Evaluation and Categorization of Hashing Algorithms Based on Their Applications

Suparn Padma Patra, and Mamta Rani

***Abstract*—Cryptographic hash algorithms are crucial in the fast expanding field of digital security in maintaining data integrity and authenticity. With a focus towards their uses in many fields, this study evaluates and classifies cryptographic hash functions. Based on their strengths and weaknesses in ensuring data integrity, security, and efficiency, a thorough comparison study evaluates at the performance, security, and usability of several hash techniques. Key domains where cryptographic hash is applied—including cryptographic security, data management, authentication, blockchain technology, and forensic analysis—are found by the study. This work attempts to fill the literature gap on the thorough assessment of hash systems by providing a methodical classification based on construction type, standardising, compatibility, and sensitivity to input modifications. Our results support the discipline of digital cryptography by offering understanding of the choice of suitable hash algorithms for particular uses, considering the differences between security, efficiency, and adaptability to quantum attacks.**

***Index Terms*—Cryptographic Hashing; Chaos-Based Hash Functions; Data Security; Algorithm Evaluation; Digital Cryptography.**

I. Introduction

Cryptographic hashing is a foundation technology in digital security to ensure data integrity and safety. The development and application of hash functions have been significant in many domains, from secure communication to data storage and blockchain technology [1]. Chaos-based hash functions have appeared as a novel and fascinating area of study. By utilisation of chaos theory principles, these functions present unpredictability and sensitivity to initial conditions that traditional cryptographic methods usually lack [2]. This unique approach in cryptographic hashing has opened a new way for enhancing security protocols, especially in environments where unpredictability and complex behaviour are crucial. Classical hash functions like SHA-256 and MD5 have set foundational standards in Cryptographic hashing [3]. However, advanced computational techniques and emerging security threats continually challenge the security and operational efficiency of these traditional hash functions for their use in various domains. Integrating chaos theory into hashing algorithms upgrades the complexity of hash functions, making them more robust [4]. This paper comprehensively reviews existing cryptographic hashing algorithms, focusing on categorising Hashing Algorithms based on their applications. Previous research has laid substantial groundwork for understanding the strengths and weaknesses of conventional hash functions. However, there still needs to be a significant gap in comprehensively evaluating and categorising hash functions and considering them where chaos theory has applied, especially with their varied applications. We have discussed hash functions' performance, security, and applicability in various domains. Hence, it fills the gaps in the current literature by offering a comprehensive categorisation and evaluation of hashing methods.

The paper is structured as follows: We begin by reviewing previous studies in Section II. Section III presents an overview of some popular hash functions and broader application categories for categorising hash functions. Section IV consists of the detailed evaluation and categorisation of various hashing algorithms and chaos-based methods and assessing their performance and applicability across different domains. Section V provides the evaluation and categorization of hashing algorithms based on various parameters. Various recommendations regarding use of Hash Functions by NIST are discussed in Section VI along with new use case. The paper is summarised in Section VII, focusing on our key contributions and offering insights for future research, highlighting potential routes for further advancements in chaos-based cryptographic hashing.

II. Literature Review

In 1976, Diffie and Hellman [5], and Merkle [6] independently discovered the concept of public key cryptography that laid the foundational framework for digital signatures. Based on the suggestions by Rabin [7], the framework was improved by integrating a hash function before signing, leading to performance and security improvements. Subsequently, Matyas et. al. [8], contributed to the field by detailing block cipher-based hash function constructions that remain significant. Emerged from IBM, the MDC-2 construction, utilized an n-bit block cipher to produce a 2n-bit hash function in 1987 [9]. Rivest introduced the MD2 in 1988 as the first recognized dedicated cryptographic hash function, followed by MD4 in 1990 and MD5 in 1991. The evolution of hash functions with MD4 and MD5 formed the basis for the Secure Hash Standard (SHA) developed by the U.S. National Institute of Standards and Technology (NIST) in 1993. The transition from SHA to SHA-1 in 1995, with the prior version becoming SHA-0, represented iterative progress in hash function design [10] [11] [12] [13] [14] [15] [16] [17]. However, the cryptographic community soon faced challenges as vulnerabilities in these hash functions emerged. Published weaknesses in MD4 in 1991, partial cryptanalysis of MD5 in 1993, and the discovery of the first collision in MD4 by Dobbertin in 1996 highlighted the potential

Manuscript received July 9, 2024; revised January 9, 2025.

Suparn Padma Patra is a Research Scholar at the Department of Computer Science, Central University of Rajasthan, Dist-Ajmer, 305817, Rajasthan, India (corresponding author to provide email: suparnpatra@gmail.com).

Mamta Rani is a Professor at the Department of Computer Science, Central University of Rajasthan, Dist-Ajmer, 305817, Rajasthan, India (email: mamtarsingh@curaj.ac.in).

for security breaches. The cryptanalysis of SHA-0 in 1998 and subsequent discoveries of near-collisions for SHA-0 in 2004, along with significant breakthroughs in 2005 including collision attacks on MD4, MD5, and SHA-1, underscored the evolving landscape of cryptographic security. Xiaoyun Wang's contributions were notably influential during this period. These disclosures, especially concerning MD5 and SHA-1, prompted a re-evaluation of hash function reliability [18] [19] [20] [21] [22] [23] [24] [25] wang2005efficient. Although NIST had already introduced the SHA-2 family, encompassing SHA-224, SHA-256, SHA-384, and SHA-512, the attacks on their predecessors raised the concerns about their durability, given the shared foundational principles. Recognizing the need for robust cryptographic solutions, NIST responsed to these challenges was to host Cryptographic Hash Workshops and launch a public competition to develop SHA-3 [26] [27]. Moreover, the cryptographic landscape has expanded with the development of other hash functions like HAVAL, RIPEMD and its successors RIPEMD-128 and RIPEMD-160, as well as Whirlpool. These additions, including Whirlpool's innovative design utilizing a block cipher similar to AES but with a larger block size, underscore the continuous search for advanced security mechanisms to address the dynamic challenges of digital cryptography [28] [21] [29] [30]. The research on hash functions was predominantly focused on enhancing security, efficiency, and applications in various domains. Notably, Poseidon presented a novel approach tailored for Zero-Knowledge proof systems [31], indicating a significant shift towards privacy-preserving cryptographic protocols. Concurrently, advancements in quantum-resistant cryptographic schemes were explored, as seen in developing subset-resilient hash function families, highlighting the anticipation of quantum computing threats [32]. Moreover, the era witnessed the exploration of hash functions in blockchain technologies, aiming at fortifying IoT privacy through enhanced cryptographic schemes [33], thereby underscoring the growing interconnection between cryptographic hash functions and emerging digital infrastructures.

The focus has notably shifted towards optimization and application-specific hash functions. For instance, Poseidon2 emerged as a faster variant of its predecessor, emphasizing performance improvements for ZK applications [34]. Research also ventured into the automated discovery of protocol attacks exploiting hash function weaknesses, highlighting the continuous battle against cryptographic vulnerabilities [35]. Moreover, the introduction of Tip5 for Recursive STARKs [36] and Anemoi Permutations [37] showcased innovative designs aimed at enhancing the efficiency and security of hash functions, catering to the growing demands of cryptographic applications. Additionally, the exploration of quantum hash functions [38] and applications in IoT [39] illustrated the broadening scope of hash function.

Recent advances in cryptographic and security frameworks highlight innovative solutions for data protection and authentication. Koroglu et al. [40] and Hassan et al. [41] contribute to hash function improvements, including FPGA-based non-cryptographic functions for network efficiency, while Shi et al. [42] introduce a quantum hash function with enhanced speed. Addressing password and key management, Somboonpattanakit and Wisitpongphan [43] propose Prime Decomposition Password Storing (PDPS), and Cardona-López et al. [44] enhance key exchange with high-entropy composite hash functions.

Further security innovations involve tamper detection and blockchain, as seen in SHA-SARIMAX by Srivatsa et al. [45] and the blockchain-based adoption framework by Pujari et al. [46]. In data clustering, Lv [47] utilizes cloud computation to optimize clustering for big data. In biometrics, Boonkrong [48] and Xuan et al. [49] enhance multi-factor authentication and fuzzy identity-based signatures. Additionally, Wang [50] improves face recognition with a lightweight neural network, while Tan et al. [51] advance image tampering detection through attention mechanisms. This collective work drives forward secure, efficient frameworks across cryptography and digital security.

III. Overview of Hash Function

MD2, MD4, MD5: All three are early cryptographic hash functions. MD2 is slow and secure, MD4 is faster but less secure, and MD5 is widely used but vulnerable to collision attacks. They are not recommended for cryptographic security due to vulnerabilities [52] [53]. SHA1: It produces a 160-bit hash value. Once widely used, it's now vulnerable to collision attacks and considered insecure for cryptographic purposes [54]. SHA224, SHA256: Part of SHA-2, these functions generate 224-bit and 256-bit hash values, respectively. They offer robust security and are widely used in various applications, including SSL/TLS and digital signatures [55]. SHA384: Another SHA-2 variant, SHA384 produces a 384-bit hash. It offers high security and is used in applications requiring stronger hash values than SHA256 [55]. SHA512/224, SHA512/256, SHA512: SHA-2 variants offering different hash lengths (224, 256, and 512 bits). SHA512 is particularly strong in security and is suitable for high-security requirements [56]. SHA3-224, SHA3-256, SHA3-384, SHA3-512: SHA-3 family, successors to SHA-2. They offer various hash lengths and are known for resistance to quantum attacks, used in future-proofing cryptographic applications [57]. RIPEMD128, RIPEMD160, RIPEMD256, RIPEMD320: RIPEMD family of hash functions with differing hash lengths. RIPEMD160 is recognised for its use in Bitcoin. They offer moderate security but are less common than SHA variants citeliu2023analysis. Whirlpool: It produces a 512-bit hash. It is known for its security and speed in cryptographic solid hashing applications [58]. Tiger128,3, Tiger160,3, Tiger192,3, Tiger128,4, Tiger160,4, Tiger192,4: Tiger is a cryptographic hash function designed for integrity checking. The numbers indicate different versions and hash lengths. It's faster than SHA-1 and MD5 but less widely adopted[59]. Snefru, Snefru256: Early cryptographic hash functions, known for their security but replaced by more advanced algorithms like SHA-2 [60]. Gost, Gost-Crypto: Russian cryptographic hash functions. Gost-Crypto is a variant with improved security. They are used primarily in Russian government applications [61]. Adler32: A checksum algorithm, fast but unsuitable for cryptographic purposes. It is commonly used in data transmission error detection [62]. CRC32, CRC32B,

CRC32C: Cyclic Redundancy Check algorithms. Fast and used for error-checking in networks and storage devices, but not for cryptographic security [63]. FNV132, FNV1A32, FNV164, FNV1A64: Fowler–Noll–Vo hash functions, non-cryptographic and used in hash tables, checksums, and unique identifiers. They offer good dispersion and speed [64]. Joaat: Jenkins's one-at-a-time hash, non-cryptographic, used mainly in hash tables for quick, simple hashing [65]. Murmur3a, Murmur3c, Murmur3f: Non-cryptographic, fast hash functions for hash tables, databases, and bloom filters. They offer high performance and good collision resistance [65]. XXH32, XXH64, XXH3, XXH128: High-speed non-cryptographic hash functions used for checksums and fingerprinting, offering high performance and low collision rates [66]. Haval128,3; Haval160,3; Haval192,3; Haval224,3; Haval256,3; Haval128,4; Haval160,4; Haval192,4; Haval224,4; Haval256,4; Haval128,5; Haval160,5; Haval192,5; Haval224,5; Haval256,5: HAVAL is a family of cryptographic hash functions that offer variable length (128 to 256 bits) and a variable number of rounds (3 to 5). They are known for flexibility but are less common than SHA or MD5 [29]. CityHash: Developed by Google in 2011, CityHash is optimised for 64-bit architectures and ideal for data structures. It's fast but non-cryptographic and not suited for high-security applications [66]. FarmHash: A Google creation in 2014, FarmHash is fast and efficient for modern CPUs and GPUs. It's great for data structures but not for high-security uses [67]. SpookyHash: Developed in 2011, it is fast and efficient for data structures with good collision resistance. However, it's vulnerable to hash flooding attacks [65]. Jenkins Hash: A non-cryptographic function from the 1990s, Jenkins hash is fast with good distribution but limited collision resistance. It's mainly used in data structures [68]. SipHash: A cryptographic function from 2012, SipHash is fast, secure, and flexible. Vulnerable to length extension attacks, it's used in message authentication [69]. Bcrypt: A password hashing function from 1999, bcrypt is secure and resistant to brute-force attacks but computationally expensive. It's widely used for password storage [70]. Scrypt: Since 2009, Scrypt has been memory-hard and resistant to large-scale attacks but computationally intensive. It's used for password storage and verification [71]. CRC64: It is a non-cryptographic function for error detection and storage in digital networks, unsuitable for cryptographic applications. It is fast and efficient for real-time applications [72]. BLAKE is a cryptographic hash function finalist in the NIST SHA-3 competition. It is known for its high speed and security. The variants (224, 256, 384, 512) refer to different output hash lengths in bits [73]. Chaos-based hash functions: These functions use chaos theory in cryptography. They are unique in their sensitivity to initial conditions, making them unpredictable and secure. Their complex behaviour is excellent for encryption. They're different from traditional hash functions like MD5 or SHA-1. Chaos-based functions are hard to crack due to their unpredictable nature. This makes them ideal for high-security data protection. They are still evolving and less common than SHA or MD variants. However, their potential in cryptographic hashing is significant [4] [74] [75].

IV. Applications Areas of Hash Functions

Cryptographic Security: This classification pertains to hash functions that ensure data security and integrity within cryptographic frameworks. These algorithms play a crucial role in processes such as encryption, the creation of digital signatures, and secure data transmission [76].

Data Management and Integrity: This category is dedicated to hash functions designed for optimizing data storage, retrieval, and integrity assurance. Their widespread application spans database management, file systems, and the efficient processing of substantial data volumes [77].

Authentication and Verification: Under this category fall hash functions that are critical in the authentication of user credentials and the verification of data authenticity. These functions are vital in the management of password storage, the issuance of digital certificates, and the generation of message authentication codes [78].

Blockchain and Distributed Systems: This segment includes hash functions that preserve integrity and security in blockchain networks and distributed systems. They are employed in tasks such as cryptocurrency mining, transaction validation, and the establishment of immutable digital ledgers [79].

Forensics and Data Analysis: This category focuses on applying hash functions in digital forensics and data analysis. These functions are utilized in the detection of malware, the verification of file integrity, and the creation of unique data identifiers crucial for forensic inquiries [80].

V. Evaluation and Categorization of Hashing Algorithms

Table I shows the hashing time for various algorithms as the file size increases from 1MB to 100MB and finally to 1GB. The time taken for hashing increases with the file size for all algorithms. However, some algorithms scale better than others with increasing file size. For instance, the algorithm represented by the blue line (md2) shows a significant increase in time for hashing when moving from 1MB to 1GB compared to the others. Most hashing algorithms show a modest increase in hashing time between 10MB and 100MB but demonstrate a more pronounced increase from 100MB to 1GB. Numerous algorithms are listed. The behaviour of each algorithm in terms of scaling with a file size can be an essential factor in their evaluation and categorization, particularly for applications that require hashing of large files where performance may be a critical factor. Figure 1 shows hashing time for major hashing algorithms at file sizes of 10MB, 100MB, and 1GB.

Table II provides a classification of various hashing algorithms based on their construction type. Each construction type represents a different methodological approach to how the hash functions process the input data to produce a hash value. Merkle–Damgård category includes well-known hash functions like MD5 and SHA1, which are widely used for data integrity verification. It is characterised by breaking the input into blocks and processing them sequentially, where each step depends on the output of the previous step. The

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

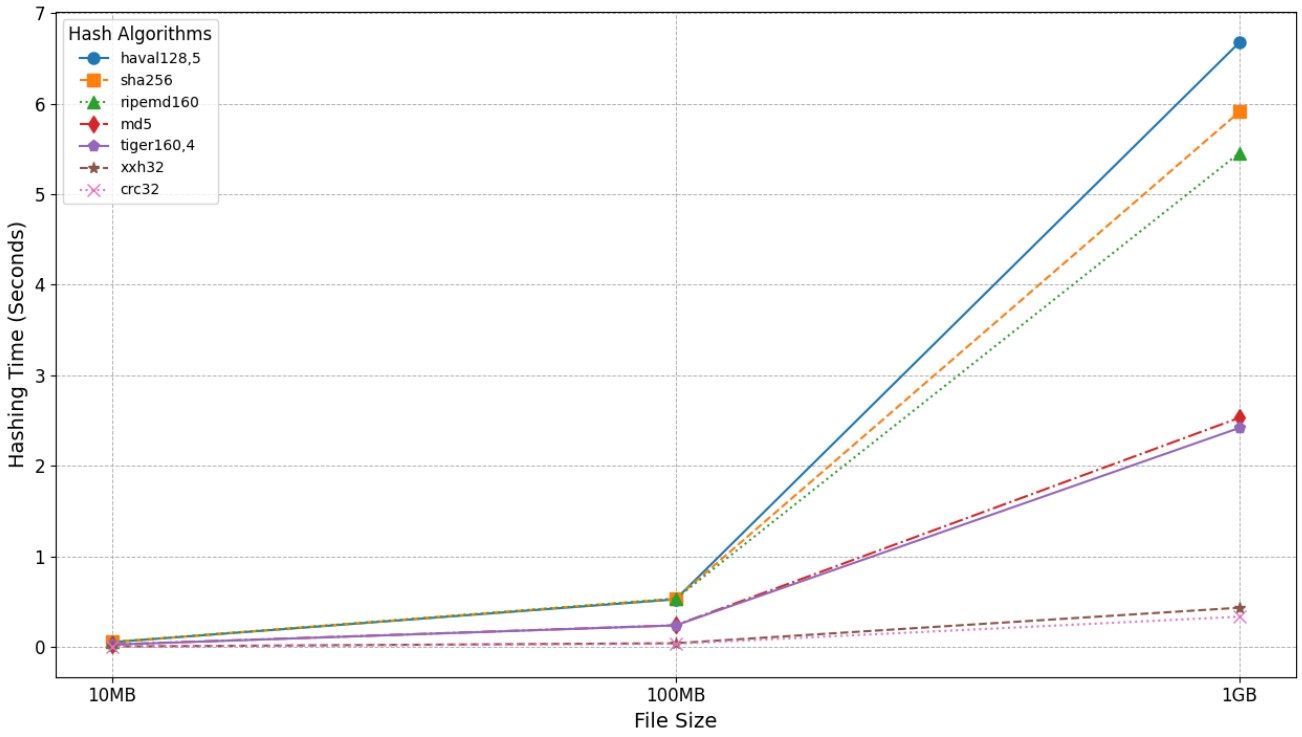

Fig. 1. Hashing Time for Different Algorithms at File Sizes of 10MB, 100MB, and 1GB.

SHA-3 family of hash functions uses sponge construction and is known for absorbing data into a state and then squeezing the hash value out of it. It's versatile and allows for variable output lengths. A Wide Pipe generally refers to hash function designs that use internal state sizes that are more significant than the output hash value. Checksum includes non-cryptographic functions like Adler32 and CRC32, mainly used for error-checking in data transmission rather than security. Algorithms such as CityHash and Murmur3 are Non-cryptographic algorithms designed for speed and used in applications like hash tables rather than for security. Bcrypt and Scrypt are known primarily for password hashing, designed to be slow and computationally expensive as a defence against brute-force attacks. Hash functions like Snefru and BLAKE don't fit into the different categories but are used for secure hashing.

Table III assesses various hash functions, evaluating them regarding standardization and compatibility. The status of the hash function in terms of standardization processes and security, with entries such as Widely Used, Obsolete, or references to specific standards like SHA-2 Family. Compatibility of hash functions in a qualitative measure regarding how well the hash function is supported across different systems and platforms, ranging from 'Low' to 'High'.

Figure 2 illustrates the avalanche effect in different hash functions, a desirable property where a slight change in the input results in a significant and unpredictable change in the output. The graph compares the performance of various hash functions based on the percentage of bits changed in the output hash when a single input bit is changed.

Figure 3 displays a horizontal bar chart that compares the hash rates of a range of hash functions, measured in millions of hashes per second. Each bar represents a different hash function, with the length of the bar indicating the hash rate performance. The hash functions are ordered vertically from the lowest to the highest hash rates. This comparison provides insight into the efficiency of each algorithm in processing data to generate hashes, which is a crucial factor in applications where processing speed is a significant concern, such as in mining cryptocurrencies or processing large volumes of data.

Table IV represents the hash values generated by various cryptographic algorithms for two inputs: "CURaj" and "CURAJ", and the hash value size in bits. This comparison highlights each hash function's sensitivity to input changes, a fundamental property known as the avalanche effect, which is crucial for ensuring that even minor variations in input produce significantly different hashes.

Table V categorizes a range of hash algorithms by their typical application domain, suitability for high-security applications, operational speed, and resistance to quantum attacks. Five application domains are identified: Cryptographic Security, Data Management and Integrity, Authentication and Verification, Blockchain and Distributed Systems, and Forensics and Data Analysis. Cryptographic Security includes SHA variants and Whirlpool, which are very suitable for high-security applications due to their wide usage in encryption and digital signatures. They possess moderate to high speed and resistance to quantum attacks. Data Management and Integrity lists MD and RIPEMD series, which are less suitable for high-security applications due to their high speeds but are ideal for data storage and checksums. The Authentication and Verification domain contains hash functions like SHA-224 and Scrypt, with suitable security properties for user authentication and data authenticity verification. Blockchain and Distributed Systems focus on algorithms like SHA-256 and BLAKE variants, which are ideal for cryptocurrency mining and transaction validation applications. The Forensics and Data Analysis section includes hash functions like the SHA-3 series, emphasizing their effectiveness for digital forensics and data integrity verification due to their strong cryptographic properties. SHA-3's enhanced resistance to quantum attacks further strengthens its suitability for forensics, where preserving long-term data integrity and authenticity is essential.

NIST statistical tests on hash values generated from 10,000 samples for each selected eight hash function. The tests evaluate the randomness of the hash outputs with three primary aspects: Bit Stream Distribution: Table VI shows the frequency distribution of 0s and 1s for different hash algorithms, indicating uniformity across bit positions. P-values: Table VII provides the p-values for each NIST test, such as Frequency, BlockFrequency, and CumulativeSums, across various algorithms, assessing the statistical significance of randomness. Proportion of Successes: Table VIII displays the success rate of tests (e.g., "10/10" indicates all tests passed) for each algorithm. SHA-256 and Whirlpool demonstrate excellent randomness properties with balanced bit distributions, high p-values, and perfect success proportions, making them highly reliable for cryptographic applications.

Figure 4 presents a bar graph depicting the distribution of hash algorithms across five key application domains: Data Management and Integrity, Cryptographic Security, Authentication and Verification, Forensics and Data Analysis, and Blockchain and Distributed Systems. The x-axis categorizes these domains, each representing a unique context in which hash algorithms are utilized, while the y-axis quantifies the number of algorithms applied within each area. The tallest bars are seen in the domains of Data Management and Integrity and Cryptographic Security, suggesting a heavier reliance on hash algorithms in these fields to ensure data integrity, consistency, and protection. This high usage likely reflects the essential role that hash algorithms play in maintaining secure data management and encryption standards. Conversely, the shorter bars for Authentication and Verification, Forensics and Data Analysis, and Blockchain and Distributed Systems indicate a more limited or specialized application of hash algorithms. This may suggest that hash algorithms in these fields are applied selectively, tailored to specific needs related to identity verification, forensic analysis, and decentralized data handling, rather than widespread usage. Overall, Figure 4 provides a comprehensive view of how hash algorithms are deployed differently across domains, underscoring both the widespread and targeted roles these algorithms play based on the requirements of each field.

TABLE II
CLASSIFICATION OF HASH FUNCTIONS BASED ON CONSTRUCTION TYPE

Construction Type	Hash Functions
Merkle-Damgard	MD2, MD4, MD5, SHA1, SHA224, SHA256, SHA384, SHA512, SHA512/224, SHA512/256, RIPEMD128, RIPEMD160, RIPEMD256, RIPEMD320, Tiger, Gost, Haval
Sponge	SHA3-224, SHA3-256, SHA3-384, SHA3-512
Wide Pipe	Whirlpool
Checksum	Adler32, CRC32, CRC64
Non-cryptographic	FNV1, Joaat, Murmur3, XXHash, CityHash, FarmHash, SpookyHash, Jenkins
Password Hashing	Bcrypt, Scrypt
Other Cryptographic	Snefru, BLAKE

TABLE III
STANDARDIZATION AND COMPATIBILITY ANALYSIS OF HASH FUNCTIONS

Hash Function	Standardization	Compatibility
MD2	Obsolete	Low
MD4	Obsolete	Low
MD5	Widely Used, Not Secure	High
SHA1	Insecure, Phasing Out	High
SHA224	SHA-2 Family	High
SHA256	SHA-2 Family	High
SHA384	SHA-2 Family	High
SHA512	SHA-2 Family	High
SHA512/224	SHA-2 Variant	Moderate
SHA512/256	SHA-2 Variant	Moderate
SHA3-224	SHA-3 Family	Moderate
SHA3-256	SHA-3 Family	Moderate
SHA3-384	SHA-3 Family	Moderate
SHA3-512	SHA-3 Family	Moderate
RIPEMD128	Less Common	Low
RIPEMD160	Used in Bitcoin	Moderate
RIPEMD256	Less Common	Low
RIPEMD320	Less Common	Low
Whirlpool	ISO/IEC 10118-3:2004	Moderate
Tiger	Less Common	Moderate
Snefru	Obsolete	Low
Gost	Russian Standard	Low
Adler32	Checksum, Non-Cryptographic	High
CRC32	Checksum, Non-Cryptographic	High
FNV1	Non-Cryptographic	High
Joaat	Non-Cryptographic	High
Murmur3	Non-Cryptographic	High
XXHash	Non-Cryptographic	High
Haval	Less Common	Low
CityHash	Non-Cryptographic, Google	High
FarmHash	Non-Cryptographic, Google	High
SpookyHash	Non-Cryptographic	High
Jenkins	Non-Cryptographic	High
SipHash	Cryptographic, Recent	Moderate
Bcrypt	Widely Used in Password Storage	Moderate
Scrypt	Widely Used in Password Storage	Moderate
CRC64	Checksum, Non-Cryptographic	High
BLAKE	SHA-3 Finalist	Moderate
Chaos based hash functions	Emerging Field	Uncertain

VI. NIST RECOMMENDATIONS AND NEW USE CASES

The National Institute of Standards and Technology (NIST) is a leading authority in developing and maintaining cryptographic standards to ensure data security and integrity in digital systems. Focusing on security, efficiency, and future-proofing against developing threats, NIST offers clear recommendations for the use of hash functions in cryptographic applications. For general cryptographic use, the SHA-2 family—which comprises SHA-224, SHA-256, SHA-384, and SHA-512—is highly recommended [81]. Because of their strong security features and fit with current systems, these algorithms are becoming very popular. Apart from SHA-2, the SHA-3 family—which comprises SHA3-224, SHA3-256, SHA3-384, and SHA3-512—is also a substitute, particularly fit for settings needing better resistance to possible future weaknesses, such as those presented by developments in quantum computers [82].

Since earlier hash methods like MD5 and SHA-1 have major flaws, NIST clearly warns against using them. MD5 is unfit for safe cryptographic operations as it is very vulnerable to pre-image attacks and collisions. Likewise, SHA-1 is no longer regarded as safe for important uses, such as digital signatures or authentication, because it has been realistically broken through demonstrated collision attacks [83].

NIST suggests selecting hash algorithms that satisfy the security needs of the application for certain cryptographic usage situations. Using SHA-2 or SHA-3, for instance, guarantees in digital signatures that the hash output length

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	72fb20f47e4d4df2f2961e9468d727c937d6 c639	63ca79292c3d4154424a135558aba57c93b48808
sha256	256	068b7d496eff1717aa22346b90961ac3da5 d8c33b859e9da71927d19dedc2953	0f76630016ee1099564d837f1e6d949979 a67ad2184cc0795df7f94ac3110fe8
sha384	384	a6d20eca21eda60cfa01f00e29ffae348a a035d1b7a7cbfa734961c39f6c71e2d7c7 0aa8ace1fef8c4de97dad8e8b0de	59bcde11a94f338d506792cdd54bbbc4f8 e0684b8e0c61c5be13dca4e656e442828b 79da46fab66253b5adb35f1b0086
sha512/224	224	e76d74d8dd29e08ad5d621d1f4d633838 bcd4780e5926644953596e9	503f71d3388b72b795d2209ee1476ea7 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	49bf12c6f402a526ac8ddb0067b8b6df4e3bec0	d314c22851cfad4f5ae7f57e632de8e0ed6efce7
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	4ce20b0483cbbeec6c530edf551d030418510dd0	60ef2abd088c6c2f51bd2db12623b809fb91868a
tiger160,4	160	c43ae734675c6c6f583930f502d674de6aaef275	45a8018698dbd6fa219b027719079e6ae9f8905a
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	7db36c2704f972388fb64ea63b333da87bd20776	1ffec659e65e91a63f6f302aa19cc999e9e6ae87
haval192,3	192	da683dd25ab130471a8ce8fc7fe5e3d1b b3e1ab60b7a78a6	90fa535008acf10a073c512acca4df9c 14861a283866183b
haval160,4	160	683774205ae9cf80436a7116 701b9b9de3d05aed	69afd4d4ce0a1f535d9365e5 6c346940b6aa8a2a
haval192,4	192	f428cea675ed696ffa9407826 02cec8c0196df136b6ba818	e143d50baa55f18e0070c8e6 e6a658dfd777cae3286b4ecc
haval224,4	224	f1f6dc8ebcfcf3cb8d3d2a958b1ac76122d9a32bc d957ebc54da969f	a202d380acce8ccc847b747f6b7e4adb6af5d992 43ece266b555ef8f
haval256,4	256	e30a845a64826bf66e4cb948972a5b02694c4d d666240a8ce6bb312fb29dba73	32315e9dfc0067052d6f8388b180e3541aa2d cf98acee9c54f379ad7c9017434

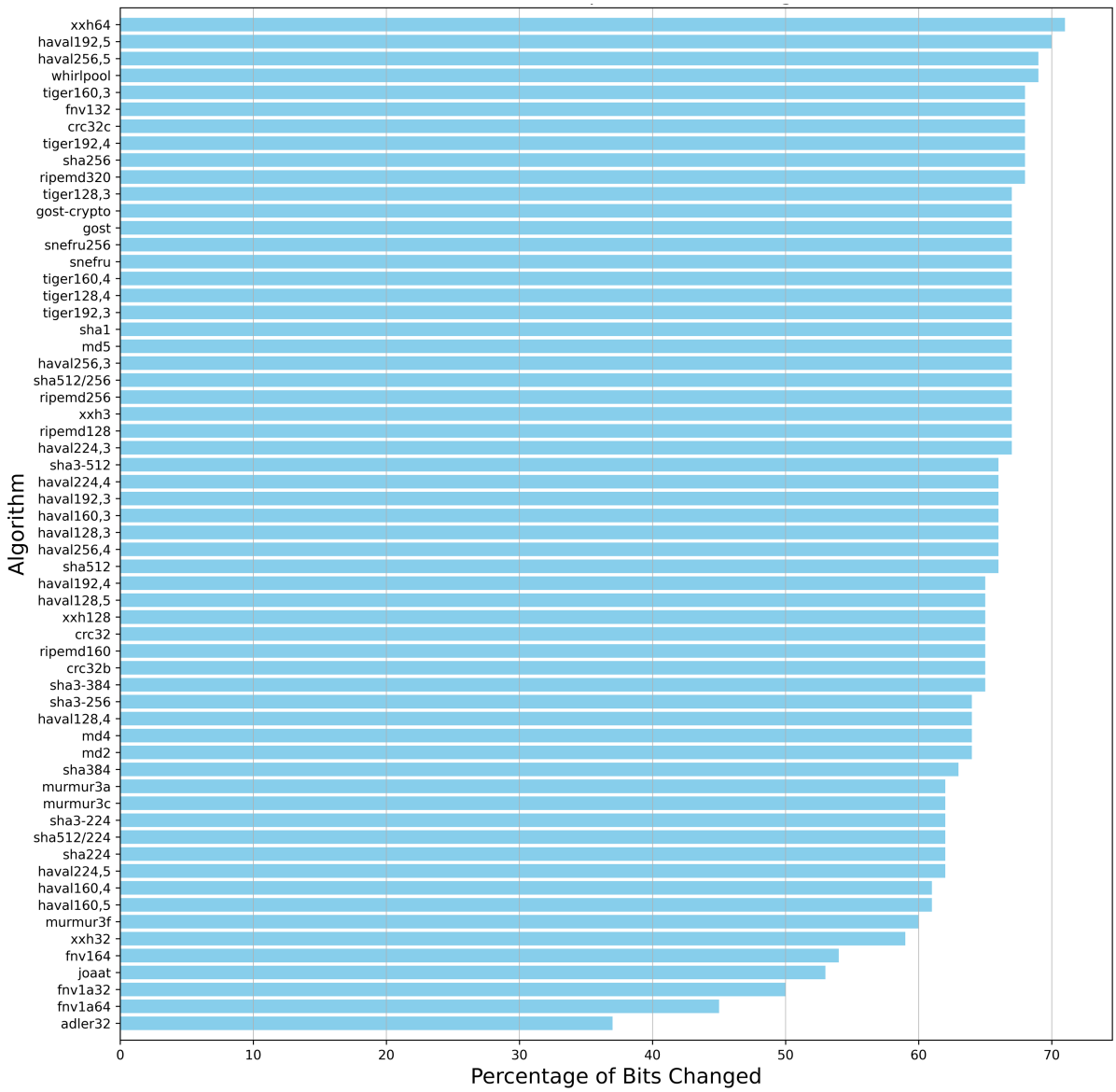

Fig. 2. Comparative Analysis of Avalanche Effect in Hash Functions

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 encryption, digital signatures, secure 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, malware detection

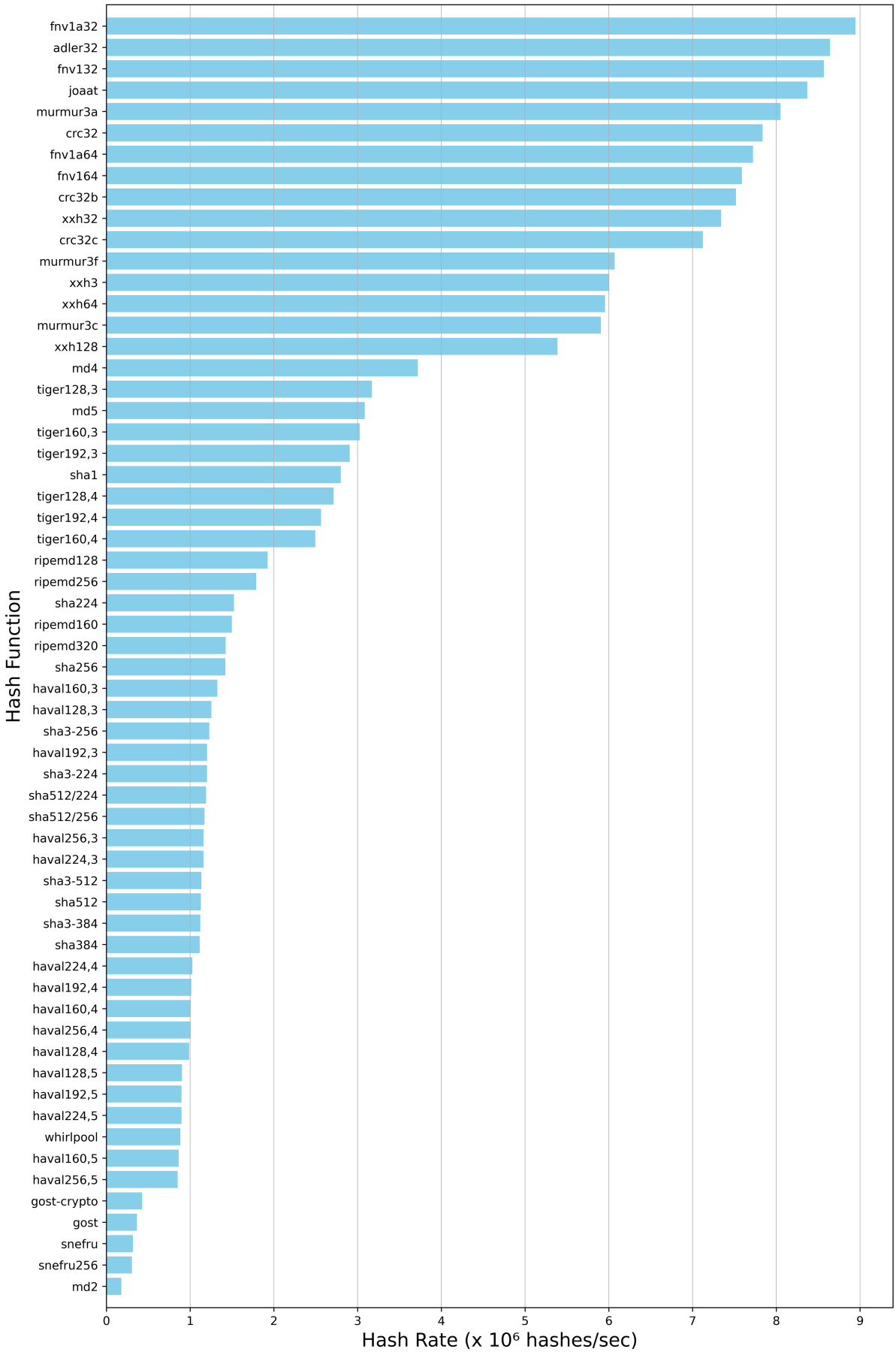

Fig. 3. Performance Comparison of Hash Functions by Hash Rate

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

corresponds with the intended security level of the signature technique [81]. Similarly, for HMAC (Hash-Based Message Authentication Codes) and key derivation functions (KDFs), SHA-2 is still the recommended option because of its great security and computational efficiency [84]. While NIST does not prescribe particular methods for securely storing passwords, specialized password-hashing systems like bcrypt or scrypt are advised over generic cryptographic hashes.

NIST highlights SHA-3's durability against theoretical flaws and quantum computing developments in addressing newly arising concerns [82]. While SHA-2 is still the most sensible and efficient option for ordinary usage, the sponge architecture employed in SHA-3 makes it especially strong in circumstances needing sophisticated security. Furthermore, NIST underlines the need to select the hash function's output length depending on the necessary security degree. For applications with strong security requirements or to reduce dangers from next-generation computational developments, SHA-512 is a perfect choice.

NIST also promotes adherence to accepted standards such as FIPS PUB 180-4 (Secure Hash Standard) and guidance in NIST Special Publication 800-57. These guarantee that cryptographic systems follow best practices and satisfy legal criteria [81], [84]. Following these guidelines will help organizations properly protect digital security, authentication, and data integrity against evolving threats.

Hash Functions could be used at various other places. as we know, web crawlers and scrapers are important for analysing and indexing web information. But they often use up a lot of computer resources and traffic. It is better to include both the URL and a hash of the HTML text for each page in the sitemap. This way not only makes things run more smoothly, but it also makes security and content control better. Each page's URL and a hash of its HTML text are stored in the suggested index structure. For example, if the content of a webpage changes, so will its hash. This lets robots focus on only the new content. This stops reading pages that haven't changed, which saves bandwidth and speeds things up. Crawlers can compare the current hash to the one they have saved, so they only get the pages that have been changed. This method not only improves searching, but it also lowers server load and stops wasted data. Implementing a hash-based sitemap helps websites stay compatible with search engines while also making it easier and smarter for crawlers to do their jobs. This approach works especially well for websites that are big or that are changed often, since reducing the number of unnecessary calls is very important. Assume that we wish to find out whether the content of any Wikipedia page has been

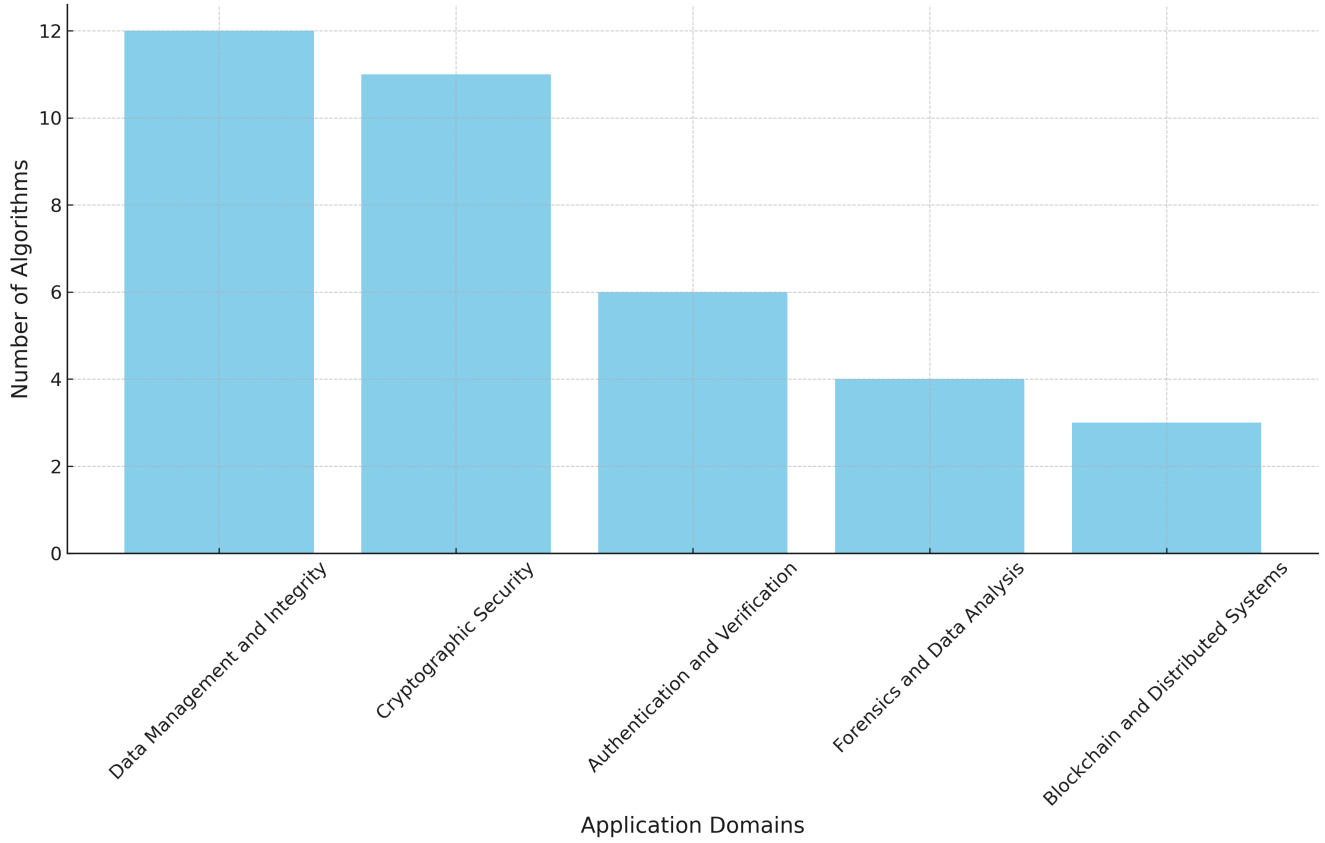

Fig. 4. Distribution of Hash Algorithms Across Application Domains

changed. We would need to browse and record the content of all relevant pages, and then compare each page with its previously stored version to identify changes. This procedure is resource-intensive, needing multiple queries to retrieve the entirety of each page. Nevertheless, this task is significantly more efficient when utilising a hash-based sitemap. A single request is sufficient to retrieve the sitemap when employing a hash-based sitemap.

VII. Conclusion and Future Work

Our study provides an in-depth evaluation and categorization of cryptographic hashing algorithms, highlighting their crucial role in digital security across diverse applications. By examining traditional and chaos-based hash functions, we have identified critical considerations for choosing appropriate hashing algorithms, focusing on their performance, security, and applicability. Our findings underscore the importance of balancing efficiency and security in response to evolving digital threats. This research contributes to the field of digital cryptography by offering guidance on selecting hashing algorithms suited to specific security requirements and applications, paving the way for future advancements in secure and efficient cryptographic solutions. In future, we will focus on following:

- Chaos-Based and Quantum-Resistant Algorithms: Continued research on how chaos theory can be used and how to make hashing methods that are not affected by quantum mechanics. Combining these methods into hybrid models might make them more resistant to advanced cryptoanalysis. Combining quantum physics and chaos theory could lead to new, robust coding methods that are good for security after quantum computing.
- Adaptable Hash Systems: Future hashing methods should include adaptable systems that can change with threats in real time and keep security high.
- Efficiency in Real-World Applications: Making sure that new hash functions are easy to use on various devices will ensure that they can be used in real-world systems.
- For the future of safe security systems, it is essential to work on chaos-based and quantum-resistant hashes and flexible and quick solutions.

References

[1] H. Ahmed, "A review of hash function types and their applications," *Wasit Journal of Computer and Mathematics Science*, vol. 1, no. 3, pp. 120–139, 2022.
[2] 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.
[3] L. E. Hughes, "Basic cryptography: Hash function," in *Pro Active Directory Certificate Services: Creating and Managing Digital Certificates for Use in Microsoft Networks*, pp. 19–22, Springer, 2022.
[4] M. Rasool and S. B. Belhaouari, "From collatz conjecture to chaos and hash function," *Chaos, Solitons & Fractals*, vol. 176, p. 114103, 2023.
[5] W. Diffie and M. E. Hellman, "New directions in cryptography," in *IEEE Transactions on Information Theory*, pp. 644–654, 1976.
[6] R. C. Merkle, "Secure communications over insecure channels," *Communications of the ACM*, vol. 21, no. 4, pp. 294–299, 1978.
[7] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," 1979.
[8] S. M. Matyas, "Generating strong one-way functions with cryptographic algorithm," *IBM Technical Disclosure Bulletin*, vol. 27, pp. 5658–5659, 1985.
[9] C. H. Meyer and M. Schilling, "Secure program load with manipulation detection code," in *Proc. Securicom*, vol. 88, pp. 111–130, 1988.

[10] B. Kaliski, "Rfc1319: The md2 message-digest algorithm," 1992.

[11] J. Linn, "Rfc1115: Privacy enhancement for internet electronic mail: Part iii-algorithms, modes, and identifiers," 1989.

[12] R. L. Rivest, "The md4 message-digest algorithm. internet request for comments," tech. rep., April 1992. RFC 1320, 1990.

[13] R. L. Rivest, "The md4 message digest algorithm," *Lecture notes in Computer Science, Advances in Cryptography CRYPTO '90*, 1998.

[14] R. Rivest, "The md5 message-digest algorithm, internet request for comments: Rfc 1321," *Internet Engineering Task Force*, 1992.

[15] N. I. of Standards and T. U. T. Administration, *Secure hash standard*, vol. 180. US Department of Commerce, Technology Administration, National Institute of . . . , 1993.

[16] A. J. Menezes and S. A. Vanstone, *Advances in Cryptology-CRYPTO'90: Proceedings*, vol. 537. Springer, 2003.

[17] R. Rivest, "Rfc1321: The md5 message-digest algorithm," 1992.

[18] 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, Springer, 2007.

[19] 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.

[20] 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.

[21] H. Dobbertin, A. Bosselaers, and B. Preneel, "Ripemd-160: A strengthened version of ripemd," in *International Workshop on Fast Software Encryption*, pp. 71–82, Springer, 1996.

[22] F. Chabaud and A. Joux, "Differential collisions in sha-0," in *Annual International Cryptology Conference*, pp. 56–71, Springer, 1998.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] R. F. Kayser, "Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (sha-3) family," *Federal Register*, vol. 72, no. 212, p. 62, 2007.

[28] A. Bosselaers and B. Preneel, *Integrity Primitives for Secure Information Systems: Final Ripe Report of Race Integrity Primitives Evaluation*. No. 1007, Springer Science & Business Media, 1995.

[29] Y. Zheng, J. Pieprzyk, and J. Seberry, "Haval—a one-way hashing algorithm with variable length of output," in *Advances in Cryptology—AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques Gold Coast, Queensland, Australia, December 13–16, 1992 Proceedings 3*, pp. 81–104, Springer, 1993.

[30] 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.

[31] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for {Zero-Knowledge} proof systems," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 519–535, 2021.

[32] 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.

[33] Y. P. Khanal, A. Alsadoon, K. Shahzad, A. B. Al-Khalil, P. W. Prasad, 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.

[34] L. Grassi, D. Khovratovich, and M. Schofnegger, "Poseidon2: A faster version of the poseidon hash function," *Cryptology ePrint Archive*, 2023.

[35] V. Cheval, C. Cremers, A. Dax, L. Hirschi, C. Jacomme, and S. Kremer, "Hash gone bad: Automated discovery of protocol attacks that exploit hash function weaknesses," in *32nd USENIX Security Symposium*, 2023.

[36] A. Szepieniec, A. Lemmens, J. F. Sauer, B. Threadbare, *et al.*, "The tip5 hash function for recursive starks," *Cryptology ePrint Archive*, 2023.

[37] 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.

[38] 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.

[39] 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.

[40] T. Koroglu and R. Samet, "Can there be a two way hash function?," *IEEE Access*, vol. 12, pp. 18358–18386, 2024.

[41] 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.

[42] 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.

[43] C. Somboonpattanakit and N. Wisitpongphan, "Secure password storing using prime decomposition," *IAENG International Journal of Computer Science*, vol. 48, no. 1, pp. 152–160, 2021.

[44] M. A. Cardona-López, J. C. Chimal-Eguía, V. M. Silva-García, and 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.

[45] A. Srivatsa, T. Ananthapadmanabha, L. K. MV, and A. Suma, "Enhancing 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.

[46] C. Pujari, C. CB, S. R. Muppidi, and M. C. Belavagi, "A novel method of secure child adoption using blockchain technology.," *IAENG International Journal of Applied Mathematics*, vol. 53, no. 4, pp. 1531–1539, 2023.

[47] 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.

[48] 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.

[49] 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.

[50] S. Wang, "A face recognition method based on lightweight neural network and multi hash recognition degree weighting.," *IAENG International Journal of Applied Mathematics*, vol. 54, no. 3, pp. 581–586, 2024.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] E. Swathi, G. Vivek, and G. S. Rani, "Role of hash function in cryptography," *Int. J. Adv. Eng. Res. Sci.(IJAERS)*, 2016.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] P. Koopman, K. Driscoll, and B. Hall, "Selection of cyclic redundancy code and checksum algorithms to ensure critical data integrity," 2015.

[64] C. Hayes, *Non-Cryptographic Hash Functions: Focus on FNV*. PhD thesis, National University of Ireland Maynooth, 2023.

[65] 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.

[66] M. Cheng, Y. Wu, X. Zhou, J. Li, and L. Zhang, "Efficient web archive searching," 2020.

[67] 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.

[68] 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.

[69] 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.

[70] N. Provos and D. Mazieres, "Bcrypt algorithm," in *USENIX*, 1999.

[71] C. Percival and S. Josefsson, "The scrypt password-based key derivation function," tech. rep., 2016.

[72] 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.

[73] N. Mouha, "Exploring formal methods for cryptographic hash function implementations," in *Australasian Conference on Information Security and Privacy*, pp. 177–195, Springer, 2023.

[74] 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.

[75] 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.

[76] B. Preneel, "Cryptographic hash functions," *European Transactions on Telecommunications*, vol. 5, no. 4, pp. 431–448, 1994.

[77] M. R. Anwar, D. Apriani, and I. R. Adianita, "Hash algorithm in verification of certificate data integrity and security," *Aptisi Transactions on Technopreneurship (ATT)*, vol. 3, no. 2, pp. 181–188, 2021.

[78] G. Hatzivasilis, "Password-hashing status," *Cryptography*, vol. 1, no. 2, p. 10, 2017.

[79] X. Zhao, Z. Lei, G. Zhang, Y. Zhang, and C. Xing, "Blockchain and distributed system," in *Web Information Systems and Applications: 17th International Conference, WISA 2020, Guangzhou, China, September 23–25, 2020, Proceedings 17*, pp. 629–641, Springer, 2020.

[80] 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.

[81] 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.

[82] National Institute of Standards and Technology (NIST), "SHA-3 Finalist Evaluation Report," tech. rep., National Institute of Standards and Technology (NIST), 2012.

[83] 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.

[84] National Institute of Standards and Technology (NIST), "Recommendation 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.