ChaoticRIPE: Strengthening RIPEMD-160 with the Chirikov Standard Map for Enhanced Cryptographic Security

Suparn Padma Patra[1] and Mamta Rani[2]

[1,2]Central University of Rajasthan, Ajmer, Rajasthan 305817

Abstract. Advancements in communication and storage technologies need robust encryption tools to protect data. Cryptographic hash functions are crucial in maintaining data integrity and security by turning variable-length inputs into fixed-size digests. RIPEMD-160 is a popular hash algorithm that balances speed and security. However, advances in cryptanalysis have revealed weaknesses, requiring upgrades to bolster its defences against new dangers. Our research offers a new method to boost RIPEMD-160's security using the Chirikov Standard Map's chaotic properties, which increase unpredictability. We developed the ChaoticRIPE algorithm, incorporating the Chirikov Standard Map into RIPEMD-160. The experimental analysis demonstrates that ChaoticRIPE exhibits improved resistance to cryptographic attacks, heightened sensitivity to input variations, and a more uniform hash distribution than the original RIPEMD-160. The method maintains efficiency with a minimal computational overhead of approximately 0.05 milliseconds per hash computation. ChaoticRIPE is a potential enhancement for cryptographic hash functions used in modern security applications; National Institute of Standards and Technology (NIST) statistical testing results show the suggested technique's resilience.

Keywords: RIPEMD-160, Chirikov Standard Map, Chaotic Maps, Cryptographic Hash Functions

1 Introduction

The rapid development of digital storage and communication systems in recent years has made it clear how important strong and safe security methods are for keeping data's quality and authenticity. One important part of these systems is the cryptographic hash functions, which take raw data of different lengths and turn it into a fixed-size digest that can be used for many things, like verifying messages, creating digital signatures, and storing passwords [8].

By adding chaos to the hash, researchers aim to increase security. Ayubi *et al.,* [4] proposed a cryptographic hashing algorithm that uses a complicated quadratic map with chaotic behaviour in both the real and imaginary parts. Its long key length and sensitivity to small changes improve the map's resilience and security. Various analyses demonstrate the function's effectiveness and safety. To address problems with current structures like multi-collision and length extension attacks, Zellagui *et al.,* [33] suggested a new hash function that uses a sponge structure and two chaotic maps. It has statistically solid features and excellent resilience to collision attacks, making it appropriate for cloud computing and digital signatures.

Zellagui *et al.,* [32] integrated Henon map in MD4 hashing algorithm in order to enhance the security of original MD4 algorithm. Their method offered a possible enhancement to MD4 by improving statistical distribution and chaotic performance along with increasing collision resistance. Similarly, RIPEMD-160, a popular cryptographic hash algorithm known for its decent security and speed was created to improve the original RIPEMD[27]. It offers better protection against brute-force attacks [16]. However, the security of the existing hashing algorithms has come under scrutiny due to the ongoing improvements

David C. Wyld et al. (Eds): SIPO, BDHI, NET, SOEA, CSML, AISCA, MLIOB, DNLP – 2025
pp. 169-182, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.150416

in cryptanalysis methods[17]. As a result, researchers are exploring novel techniques to increase the security of hash functions that are frequently used.

One strategy is implementing chaotic dynamic processes into the hashing algorithms used in cryptography. Chaotic systems are known for ergodicity, sensitivity to initial circumstances, and mixed-phase space that enhances their unpredictability and defense against various attacks [31]. In this paper, we have introduced a technique that uses the chaotic dynamics of the Chirikov Standard Map to increase the security of the RIPEMD-160 hash function.

The paper has been organized as follows. We begin by reviewing previous studies in Section 2. In Section 3, the preliminaries are given. Section 4 discusses the proposed model. In Section 5, we have analyzed the results of the experiments. Section 6 concludes the article.

2 Literature Review

Grassi *et al.* [13] explored the advancements in zero-knowledge (ZK) proof systems for computational integrity, focusing on arithmetization-friendly hash functions. Specifically, they introduced Poseidon2, an optimized version of Poseidon, featuring a flexible instantiation as either a sponge or compression function. Poseidon2 incorporates more efficient linear layers, reducing multiplications by up to 90% and constraints in Plonk circuits by up to 70%. Addressing security concerns, they proposed a modification to thwart algebraic attacks on both Poseidon and Poseidon2. Poseidon2's security and efficiency improvements make it a leading arithmetization-oriented hash function, mitigating known vulnerabilities and demonstrating practical advantages in computational integrity-proof systems. Similarly, Alan *et al.* [28] presented Tip5, an arithmetization-oriented hash function with $p = 2^{64} - 2^{32} + 1$ components that employ the field-specific SHARK design technique. The arithmetization of Tip5 is described in the context of particular design restrictions, motivated by the recursive verification of STARKs. Three design techniques are compared in the suggested methodology: Marvellous, Hades, and Reinforced Concrete. The SHARK strategy with full S-box layers and MDS matrices is finally adopted for Tip5.

Moreover, Bouvier *et al.* [7] proposed Anemoi, a family of ZK-friendly permutations designed for efficient hash functions and compression functions, catering to cryptographic protocols such as Zcash, Monero, and others. Anemoi did well, with R1CS constraints that were two times better than Poseidon and Rescue-Prime's, a drop of 21% to 35% in Plonk constraints, and, based on the field size, going two to three times faster than Rescue-Prime. They came up with a new way to work and a new S-box structure called Flystel. Both of these meet the design needs of arithmetization-oriented hash functions.

Pibiri and Trani [22] introduced a new way to build PTHash, a minimal perfect hash function (MPHF), with the goal of making it work well in a wide range of situations. PTHash is crucial for applications in computing, such as search engines and databases, offering fast evaluation and minimal space consumption (2-3 bits/key). The proposed algorithm enables multi-threading and external-memory processing, addressing the need to scale efficiently to large datasets. Numerous tests conducted on real-world string collections show that PTHash performs exceptionally well in search times while being competitive in construction times and space usage.

Sideris *et al.* [26] focuses on enhancing the throughput rate of the Keccak hash algorithm, also known as SHA-3, by proposing a novel architecture based on FPGA devices (Virtex-5, Virtex-6, and Virtex-7). The Keccak algorithm is recognized for its excellent hardware performance and resistance to cryptanalysis. The proposed methodology involves

optimizing the Keccak algorithm through unrolling and pipe-lining techniques, introducing a new Round Constant (RC) generator format, and conducting thorough validation using NIST examples.

Mishall [2] addressed securing health data exchange in Internet of Things (IoT) applications, particularly within Wireless Sensor Networks in the electronic health sector. Based on Elliptic-curve Diffie–Hellman (ECDH) and QUARK hash, the proposed key exchange protocol aims to balance security and performance. The methodology involves strict rules for parameter security, hiding public key exchange, integrating QUARK with ECDH, and conducting a comprehensive security analysis using the Scyther tool. The proposed protocol successfully addresses security vulnerabilities, and the performance evaluation demonstrates efficiency compared to existing methods.

Nicky Mouha [20] focused on identifying vulnerabilities in cryptographic hash function implementations, specifically revisiting the finalists of the NIST SHA-3 competition. The paper discussed the recent discovery of a buffer overflow in SHA-3 (Keccak) in the extended Keccak Code Package (XKCP), affecting Python, PHP, and other projects. The study introduced a novel approach using formal methods, including symbolic execution with the KLEE framework, to find vulnerabilities in SHA-3 finalists (BLAKE, Keccak, Grøstl) and Apple's CoreCrypto library.

Stefano and Chenzhi [29] introduced two-round multi-signatures and threshold signatures, demonstrating security based on the plain discrete logarithm problem or the RSA assumption, relying on random oracles. The proposed protocols are partially non-interactive, with the first round independent of the message. Building on efficient discrete-logarithm-based schemes, the authors extend MuSig2 and FROST to incorporate linear hash functions, allowing security under either discrete logarithm or RSA assumptions. The results proposed a general framework for transforming schemes secure under one more discrete logarithm into those secure under plain DL or RSA assumptions.

Masrat and Samir [23] proposed a chaos-based hash function, leveraging the generalized Collatz process to enhance security in cryptographic applications. The proposed method incorporates chaotic variables governed by cryptographic keys, optimizing the design for desirable properties such as randomness, collision resistance, uniformity, and sensitivity to initial conditions. Through extensive evaluations, including comparisons with SHA-3 and SHA-2, the proposed hash function consistently outperforms existing alternatives, demonstrating superior statistical features, collision resistance, and efficiency. The algorithm exhibits robustness against common attacks, making it promising for digital signatures and data integrity applications. The study underscores the effectiveness of the proposed chaos-based hash function, positioning it as a reliable solution for cryptographic systems in real-world scenarios.

Peyman *et al.* [5] introduced a generalized chaotic map based on a complex quadratic map for designing chaotic hash functions in cryptographic systems. The proposed map exhibits chaotic behavior in real and imaginary parts, ensuring a high key length for cryptographic security. Dynamical tests, including bifurcation diagrams and Lyapunov exponent analyses, confirm the chaotic nature of the map. The proposed method successfully generates hash codes ranging from 32 to 2048 bits, with a key length of 234 bits.

Salwa *et al.* [24] addressed enhancing blockchain security by proposing a modified SHA-256 hash algorithm. The modification occurs after public key generation and is based on four chosen chaotic maps and DNA sequences, increasing the complexity between the original message and hash digest for heightened security. The proposed algorithm's efficiency is assessed, analyzed, and compared to SHA-256, considering confusion, diffusion,

and distribution properties. Security performance is evaluated through collision analysis, demonstrating improved robustness compared to SHA-256.

Hesam *et al.* [21] proposed a novel image encryption method, combining chaos functions and an evolutionary algorithm for enhanced security. Chaos functions contribute random occurrences and sensitivity to initial values, ensuring a secure encryption process. The evolutionary algorithm optimizes layout and mapping to enhance image entropy. Image components are disrupted using the evolutionary algorithm, coding rules, and logistic mapping with an initial value from a hash function. The method exhibits good speed with simple operators (Addition and XOR) and resistance to attacks due to a 256-bit hash function and a large search space for the evolutionary algorithm. The proposed encryption algorithm offers high security, resistance against differential attacks, and uncertainty in decryption, making it superior to other algorithms. The decryption involves solving equations with two secret keys resembling a digital signature, offering a reversible alternative for encryption algorithms.

Jiandong *et al.* [18] introduced a Spark-based hash function and a two-dimensional linked dynamic integer tent map to deal with security and speed issues that come up when working with big amounts of raw data. The approach involves partitioning plaintext in the Spark platform, parallel processing of data blocks, and utilizing a Merkle tree structure for compression. The compression function employs a two-dimensional coupled image lattice with a dynamic integer tent map and additional dynamic parameters to enhance obfuscation.

In order to overcome shortcomings in conventional 1D maps, Abdullah *et al.* [3] presented a one-dimensional (1D) chaotic map with three control parameters. Using numerical techniques, the new map's dynamic behavior is examined; bifurcation diagrams and Lyapunov exponent tests show a complex and varied behavior. In a way to secure images, the suggested map is used to make a pseudo-random number. A lot of statistical tests, like mean square error (MSE), peak signal-to-noise ratio (PSNR), NPCR, and UACI tests on 28 shots, show that the encryption system is strong. In safe image cryptography applications, the newly presented 1D map works well and offers a more secure option than conventional chaotic 1D maps. Keyspace analysis facilitates high randomness, homogeneous pixel distribution, sensitivity to small vital changes, and resilience against brute force attacks.

Gaurav *et al.* [12] introduced a hash-based secure chaotic steganography technique for concealing secret information within a cover image. The proposed method utilizes a hash function to compute non-LSB positions for hiding secret data bits. Encoding the secret involves chaotic sequences, and the randomness of these sequences is validated using the NIST test suite. Standard statistical validation tests, including PSNR, Euclidean distance, histogram analysis, and SSIM index, demonstrate the satisfactory quality of the stego image.

Yu-Jie *et al.* [19] proposed a hash function construction scheme combining a two-dimensional coupled map lattice and a dynamic integer tent map. The dynamic integer tent map is the nonlinear function for the two-dimensional coupled map lattice, with dynamic parameters added to enhance security. The bit logic decision function of the dynamic integer tent map controls the change in dynamic parameters. Test results demonstrate the hash function's strong security, simple implementation, and potential as an ideal replacement for traditional hash functions. The algorithm offers flexibility by allowing users to select different output lengths (128 bits, 256 bits, or 512 bits) based on their requirements.

Hang *et al.* [25] used a 4-D chaotic system based on a flux-controlled memristor model to show a chaos-based image encryption method. The proposed algorithm integrates a

Hash process using the MD5 algorithm to disturb the initial values of the chaotic system, enhancing plaintext sensitivity and security. Additionally, S-box substitution and bit-XOR operations are introduced to further scramble the pixel values and improve the algorithm's security. Several measures, such as information entropy, association coefficient, greyscale histogram, plaintext sensitivity, key sensitivity, and ciphertext sensitivity, show that the suggested encryption method works well.

Emmanuel *et al.* [1] focused on enhancing information security by implementing the RSA and ElGamal cryptographic algorithms, accompanied by the SHA-256 hash function for digital signature formulation. The primary objective is to authenticate shared data and ensure its integrity. The methodology involves implementing the RSA and ElGamal cryptographic algorithms using the C programming language and incorporating the SHA-256 hash function for digital signatures. The study emphasized the importance of information security in the face of increasing cybercrimes, piracy, scams, and fraud. The implemented cryptographic algorithms and hash functions aim to protect sensitive data, giving users control over their information. The study recommends further implementation for secure submission, storage, and extraction operations to protect sensitive data comprehensively.

Susila *et al.* [30] addressed the critical data integrity issue in the IoT security context, focusing on lightweight cryptographic hash functions suitable for resource-constrained devices. The study contributed by conducting a comprehensive survey of state-of-the-art lightweight cryptographic hash functions up to early 2022. They classified design trends, providing insights into diverse development approaches. The analysis and comparison of these functions consider cryptographic properties and implementation aspects. Challenges in designing lightweight cryptographic hash functions are discussed, and potential gaps in future research are identified. Recently, Considering the importance of biometric security, [9] proposed Chaos-based hashing scheme for cancellable biometrics security whereas [15] utilized chaotic map for more better way to perform image encryption.

3 Preliminaries

In this section, we have discussed the existing hashing algorithms that are required for understanding our proposed scheme. We have explained the Chirikov standard map and the RIPEMD-160 hash function, which are main components of our proposed algorithm.

3.1 Chirikov Standard Map

The Chirikov Standard Map is a simple mathematical model used in the study of chaotic dynamical systems [11]. It is a two-dimensional area-preserving chaotic map. It describes how the position and velocity of this particle change over time under specific conditions. The following iterative Equations 1 and 2 define the map:

$$\theta(n+1) = (\theta(n) + p(n+1)) \mod 2\pi \tag{1}$$

$$p(n+1) = (p(n) + K \cdot \sin(\theta(n))) \mod 2\pi \tag{2}$$

Where $\theta(n)$ represents the angle, p(n) is an angular momentum at the nth iteration, and $\theta(n+1)$ and $p(n+1)$ are the updated position and momentum of the particle, respectively. K is a control parameter determining the degree of chaos in the system. When K is small, the Chirikov Standard Map shows regular behavior. However, the K increases beyond a critical value, and the map starts showing chaotic behavior. We can see that even small changes in the beginning can have huge effects on the results in the future.

The Chirikov Standard Map has several beneficial characteristics that make it suitable for cryptographic applications. Some of the characteristics include ergodicity, sensitivity to initial conditions, and mixed-phase space. Ergodicity ensures that the map fully traverses the phase space and offers great unpredictability. The sensitivity to starting conditions, also known as the "butterfly effect," states that even small changes in the initial values of *theta* and p lead to drastically different trajectories. Lastly, the mixed-phase space attribute describes the presence of both regular and chaotic zones, making it challenging to forecast the system's behaviour[14].

3.2 RACE Integrity Primitives Evaluation Message Digest (RIPEMD-160)

A popular cryptographic hash algorithm called RIPEMD-160 takes input data and outputs a 160-bit hash result [10][6]. It was created as a substitute for the MD4 and MD5 hash algorithms in order to overcome their security flaws. For a variety of applications, it has been discovered that the hash function offers a fair level of security and performance.

The RIPEMD-160 algorithm processes input data in 512-bit blocks, iteratively applying compression functions to update an internal state. The internal state comprises five 32-bit words combined to produce the final 160-bit hash value. While the compression functions work, they use a mix of logical operations like AND, OR, XOR, and NOT, as well as modular math and bitwise rotations. These operations provide nonlinearity, dispersal, and chaos, which are essential for a secure hash function.

4 Proposed Scheme of ChaoticRIPE

In this section, we have discussed our proposed ChaoticRIPE method, which uses the Chirikov Standard Map to improve the existing RIPEMD-160 hash function. Two main components of our proposed Algorithm 1 are secure_ripemd160 and chirikov_standard_map. In this, we first calculate the starting angle theta, the initial angular momentum p, the control parameter K, and the number of iterations to be carried out are the four inputs that the chirikov_standard_map function requires. Theta and p are input values, and the function iteratively applies the Chirikov Standard Map equations to them for the set number of iterations. The final value of theta and p are sent back as output. The essential component of our proposed ChaoticRIPE method is the secure_ripemd160 function. Three parameters are required to calculate the hash: the input data, the control parameter K, and the number of iterations of the Chirikov Standard Map.

The function first calculates the SHA3-512 hash of the supplied data. The resultant hash digest is divided into two pieces in order to determine the starting values of theta and p and is transformed into floating-point integers in the $[0, 2\pi)$ range. Further, the theta, p, K, and iterations are called together with the starting values of the chirikov_standard_map function. In this stage, the hash computation incorporates the chaotic dynamics of the Chirikov Standard Map. The original byte sequences in the RIPEMD-160 hash digest are then replaced with the new values of theta and p, which are subsequently transformed back to byte sequences.

In order to get the improved hash value, the altered digest is finally hashed using the RIPEMD-160 method. A more secure and unpredictable hash function is produced due to the second layer of hashing, which ensures that any possible patterns or correlations generated by the Chirikov Standard Map are scattered. Figure 1 shows the workflow of our proposed ChaoticRIPE method.

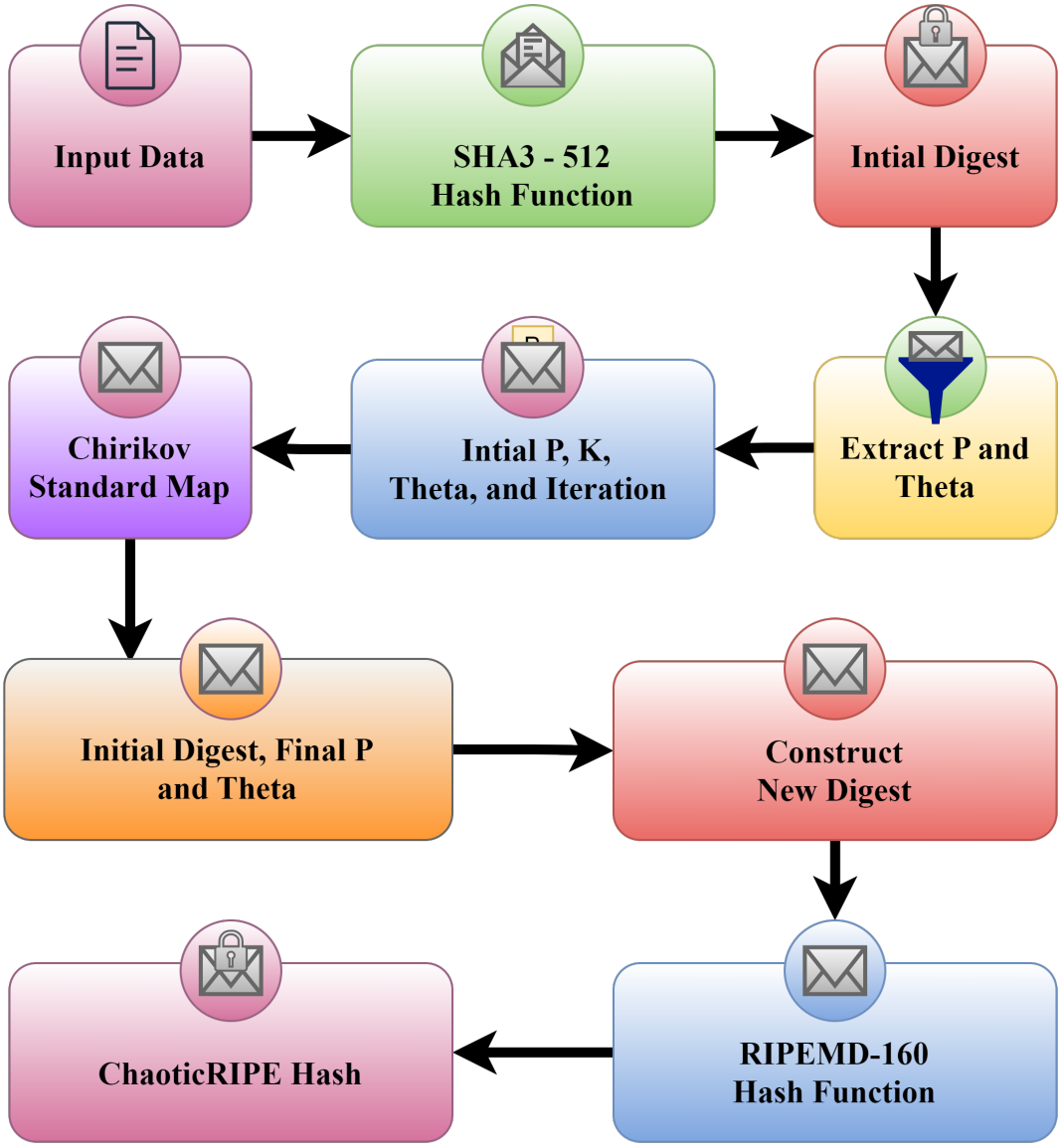

Fig. 1. The workflow of our proposed model ChaoticRIPE

5 Analysis

The sensitivity to small changes in the input data of the original RIPEMD-160 and the proposed ChaoticRIPE are evaluated. We have used the Hamming distance for comparison between the hash values of an original data and the same data with little alteration, as shown in Figure 2. Based on our test, the average Hamming distance was 36.99 for the original RIPEMD-160's whereas the proposed ChaoticRIPE's has an average hamming distance of 37.27. The average Hamming distance is more when the hash function is more sensitive to small changes. This shows that the proposed ChaoticRIPE is more sensitive to input changes as compared to original RIPEMD-160.

Algorithm 1: Algorithm for ChaoticRIPE

Input: data, K, iterations
Output: Hash
// data is any input string
// K is initial value for Chirikov Standard Map
Function chirikov_standard_map(θ, p, K, *iterations*):
  for $i = 1$ *to iterations* **do**
    $p = (p + K \cdot \sin(\theta)) \mod (2\pi)$
    $\theta = (\theta + p) \mod (2\pi)$
  end
  return θ, p
End Function
Function secure_ripemd160(*data*, K, *iterations*):
  m = hashlib.new('sha3_512')
  m.update(data.encode('utf-8'))
  digest = m.digest()
  θ = int.from_bytes(digest[:8], 'big') / $(2^{64}) * 2 * math.pi$ p= $int.from\_bytes(digest[8:16], 'big')/(2^{64}) * 2 * math.pi$
  θ, p = chirikov_standard_map(θ, p, K, iterations)
  θ_bytes = int(θ / (2 * math.pi) * $(2^{64})).to\_bytes(8, 'big')$ p_bytes= $int(p/(2 * math.pi) * (2^{64})).to\_bytes(8, 'big')$
  m = hashlib.new('ripemd160')
  m.update(digest[:8] + θ_bytes + digest[16:24] + p_bytes + digest[24:])
  return m.hexdigest()
End Function

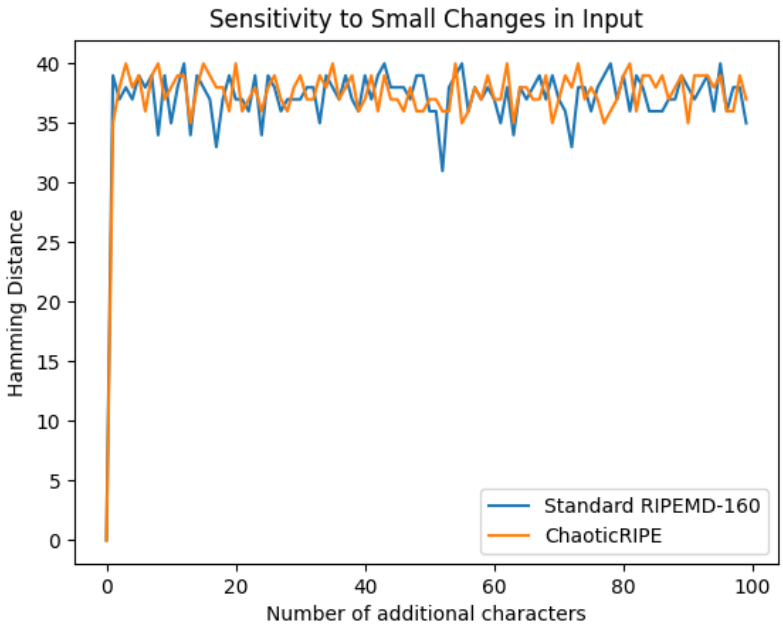

Fig. 2. Hamming Distance Comparison

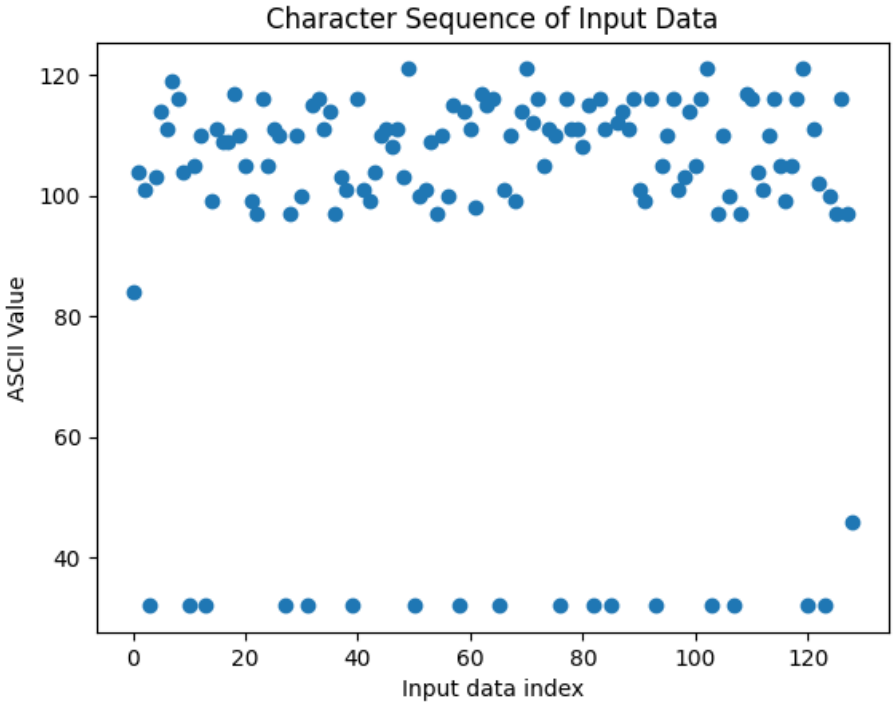

Fig. 3. Character Sequence of Input Data

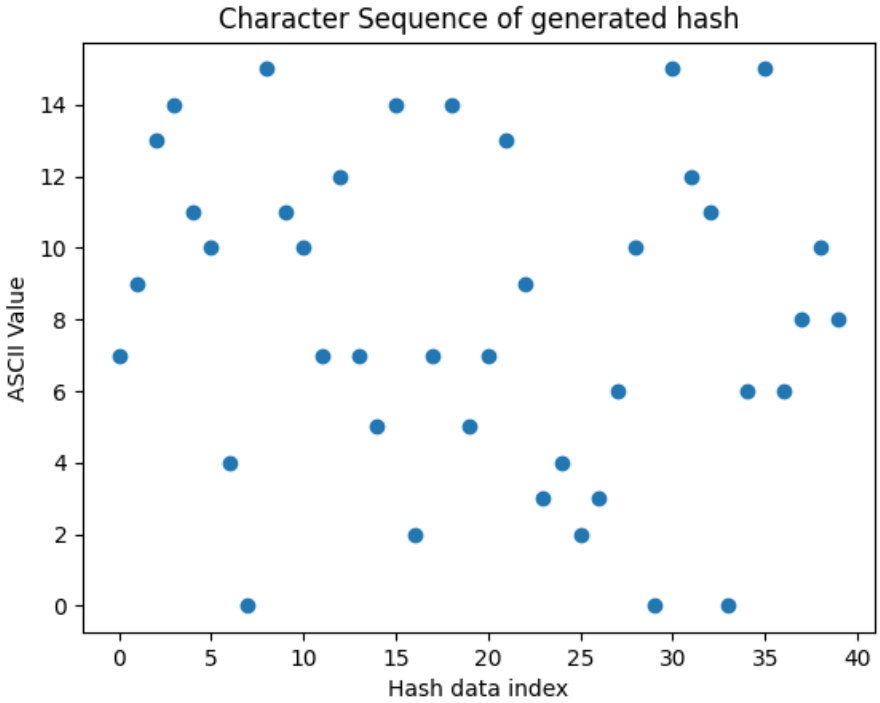

Fig. 4. Character Sequence of Generated Hash

The key characteristic of a hashing algorithm is the uniform distribution of hexadecimal hash values. To demonstrate this, we generate a message using random characters, convert it to ASCII decimal codes, and plot it in Figure 3. ASCII code of the hash value generated by our proposed technique are then uniformly distributed across the possible hash value range shown in Figure 4. This ensures that the hash distribution is sufficiently uniform to hide information and serve as a robust security measure.

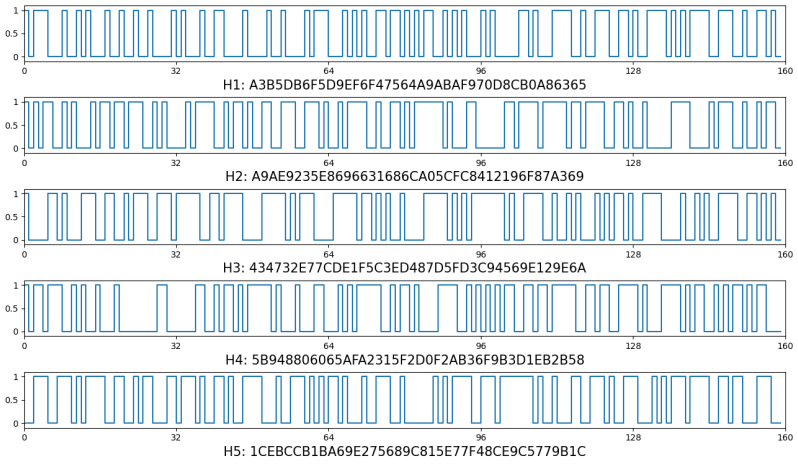

Fig. 5. Hash Value of Proposed Scheme with Different Conditions

The proposed ChaoticRIPE method is very sensitive to small changes in the starting conditions or the messages. The simulation for sensitivity is run under the conditions listed below.
Condition 1: The original message is: "Central Class is a listing of online courses"
Condition 2: Replace the first character of the original message "C" by "c".
Condition 3: Add "." at the last of the original message.
Condition 4: change K = 0.711 to 0.710
Condition 5: change K = 0.711 to 0.710 and Replace the first character of the original message "C" by "c".

The corresponding 160-bit hash values in hexadecimal format are the following:
Condition 1: A3B5DB6F5D9EF6F47564A9ABAF970D8CB0A86365
Condition 2: A9AE9235E8696631686CA05CFC8412196F87A369
Condition 3: 434732E77CDE1F5C3ED487D5FD3C94569E129E6A
Condition 4: 5B948806065AFA2315F2D0F2AB36F9B3D1EB2B58
Condition 5: 1CEBCCB1BA69E275689C815E77F48CE9C5779B1C

In Figure 5, each hash value for the proposed system under various situations. We may state that the suggested method has considerable sensitivity since the results of the binary representations of the hash show that a slight change in the message, starting condition, or control parameter can affect all the hash values.

Based on the results summarised in Table 1, it can be observed that the proposed ChaoticRIPE passed all NIST 2.1.2 tests with outstanding P-values. Table 2 shows the position-wise frequencies of 0s and 1s for 1 million rows containing hash values of different input messages. Hence the proposed scheme is complex enough.

The time-dependent efficiency of the two hash algorithms was compared by calculating the average time required to calculate the hash of the data. According to the experiment's findings shown in Figure 6, original RIPEMD-160 took an average of 0.0026 milliseconds,

whereas proposed ChaoticRIPE took an average of 0.0529 milliseconds. ChaoticRIPE requires more time than Standard RIPEMD-160. However, the difference is minimal (around 0.05 milliseconds). This minor increase in calculation time is a fair trade-off for enhanced sensitivity to input changes since security activities like hashing are often not time-critical.

Table 1. Results of NIST Test Suit 2.1.2

P-VALUE	PROPORTION	STATISTICAL TEST	Result
0.460664	156/160	Frequency	PASS
0.788728	160/160	BlockFrequency	PASS
0.546791	156/160	CumulativeSums	PASS
0.220448	155/160	CumulativeSums	PASS
0.425817	160/160	Runs	PASS
0.350485	156/160	LongestRun	PASS
0.275709	158/160	Rank	PASS
0.180322	159/160	FFT	PASS
0.934318	160/160	NonOverlappingTemplate	PASS
0.141256	159/160	Universal	PASS
0.559523	157/160	ApproximateEntropy	PASS
0.917870	102/103	RandomExcursions	PASS
0.995373	103/103	RandomExcursionsVariant	PASS
0.919445	158/160	Serial	PASS
0.739918	156/160	LinearComplexity	PASS

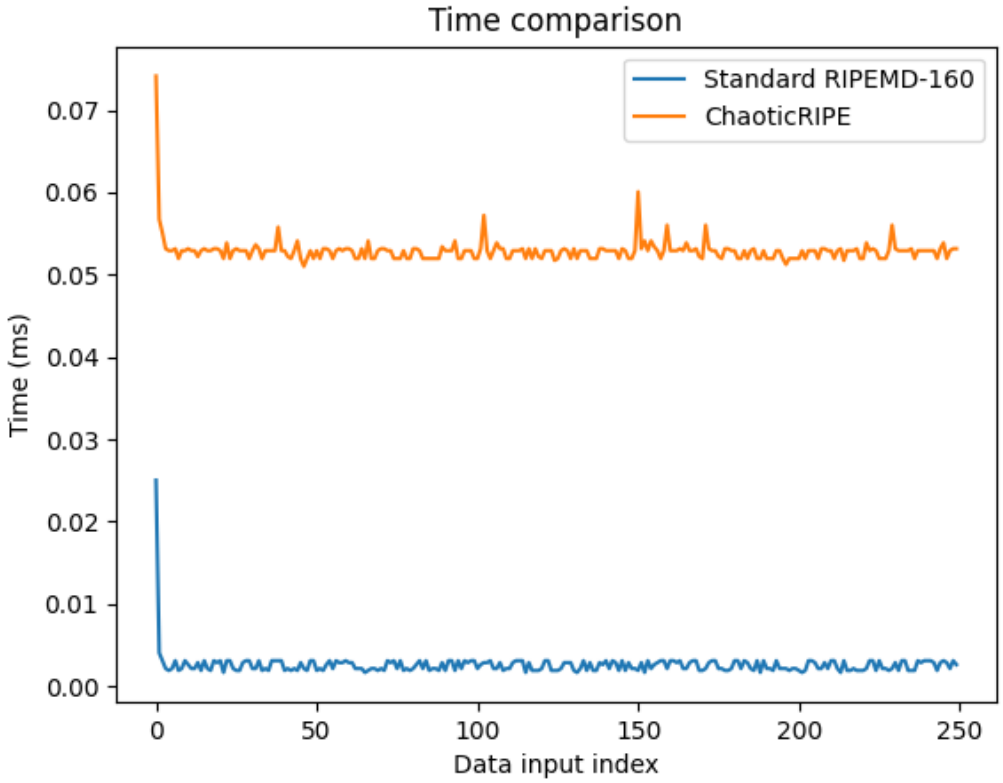

Fig. 6. Time Comparison

Table 2. Frequency of 0s and 1s in the sequence of 1M BITSREAD

P.	0s	1s	P.	0s	1s	P.	0s	1s	P.	0s	1s
1	500664	499336	41	500270	499730	81	500904	499096	121	500516	499484
2	500006	499994	42	499984	500016	82	500683	499317	122	499969	500031
3	500054	499946	43	499658	500342	83	499489	500511	123	500648	499352
4	500171	499829	44	499426	500574	84	499947	500053	124	500594	499406
5	500692	499308	45	500686	499314	85	500501	499499	125	499760	500240
6	499743	500257	46	500502	499498	86	498299	501701	126	500748	499252
7	499460	500540	47	499708	500292	87	500523	499477	127	499754	500246
8	499204	500796	48	500877	499123	88	499739	500261	128	499623	500377
9	500188	499812	49	500354	499646	89	500302	499698	129	500929	499071
10	500184	499816	50	499654	500346	90	499272	500728	130	500293	499707
11	500744	499256	51	500315	499685	91	499813	500187	131	499494	500506
12	499551	500449	52	498944	501056	92	500521	499479	132	500317	499683
13	500266	499734	53	500165	499835	93	499837	500163	133	498501	501499
14	500474	499526	54	499803	500197	94	500608	499392	134	500217	499783
15	499391	500609	55	499782	500218	95	499380	500620	135	499957	500043
16	499871	500129	56	499999	500001	96	498610	501390	136	499709	500291
17	499092	500908	57	499483	500517	97	499249	500751	137	500494	499506
18	500088	499912	58	500090	499910	98	499050	500950	138	500748	499252
19	500781	499219	59	500314	499686	99	500295	499705	139	500166	499834
20	500404	499596	60	500210	499790	100	500176	499824	140	499707	500293
21	499698	500302	61	499238	500762	101	500171	499829	141	500057	499943
22	500563	499437	62	500777	499223	102	499320	500680	142	499845	500155
23	499657	500343	63	499714	500286	103	499286	500714	143	499071	500929
24	500268	499732	64	499477	500523	104	499838	500162	144	499879	500121
25	500288	499712	65	499891	500109	105	500225	499775	145	499876	500124
26	500818	499182	66	499639	500361	106	500832	499168	146	499728	500272
27	499308	500692	67	500292	499708	107	501020	498980	147	499664	500336
28	499824	500176	68	500239	499761	108	500376	499624	148	500125	499875
29	500418	499582	69	500305	499695	109	499829	500171	149	499058	500942
30	499955	500045	70	499931	500069	110	500683	499317	150	499897	500103
31	500118	499882	71	500864	499136	111	500662	499338	151	500567	499433
32	499843	500157	72	500520	499480	112	499294	500706	152	500524	499476
33	499738	500262	73	500737	499263	113	499369	500631	153	498964	501036
34	500979	499021	74	500032	499968	114	499904	500096	154	498616	501384
35	499945	500055	75	499851	500149	115	500198	499802	155	499776	500224
36	500417	499583	76	500384	499616	116	499282	500718	156	500405	499595
37	500391	499609	77	499859	500141	117	500122	499878	157	499931	500069
38	500479	499521	78	499993	500007	118	499943	500057	158	500045	499955
39	500074	499926	79	500412	499588	119	499169	500831	159	501022	498978
40	500280	499720	80	499785	500215	120	499707	500293	160	499913	500087

6 Conclusion & Future Work

In this paper, we have proposed ChaoticRIPE, which is an improved version of RIPEMD-160 hash function. It uses the chaotic characteristics of the Chirikov Standard Map to increase its security. The ergodicity, sensitivity to initial conditions, and mixed-phase space of the Chirikov Standard Map contributes to improved security the hash function by making it unpredictable and resistance to various attacks. According to our results, the ChaoticRIPE is more sensitive to input changes.

Further research might investigate how other chaotic maps, including the Henon Map, might be integrated to evaluate their effects on security and performance. Furthermore, maximising ChaoticRIPE for hardware implementations could help resource-limited settings be more efficient. Its use in cryptographic systems like digital signatures and key

derivation mechanisms should also be explored in further research. ChaoticRIPE's potential use in post-quantum cryptography makes evaluating its resistance to quantum cryptanalysis rather important. Finally, thorough real-world security testing may confirm its resilience against sophisticated cryptographic attacks.

References

1. Adeniyi, E.A., Falola, P.B., Maashi, M.S., Aljebreen, M., Bharany, S.: Secure sensitive data sharing using rsa and elgamal cryptographic algorithms with hash functions. Information **13**(10), 442 (2022)
2. Al-Zubaidie, M.: Implication of lightweight and robust hash function to support key exchange in health sensor networks. Symmetry **15**(1), 152 (2023)
3. Alnajim, A.M., Abou-Bakr, E., Alruwisan, S.S., Khan, S., Elmanfaloty, R.A.: Hybrid chaotic-based prng for secure cryptography applications. Applied Sciences **13**(13), 7768 (2023)
4. Ayubi, P., Setayeshi, S., Rahmani, A.M.: Chaotic complex hashing: A simple chaotic keyed hash function based on complex quadratic map. Chaos, Solitons & Fractals **173**, 113647 (2023)
5. Ayubi, P., Setayeshi, S., Rahmani, A.M.: Chaotic complex hashing: A simple chaotic keyed hash function based on complex quadratic map. Chaos, Solitons & Fractals **173**, 113647 (2023)
6. Bosselaers, A.: Ripemd family. In: Encyclopedia of Cryptography, Security and Privacy, pp. 2115–2119. Springer (2025)
7. Bouvier, C., Briaud, P., Chaidos, P., Perrin, L., Salen, R., Velichkov, V., Willems, D.: New design techniques for efficient arithmetization-oriented hash functions: anemoi permutations and jive compression mode. In: Annual International Cryptology Conference. pp. 507–539 (2023)
8. Buchmann, J.: Introduction to cryptography, vol. 335. Springer (2004)
9. Dai, W., Li, B., Du, Q., Zhu, Z., Liu, A.: Chaos-based index-of-min hashing scheme for cancellable biometrics security. IEEE Transactions on Information Forensics and Security (2024)
10. Dobbertin, H., Bosselaers, A., Preneel, B.: Ripemd-160: A strengthened version of ripemd. In: International Workshop on Fast Software Encryption. pp. 71–82. Springer (1996)
11. Dong, Y., Zhao, G., Chi, K.T., Ma, Y., Ning, H.: A novel spatiotemporal chaotic model with provable security (2023)
12. Gambhir, G., Mandal, J.K., Gambhir, M.: Parallel chaos hash based lsb steganography technique using logistic map. In: 2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC). pp. 308–312. IEEE (2022)
13. Grassi, L., Khovratovich, D., Schofnegger, M.: Poseidon2: A faster version of the poseidon hash function. Cryptology ePrint Archive (2023)
14. Karimov, A.I., Butusov, D.N., Rybin, V.G., Karimov, T.I.: The study of the modified chirikov map. In: 2017 XX IEEE International Conference on Soft Computing and Measurements (SCM). pp. 341–344 (2017). https://doi.org/10.1109/SCM.2017.7970579
15. Li, L.: A novel chaotic map application in image encryption algorithm. Expert Systems with Applications p. 124316 (2024)
16. Li, Y., Liu, F., Wang, G.: New records in collision attacks on ripemd-160 and sha-256. Cryptology ePrint Archive (2023)
17. Liu, F., Wang, G., Sarkar, S., Anand, R., Meier, W., Li, Y., Isobe, T.: Analysis of ripemd-160: New collision attacks and finding characteristics with milp. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 189–219. Springer (2023)
18. Liu, J., Liu, Y., Li, B.: Design and analysis of hash function based on spark and chaos system. International Journal of Network Security **25**(3), 456–467 (2023)
19. Liu, Y.J., Liu, J.D., Zhong, M., Li, B., Xu, H.Q.: Design and analysis of hash function based on two-dimensional integer chaotic map. Journal of Computers **33**(3), 85–97 (2022)
20. Mouha, N.: Exploring formal methods for cryptographic hash function implementations. In: Australasian Conference on Information Security and Privacy. pp. 177–195. Springer (2023)
21. Omranpour, H., Mohammadi Ledari, Z., Taheri, M.: Presentation of encryption method for rgb images based on an evolutionary algorithm using chaos functions and hash tables. Multimedia Tools and Applications **82**(6), 9343–9360 (2023)
22. Pibiri, G.E., Trani, R.: Parallel and external-memory construction of minimal perfect hash functions with pthash. IEEE Transactions on Knowledge and Data Engineering (2023)
23. Rasool, M., Belhaouari, S.B.: From collatz conjecture to chaos and hash function. Chaos, Solitons & Fractals **176**, 114103 (2023)
24. SeragEldin, S.M., El-Latif, A.A.A., Chelloug, S.A., Ahmad, M., Eldeeb, A.H., Diab, T.O., Al Sobky, W.I., Zaky, H.N.: Design and analysis of new version of cryptographic hash function based on improved chaotic maps with induced dna sequences. IEEE Access (2023)

25. Shi, H., Yan, D., Wang, L., Duan, S.: A novel memristor-based chaotic image encryption algorithm with hash process and s-box. The European Physical Journal Special Topics **231**(3), 465–480 (2022)
26. Sideris, A., Sanida, T., Dasygenis, M.: A novel hardware architecture for enhancing the keccak hash function in fpga devices. Information **14**(9), 475 (2023)
27. Suhaili, S.B., Watanabe, T., Julai, N.: Throughput improvement of ripemd-160 design using unfolding transformation technique. Journal of Optimization in Industrial Engineering **15**(1), 207–216 (2022)
28. Szepieniec, A., Lemmens, A., Sauer, J.F., Threadbare, B., et al.: The tip5 hash function for recursive starks. Cryptology ePrint Archive (2023)
29. Tessaro, S., Zhu, C.: Threshold and multi-signature schemes from linear hash functions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 628–658. Springer (2023)
30. Windarta, S., Suryadi, S., Ramli, K., Pranggono, B., Gunawan, T.S.: Lightweight cryptographic hash functions: design trends, comparative study, and future directions. IEEE Access **10**, 82272–82294 (2022)
31. Wu, Q.: A chaos-based hash function. In: 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. pp. 1–4 (2015). https://doi.org/10.1109/CyberC.2015.13
32. Zellagui, A., Hadj-Said, N., Ali-Pacha, A.: Secure md4 hash function using henon. Malaysian Journal of Computing and Applied Mathematics **3**(2), 73–80 (2020)
33. Zellagui, A., Hadj-Said, N., Ali-Pacha, A.: A new hash function inspired by sponge construction using chaotic maps. Journal of Discrete Mathematical Sciences and Cryptography pp. 1–31 (2022)

Authors

Suparn Padma Patra earned his M.Sc. in Computer Science with a specialization in Artificial Intelligence from the Central University of Rajasthan, India, where he was awarded the Gold Medal. He is currently pursuing a Ph.D. in Computer Science at the same institution. His research interests include Cryptographic Hash Functions, Chaos Theory, MOOCs, and Online Education.

Mamta Rani is a Professor in the Department of Computer Science and the Dean of the School of Mathematics, Statistics, and Computational Science at the Central University of Rajasthan, India. She holds an M.C.A., Ph.D., and D.Sc., with research interests spanning Fractal Graphics, Chaos Theory, Swarm Intelligence, Chaotic Cryptography, and Vedic Sciences. She has made significant contributions to the field of computer science, particularly in the study of fractals and chaos theory.