/10	10/10	10/10	10/10
BlockFrequency	6/10*	10/10	10/10	10/10	10/10	10/10	10/10	10/10
CumulativeSums	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10
CumulativeSums	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10
Runs	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10
LongestRun	9/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10
Rank	0/10*	10/10	10/10	10/10	10/10	10/10	10/10	10/10
FFT	0/10*	10/10	10/10	10/10	10/10	10/10	10/10	10/10
OverlappingTemplate	9/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10

Fig. 4. Distribution of Hash Algorithms Across Application Domains

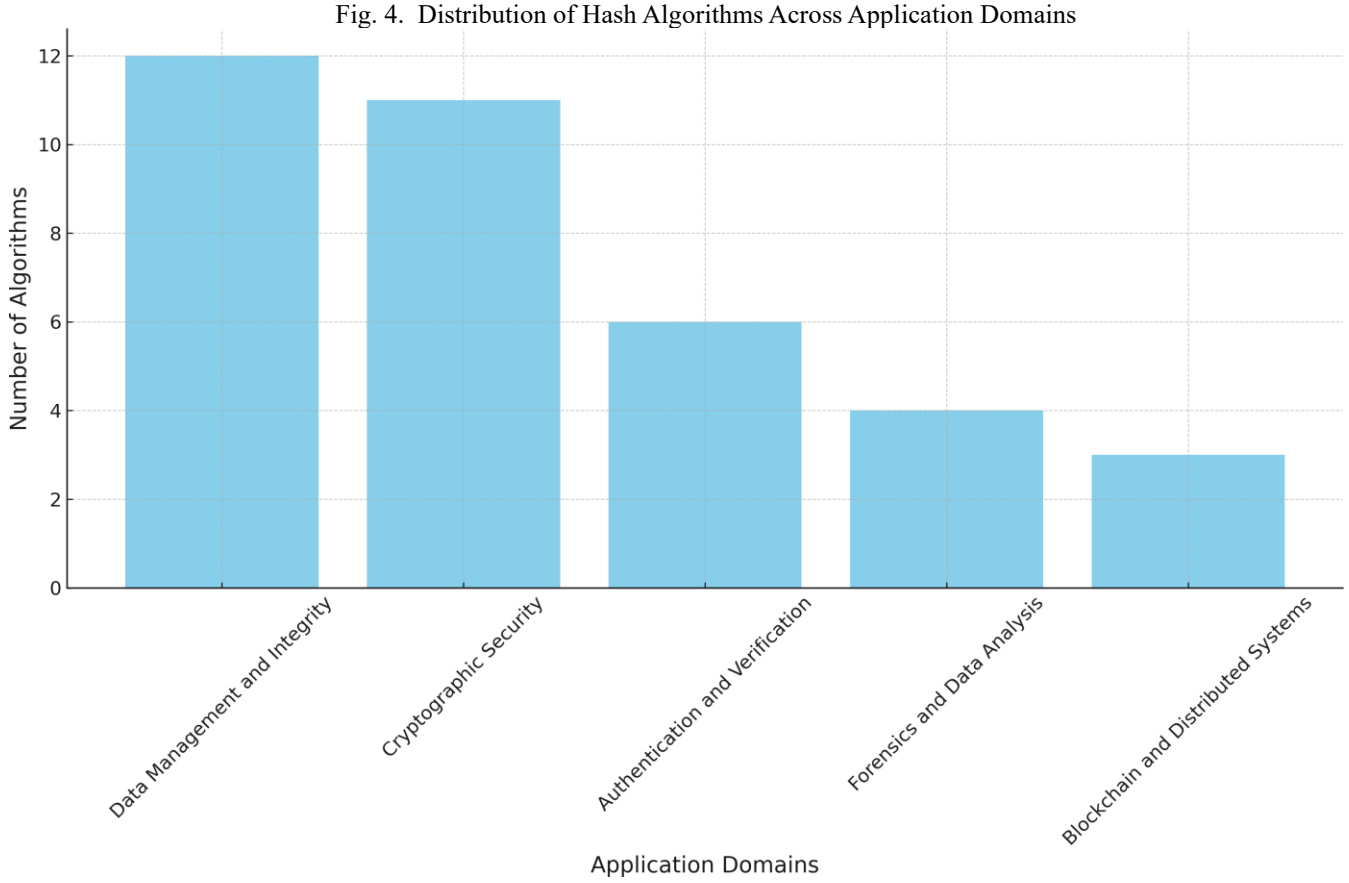

Fig. 4. Distribution of Hash Algorithms Across Application Domains

7. CONCLUSION AND FUTURE WORK

This study provides a comprehensive evaluation and taxonomy of cryptographic hash algorithms, highlighting their fundamental role in maintaining digital security. By conducting a comparative analysis of both traditional and emerging chaos-based hash functions, we have identified key criteria for algorithm selection, specifically regarding performance, security, and domain-specific applicability. Our findings emphasize the necessity of balancing computational efficiency with robust security in the face of rapidly evolving digital threats. This research serves as a practical guide for selecting optimal algorithms to meet diverse security requirements and contributes to the foundational development of more resilient cryptographic frameworks.

Looking ahead, our future research initiatives will focus on the following pillars of cryptographic innovation:

Chaos-Based and Quantum-Resistant Algorithms: We will continue exploring the integration of chaos theory into cryptographic hashing. By developing hybrid models that combine the unpredictability of chaos theory with quantum-resistant designs,

we aim to establish robust security protocols capable of withstanding the threats posed by next-generation quantum computing.

Adaptable Hash Systems: Future hashing architectures must be dynamic. We intend to focus on self-adjusting systems that can monitor emerging threats in real-time and adapt their parameters to maintain high security without manual intervention.

Efficiency in Real-World Applications: Bridging the gap between theoretical strength and practical utility is vital. Our research will prioritize the design of hash functions that maintain high performance across diverse hardware, ensuring they are scalable and easily deployable in real-world systems ranging from IoT devices to high-speed data centers.

Next-Generation Security Foundations: We remain committed to fostering the development of flexible, rapid, and highly secure hashing solutions, as these are essential to safeguarding the future of global digital security systems.

References

1. H. Ahmed, “A review of hash function types and their applications,” Wasit Journal of Computer and Mathematics Science, vol. 1, no. 3,
2. pp. 120–139, 2022.
3. H. Liu, X. Wang, and A. Kadir, “Constructing chaos-based hash function via parallel impulse perturbation,” Soft Computing, vol. 25, no. 16, pp. 11077–11086, 2021.
4. L. E. Hughes, “Basic cryptography: Hash function,” in Pro Active Directory Certificate Services: Creating and Managing Digital Cer-tificates for Use in Microsoft Networks, pp. 19–22, Springer, 2022.
5. M. Rasool and S. B. Belhaouari, “From collatz conjecture to chaos and hash function,” Chaos, Solitons & Fractals, vol. 176, p. 114103, 2023.
6. W. Diffie and M. E. Hellman, “New directions in cryptography,” in IEEE Transactions on Information Theory, pp. 644–654, 1976.
7. R. C. Merkle, “Secure communications over insecure channels,” Com-munications of the ACM, vol. 21, no. 4, pp. 294–299, 1978.
8. M. O. Rabin, “Digitalized signatures and public-key functions as intractable as factorization,” 1979.
9. S. M. Matyas, “Generating strong one-way functions with cryp-tographic algorithm,” IBM Technical Disclosure Bulletin, vol. 27,
10. pp. 5658–5659, 1985.
11. C. H. Meyer and M. Schilling, “Secure program load with manipula-tion detection code,” in Proc. Securicom, vol. 88, pp. 111–130, 1988.
12. B. Kaliski, “Rfc1319: The md2 message-digest algorithm,” 1992.
13. J. Linn, “Rfc1115: Privacy enhancement for internet electronic mail: Part iii-algorithms, modes, and identifiers,” 1989.
14. R. L. Rivest, “The md4 message-digest algorithm. internet request for comments,” tech. rep., April 1992. RFC 1320, 1990.
15. R. L. Rivest, “The md4 message digest algorithm,” Lecture notes in Computer Science, Advances in Cryptography CRYPTO ‘90, 1998.
16. R. Rivest, “The md5 message-digest algorithm, internet request for comments: Rfc 1321,” Internet Engineering Task Force, 1992.
17. N. I. of Standards and T. U. T. Administration, Secure hash standard, vol. 180. US Department of Commerce, Technology Administration, National Institute of . . . , 1993.
18. A. J. Menezes and S. A. Vanstone, Advances in Cryptology-CRYPTO’90: Proceedings, vol. 537. Springer, 2003.
19. R. Rivest, “Rfc1321: The md5 message-digest algorithm,” 1992.
20. C. De Canniere, F. Mendel, and C. Rechberger, “Collisions for 70-step sha-1: on the full cost of collision search,” in Selected Areas in Cryptography: 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers 14, pp. 56–73,
21. Springer, 2007.
22. B. Den Boer and A. Bosselaers, “An attack on the last two rounds of md4,” in Annual International Cryptology Conference, pp. 194–203, Springer, 1991.
23. B. Den Boer and A. Bosselaers, “Collisions for the compression function of md5,” in Workshop on the Theory and Application of of Cryptographic Techniques, pp. 293–304, Springer, 1993.
24. H. Dobbertin, A. Bosselaers, and B. Preneel, “Ripemd-160: A strength-ened version of ripemd,” in International Workshop on Fast Software Encryption, pp. 71–82, Springer, 1996.
25. F. Chabaud and A. Joux, “Differential collisions in sha-0,” in Annual International Cryptology Conference, pp. 56–71, Springer, 1998.
26. E. Biham and R. Chen, “Near-collisions of sha-0,” in Advances in Cryptology–CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004. Proceedings 24, pp. 290–305, Springer, 2004.
27. X. Wang, Y. L. Yin, and H. Yu, “Finding collisions in the full sha-1,” in Advances in Cryptology–CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005. Proceedings 25, pp. 17–36, Springer, 2005.
28. X. Wang and H. Yu, “How to break md5 and other hash functions,” in Annual international conference on the theory and applications of cryptographic techniques, pp. 19–35, Springer, 2005.
29. R. K. Dahal, J. Bhatta, and T. N. Dhamala, “Performance analysis of sha-2 and sha-3 finalists,” International Journal on Cryptography and Information Security (IJCIS), vol. 3, no. 3, pp. 720–730, 2013.
30. R. F. Kayser, “Announcing request for candidate algorithm nomina-tions for a new cryptographic hash algorithm (sha-3) family,” Federal Register, vol. 72, no. 212, p. 62, 2007.
31. A. Bosselaers and B. Preneel, Integrity Primitives for Secure In-formation Systems: Final Ripe Report of Race Integrity Primitives Evaluation. No. 1007, Springer Science & Business Media, 1995.

32. Y. Zheng, J. Pieprzyk, and J. Seberry, “Haval—a one-way hashing algorithm with variable length of output,” in Advances in Cryptol-ogy—AUSCRYPT’92: Workshop on the Theory and Application of Cryptographic Techniques Gold Coast, Queensland, Australia, Decem-ber 13–16, 1992 Proceedings 3, pp. 81–104, Springer, 1993.
33. S. Miyaguchi, M. Iwata, and K. Ohta, “New 128-bit hash function,” in Proc. 4th International Joint Workshop on Computer Communications, Tokyo, Japan, pp. 279–288, 1989.
34. L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofneg-ger, “Poseidon: A new hash function for {Zero-Knowledge} proof systems,” in 30th USENIX Security Symposium (USENIX Security 21),
35. pp. 519–535, 2021.
36. Q. Yuan, M. Tibouchi, and M. Abe, “On subset-resilient hash function families,” Designs, Codes and Cryptography, vol. 90, no. 3, pp. 719–758, 2022.
37. Y. P. Khanal, A. Alsadoon, K. Shahzad, A. B. Al-Khalil, P. W. Prasad,
38. S. U. Rehman, and R. Islam, “Utilizing blockchain for iot privacy through enhanced ecies with secure hash function,” Future Internet, vol. 14, no. 3, p. 77, 2022.
39. L. Grassi, D. Khovratovich, and M. Schofnegger, “Poseidon2: A faster version of the poseidon hash function,” Cryptology ePrint Archive, 2023.
40. V. Cheval, C. Cremers, A. Dax, L. Hirschi, C. Jacomme, and
41. S. Kremer, “Hash gone bad: Automated discovery of protocol attacks that exploit hash function weaknesses,” in 32nd USENIX Security Symposium, 2023.
42. A. Szepieniec, A. Lemmens, J. F. Sauer, B. Threadbare, et al., “The tip5 hash function for recursive starks,” Cryptology ePrint Archive, 2023.
43. C. Bouvier, P. Briaud, P. Chaidos, L. Perrin, R. Salen, V. Velichkov, and D. Willems, “New design techniques for efficient arithmetization-oriented hash functions: anemoi permutations and jive compression mode,” in Annual International Cryptology Conference, pp. 507–539, Springer, 2023.
44. P. Hou, T. Shang, Y. Zhang, Y. Tang, and J. Liu, “Quantum hash function based on controlled alternate lively quantum walks,” Scientific Reports, vol. 13, no. 1, p. 5887, 2023.
45. M. Al-Zubaidie, “Implication of lightweight and robust hash function to support key exchange in health sensor networks,” Symmetry, vol. 15, no. 1, p. 152, 2023.
46. T. Koroglu and R. Samet, “Can there be a two way hash function?,” IEEE Access, vol. 12, pp. 18358–18386, 2024.
47. M. Hassan, J. Vliegen, S. Picek, and N. Mentens, “A systematic exploration of evolutionary computation for the design of hardware-oriented non-cryptographic hash functions,” in Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1255–1263, 2024.
48. W.-M. Shi, P. Tian, S. Wang, Y.-G. Yang, and Y.-H. Zhou, “Quantum hash function based on continuous quantum walks,” Modern Physics Letters A, vol. 39, no. 07, p. 2350189, 2024.
49. C. Somboonpattanakit and N. Wisitpongphan, “Secure password stor-ing using prime decomposition,” IAENG International Journal of Computer Science, vol. 48, no. 1, pp. 152–160, 2021.
50. M. A. Cardona-Lo´pez, J. C. Chimal-Egu´ıa, V. M. Silva-Garc´ıa, and
51. R. Flores-Carapia, “Key exchange with diffie-hellman protocol and composite hash-functions,” in 2024 12th International Symposium on Digital Forensics and Security (ISDFS), pp. 1–6, IEEE, 2024.
52. A. Srivatsa, T. Ananthapadmanabha, L. K. MV, and A. Suma, “En-hancing smart grid security with sha-sarimax: Identifying and restoring corrupted files from fdia.,” IAENG International Journal of Computer Science, vol. 51, no. 8, pp. 1112–1121, 2024.
53. C. Pujari, C. CB, S. R. Muppidi, and M. C. Belavagi, “A novel method of secure child adoption using blockchain technology.,” IAENG Inter-national Journal of Applied Mathematics, vol. 53, no. 4, pp. 1531–1539, 2023.
54. Y. Lv, “Cloud computation-based clustering method for nonlinear complex attribute big data,” IAENG International Journal of Computer Science, vol. 49, no. 3, pp. 736–744, 2022.
55. S. Boonkrong, “Security analysis and improvement of a multi-factor biometric-based remote authentication scheme.,” IAENG International Journal of Computer Science, vol. 46, no. 4, pp. 713–724, 2019.
56. X. Y. H. Xuan and J. Tao, “A forward-secure fuzzy identity-based fully homomorphic signature over lattices,” IAENG International Journal of Computer Science, vol. 48, no. 3, pp. 605–612, 2021.
57. S. Wang, “A face recognition method based on lightweight neural network and multi hash recognition degree weighting.,” IAENG Inter-national Journal of Applied Mathematics, vol. 54, no. 3, pp. 581–586, 2024.
58. K. Tan, L. Li, and Q. Huang, “Image manipulation detection using the attention mechanism and faster r-cnn [j],” IAENG International Journal of Computer Science, vol. 50, no. 4, pp. 1261–1268, 2023.
59. A. M. Ali and A. K. Farhan, “A novel improvement with an effective expansion to enhance the md5 hash function for verification of a secure e-document,” IEEE Access, vol. 8, pp. 80290–80304, 2020.
60. A. Zellagui, N. Hadj-Said, and A. Ali-Pacha, “Secure md4 hash function using henon,” Malaysian Journal of Computing and Applied Mathematics, vol. 3, no. 2, pp. 73–80, 2020.
61. S. Debnath, A. Chattopadhyay, and S. Dutta, “Brief review on journey of secured hash algorithms,” in 2017 4th International Conference on Opto-Electronics and Applied Optics (Optronix), pp. 1–5, IEEE, 2017.
62. D. M. A. Cortez, A. M. Sison, and R. P. Medina, “Cryptographic randomness test of the modified hashing function of sha256 to address length extension attack,” in Proceedings of the 2020 8th International Conference on Communications and Broadband Networking, pp. 24–28, 2020.

63. A. Maetouq, S. M. Daud, N. A. Ahmad, N. Maarop, N. N. A. Sjarif, and H. Abas, "Comparison of hash function algorithms against attacks: A review," International Journal of Advanced Computer Science and Applications, vol. 9, no. 8, 2018.
64. N. Mouha, M. S. Raunak, D. R. Kuhn, and R. Kacker, "Finding bugs in cryptographic hash function implementations," IEEE transactions on reliability, vol. 67, no. 3, pp. 870–884, 2018.
65. E. Swathi, G. Vivek, and G. S. Rani, "Role of hash function in cryptography," Int. J. Adv. Eng. Res. Sci.(IJAERS), 2016.
66. S. Park, C.-G. Jung, A. Park, J. Choi, and H. Kang, "Tiger: tiny bandwidth key encapsulation mechanism for easy migration based on rlwe (r)," Cryptology ePrint Archive, 2022.
67. A. Sadeghi-Nasab and V. Rafe, "A comprehensive review of the security flaws of hashing algorithms," Journal of Computer Virology and Hacking Techniques, vol. 19, no. 2, pp. 287–302, 2023.
68. A. Konkin and S. Zapechnikov, "Zero knowledge proof and zk-snark for private blockchains," Journal of Computer Virology and Hacking Techniques, vol. 19, no. 3, pp. 443–449, 2023.
69. W. Xia, C. Wei, Z. Li, X. Wang, and X. Zou, "Netsync: A network adaptive and deduplication-inspired delta synchronization approach for cloud storage services," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 10, pp. 2554–2570, 2022.
70. P. Koopman, K. Driscoll, and B. Hall, "Selection of cyclic redundancy code and checksum algorithms to ensure critical data integrity," 2015.
71. C. Hayes, Non-Cryptographic Hash Functions: Focus on FNV. PhD thesis, National University of Ireland Maynooth, 2023.
72. V. Akoto-Adjepong, S. Okyere-Gyamfi, and M. Asante, "An enhanced non-cryptographic hash function," International Journal of Computer Applications, vol. 176, no. 15, 2020.
73. M. Cheng, Y. Wu, X. Zhou, J. Li, and L. Zhang, "Efficient web archive searching," 2020.
74. T. Ferdousi, D. Gruenbacher, and C. M. Scoglio, "A permissioned distributed ledger for the us beef cattle supply chain," IEEE Access, vol. 8, pp. 154833–154847, 2020.
75. F. Yamaguchi and H. Nishi, "Hardware-based hash functions for network applications," in 2013 19th IEEE International Conference on Networks (ICON), pp. 1–6, IEEE, 2013.
76. J.-P. Aumasson and D. J. Bernstein, "Siphash: a fast short-input prf," in International Conference on Cryptology in India, pp. 489–508, Springer, 2012.
77. N. Provos and D. Mazieres, "Bcrypt algorithm," in USENIX, 1999.
78. C. Percival and S. Josefsson, "The scrypt password-based key deriva-tion function," tech. rep., 2016.
79. N. H. M. Ali and R. A. Abdul-Sattar, "Data integrity enhancement for the encryption of color images based on crc64 technique using multiple look-up tables," Iraqi Journal of Science, pp. 1729–1739, 2017.
80. N. Mouha, "Exploring formal methods for cryptographic hash function implementations," in Australasian Conference on Information Security and Privacy, pp. 177–195, Springer, 2023.
81. P. Ayubi, S. Setayeshi, and A. M. Rahmani, "Chaotic complex hashing: A simple chaotic keyed hash function based on complex quadratic map," Chaos, Solitons & Fractals, vol. 173, p. 113647, 2023.
82. J. Liu, Y. Liu, and B. Li, "Design and analysis of hash function based on spark and chaos system," International Journal of Network Security, vol. 25, no. 3, pp. 456–467, 2023.
83. B. Preneel, "Cryptographic hash functions," European Transactions on Telecommunications, vol. 5, no. 4, pp. 431–448, 1994.
84. M. R. Anwar, D. Apriani, and I. R. Adianita, "Hash algorithm in ver-ification of certificate data integrity and security," Aptisi Transactions on Technopreneurship (ATT), vol. 3, no. 2, pp. 181–188, 2021.
85. G. Hatzivasilis, "Password-hashing status," Cryptography, vol. 1, no. p. 10, 2017.
86. X. Zhao, Z. Lei, G. Zhang, Y. Zhang, and C. Xing, "Blockchain and distributed system," in Web Information Systems and Applica-tions: 17th International Conference, WISA 2020, Guangzhou, China, September 23–25, 2020, Proceedings 17, pp. 629–641, Springer, 2020.
87. Z. E. Rasjid, B. Soewito, G. Witjaksono, and E. Abdurachman, "A review of collisions in cryptographic hash function used in digital forensic tools," Procedia computer science, vol. 116, pp. 381–392, 2017.
88. National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)," Tech. Rep. FIPS PUB 180-4, National Institute of Standards and Technology (NIST), August 2015.
89. National Institute of Standards and Technology (NIST), "SHA-3 Finalist Evaluation Report," tech. rep., National Institute of Standards and Technology (NIST), 2012.
90. National Institute of Standards and Technology (NIST), "Transitions: Recommendations for Cryptographic Algorithms and Key Lengths (Rev. 2)," Tech. Rep. NIST Special Publication 800-131A Revision 2, National Institute of Standards and Technology (NIST), March 2019.
91. National Institute of Standards and Technology (NIST), "Recommen-dation for Key Management, Part 1: General (Rev. 5)," Tech. Rep. NIST Special Publication 800-57 Part 1 Revision 5, National Institute of Standards and Technology (NIST), May 2020.