JOURNAL OF ENGINEERING AND COMPUTATIONAL INTELLIGENCE REVIEW (JECIR)
ISSN (Online): 3105-4811 ISSN (Print): 3106-969X
Volume 4, Issue 1, 2026
https://jecir.com/

PERFORMANCE ANALYSIS OF HASH FUNCTIONS: A COMPREHENSIVE STUDY OF AVERAGE SEARCH LENGTH AND DISTRIBUTION PROPERTIES

Aliza Ashfaq [1], Roheena Nayab [2], Tayyaba Rehman [3]

Affiliations:

[1] Mathematical Sciences,
Fatima Jinnah Women University,
Rawalpindi, Pakistan
Email: alizaashfaq44@gmail.com

[2] Mathematical sciences,
COMSATS University, Pakistan
Email: roheenanayab5@gmail.com

[3] Mathematical Sciences,
Fatima Jinnah Women University,
Rawalpindi, Pakistan
Email: rtayyaba78@gmail.com

Corresponding Author's Email

[1] alizaashfaq44@gmail.com

Abstract

This study examines the performance analysis of hash functions, with a basic focus on examining the average search length of keys using different hash functions including division remainder method, universal hashing, and multiplication method. Hashing functions from H1 class of universal hashing defined by Wegman and Carter are used to evaluate the effect of prime numbers. It was found that the choice of prime number influences the effectiveness of universal hashing functions—the smaller the prime number, the better its performance. The study also compares the performance of universal hashing against radix transformation and division remainder method, revealing that universal hashing can excel under specific conditions. The analysis includes comprehensive subsets of the Universe, examining all possible subsets to understand how hash functions distribute data properly. Results indicate that the division-remainder method performs best in terms of distributing subsets with fewer collisions, resulting in smaller average successful search lengths. This method also demonstrated the highest percentage of perfect distributions across all subset sizes. The study concludes with insights into the effect of different parameters on hash performance, providing a valuable resource for understanding the significance of hash functions in cryptographic applications.

Keywords: Hash Functions, Universal Hashing, Average Search Length, Division Remainder Method, Multiplication Method, Radix Transformation, Prime Number Effect, Performance Analysis, H1 Class, Collision Resolution

I. INTRODUCTION

A. Background and Motivation

The word cryptography comes from the Greek phrases *kryptos* (hidden) and *graphein* (writing), altogether meaning "hidden writing." Cryptography is the approach of secret message transmission between individuals or organizations to protect the secrecy of information from unauthorized parties. From the beginning of civilization, people developed the need for secure and secret communication, leading to the evolution of cryptographic methods [1].

The fundamental idea of a cryptographic tool is to cipher information to gain confidentiality of data in a manner that unauthorized parties cannot derive its meaning. Two of the most common uses of cryptography

JOURNAL OF ENGINEERING AND COMPUTATIONAL INTELLIGENCE REVIEW (JECIR)
ISSN (Online): 3105-4811 ISSN (Print): 3106-969X
Volume 4, Issue 1, 2026
https://jecir.com/

include transmitting data via insecure channels (such as the internet) and ensuring unauthorized individuals cannot understand accessed data.

FIGURE 1
CLASSIFICATION OF CRYPTOLOGY

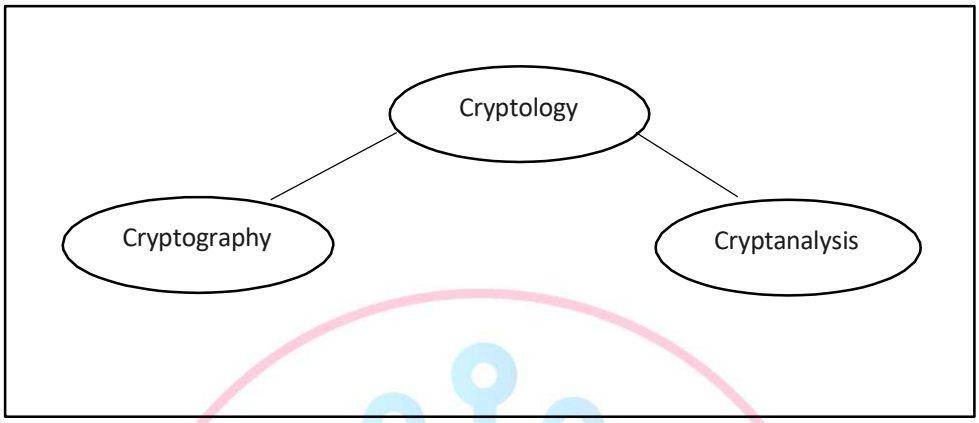
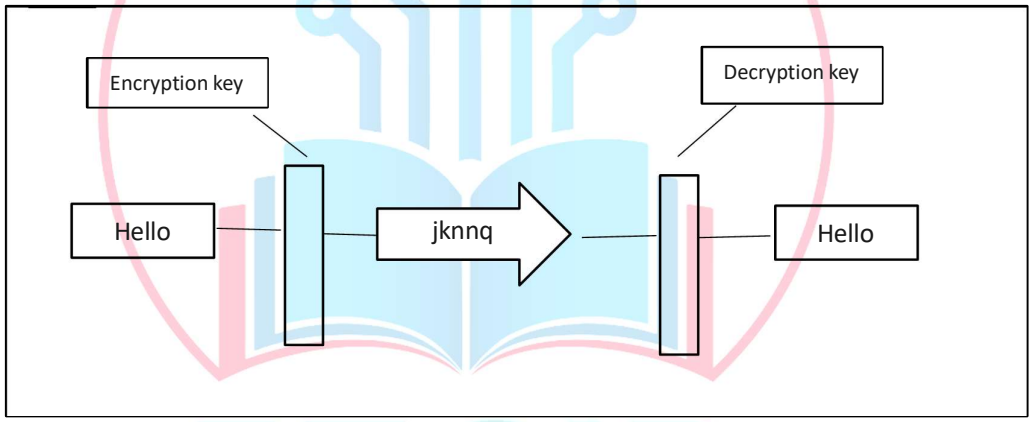


FIGURE 2
PROCESS OF CRYPTOGRAPHY



B. Classification of Cryptographic Algorithms

Cryptographic algorithms are categorized as follows:

1. **Secret key cryptography (Symmetric encryption)**: Uses a single key for both encryption and decryption, generally used for privacy and confidentiality.
2. **Public key cryptography (Asymmetric encryption)**: Uses one key for encryption and another for decryption, used for authentication and key exchange.
3. **Hash functions**: Use mathematical transformation to irreversibly encrypt data, providing a digital fingerprint.

The evaluation of computational methods is important in modern engineering and information systems, where algorithm effectiveness depends on efficient information processing and pattern identification [22]. Similar considerations of computational effectiveness motivate the performance comparison of hashing techniques in the present study.

C. Classical and Modern Cryptography

In Classical Cryptography, two main types exist:

- **Transposition method**: Scrambles letters of plaintext to occupy different positions
- **Substitution method**: Replaces letters with other letters to produce ciphertext

JOURNAL OF ENGINEERING AND COMPUTATIONAL INTELLIGENCE REVIEW (JECIR)
ISSN (Online): 3105-4811 ISSN (Print): 3106-969X
Volume 4, Issue 1, 2026
https://jecir.com/

In Modern Cryptography, three types exist:

- **Stream ciphers**: Each bit is encrypted separately (synchronous and asynchronous)
- **Public key encryption**: Allows key distribution on public channels
- **Hash functions**: One-way cryptographic algorithms

D. Hash Functions Overview

A hash function is a one-way cryptographic algorithm that converts an input of any length into a unique fixed-length output called a hash digest, hash value, or hash code. Hash functions are constructed upon factors including keys, hash tables, and costs. A hash function works by mapping a large number of integers to a smaller integer that becomes the index for that array of numbers within the hash table [2].

Fundamental Properties of Hash Functions:

1. **Collision resistance**: It is difficult to find two different inputs that produce the same hash value
2. **Pre-image resistance**: It is difficult to find an input that maps to a specific hash value
3. **Second pre-image resistance**: It is difficult to find a second input that has the same output as a given input

If a hash function maps every item to a precise location, it is called a perfect hash function. In practice, it is difficult to assign precise slots for each data item. A proper hash function should have minimum collisions and be clean and quick to compute [3].

E. Common Hash Functions

- **MD Series (MD2, MD4, MD5, MD6)**: Designed by Ronald Rivest in 1991 for RSA security [7]
- **SHA (Secure Hash Algorithm)**: SHA-0 published in 1993, operates on a 512-bit message block divided into 32-bit sizes [8]
- **RIPEMD**: Family of hash functions based on MD4 design concepts developed by Hans Dobbertin [9]
- **Whirlpool**: Based on Advanced Encryption Standard, designed by Vincent Rijmen in 2000 [10]
- **BLAKE**: Submitted to NIST hash function competition by Jean Philippe, announced in 2012 [11]
- **SHA-3**: Released by NIST in 2015
- **BLAKE3**: Improved version of BLAKE2, introduced on January 9, 2020

F. Hashing Techniques

1. **Division Remainder**: Computes hash value from key by dividing the key by table size
2. **Folding Method**: Uses addition to add numbers together and store the key
3. **Radix Transformation**: Transforms a key into another number base to obtain the hash value
4. **Mid-Square**: Squares the key and takes the middle part as the hash value [5]

G. Research Objectives

The main purpose of this research is to examine the average search length of keys using different methods. Recent research highlights the role of artificial intelligence and advanced business analytics in improving analytical and operational decision-making [10]. In a similar analytical context, the present study evaluates different hashing techniques by examining their performance through measurable indicators such as average successful search length.

For this purpose, we consider exhaustive subsets of the Universe, analysing all possible subsets to understand how hashing functions distribute data properly. Based on calculations of average successful length, we demonstrate the performance of hash functions, distinguishing between perfect and poor performance, and examine the effect of prime numbers on hash function performance.

II. LITERATURE REVIEW

A. Previous Research on Hash Functions

Rajeshwaran and Kumar [14] suggested a hash algorithm based on cellular automata, outlining a strong cryptographic hash function by spreading hash code generated using the proposed algorithm. Two main properties of the hash function were tested, and while stream features and collision probability were strengthened, additional datasets could provide more explanatory power.

JOURNAL OF ENGINEERING AND COMPUTATIONAL INTELLIGENCE REVIEW (JECIR)
ISSN (Online): 3105-4811 ISSN (Print): 3106-969X
Volume 4, Issue 1, 2026
https://jecir.com/

Kheshaifaty and Gutub [15] proposed a complex system integrating CAPTCHA, cryptography, and hash functions for study verification. After verifying CAPTCHA input, the password was encrypted and hashed using the AES algorithm. For login verification, the hash code of the password must match the code stored in the system.

Ali and Farhan [16] proposed an algorithm strengthening the MD5 algorithm against attacks by actively extending the output of MD5, preventing various attacks due to the small output range of 128-2,096 bits. The algorithm uses a key and the preliminary permutation (IP) table of DES to develop a random matrix.

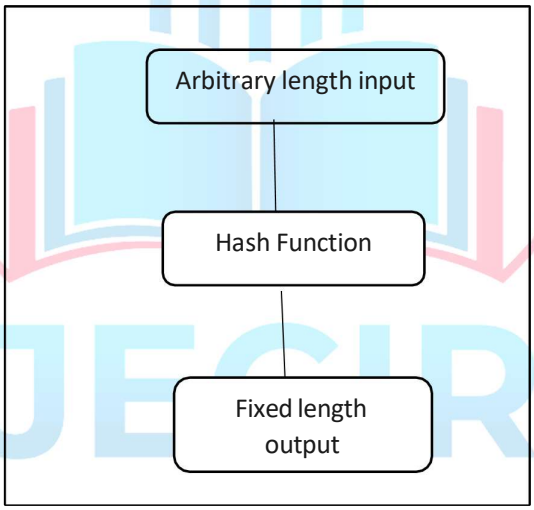
William et al. [17] aimed to increase encryption and decryption capability using a composite algorithm comprising AES, ECC, and SHA-256 algorithms. In the proposed method, messages were encrypted using a key from the ECC algorithm within the AES algorithm, then prepared as ciphertext using SHA-256.

B. Development of Cryptographic Hash Functions

Diffie and Hellman [18] are credited with developing public key cryptography in the mid-1970s. Their seminal paper "New Directions in Cryptography" introduced concepts like Digital Signatures and differentiated Confidentiality from Authentication, leading to the development of cryptographic hash functions.

Bert Rompay [19] extracted examples from the NESSIE (New European Scheme for Signature Integrity and Encryption) project to illustrate how cryptographic hash functions were evaluated. The NESSIE task introduced seventeen block ciphers and six stream ciphers, with submitted unkeyed and keyed hash functions (MACs). Rompay also discussed the open competition used by NIST to determine the Advanced Encryption Standard, where Rijndael [20] was ultimately selected.

FIGURE 3
HASH FUNCTION



III. HASH FUNCTIONS AND TYPES

A. Definition and Properties

Any function used to control data integrity that matches fixed-size values resulting from algorithmic processing of random-size data is referred to as hashing. A hash function transforms an input into a fixed-length output called a hash value or hash code.

B. Types of Hash Functions

1. **Cryptographic Hash Functions**: Established for security purposes, satisfying properties like collision resistance, pre-image resistance, and second pre-image resistance (e.g., MD5, SHA-1, SHA-2, BLAKE)
2. **Non-Cryptographic Hash Functions**: No security protocol concerns, includes hash tables and data structures (e.g., Murmur hash, City hash)

JOURNAL OF ENGINEERING AND COMPUTATIONAL INTELLIGENCE REVIEW (JECIR)
ISSN (Online): 3105-4811 ISSN (Print): 3106-969X
Volume 4, Issue 1, 2026
https://jecir.com/

3. **Universal Hashing**: Designed to reduce collision probability for any given set, chosen randomly. Includes Carter-Wegman hashing
4. **Perfect Hash Function**: Generates unique hash values for known keys with no collisions

C. Classification of Hash Functions

At the highest level, hash functions can be split into:

1. **Unkeyed hash functions**: Define a free entity (a message)
2. **Keyed hash functions**: Define two inputs—a message and a secret key

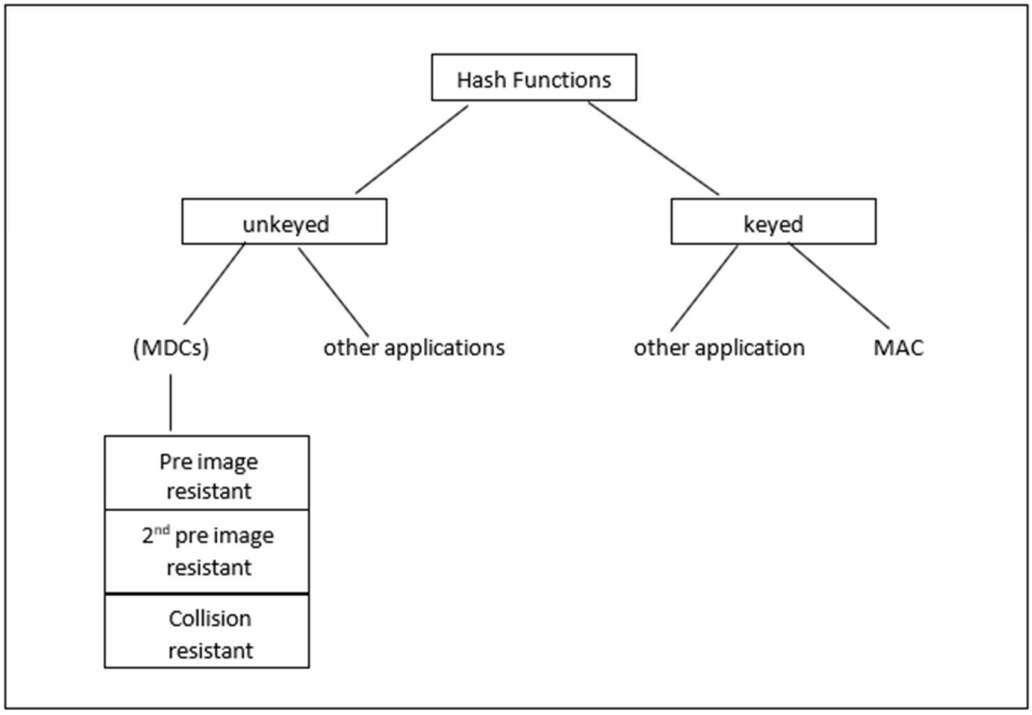
Manipulation Detection Codes (MDCs)

Also known as message integrity codes (MICs). The purpose of an MDC is to provide a representative hash of a message, facilitating data integrity assurances. MDCs are a subclass of unkeyed hash functions [4].

Message Authentication Codes (MACs)

MAC represents a value composed of messages and secret keys, used for authentication of message integrity and source authentication [5].

FIGURE 4
MESSAGE AUTHENTICATION CODES



D. Properties of Hash Functions

1. **Pre-image Resistance**: Computationally impossible to compute input corresponding to a specific output
2. **Second Pre-image Resistance**: Computationally difficult to find a second input generating the same hash
3. **Collision Resistance**: Difficult for an attacker to find messages with identical hash values [6]

E. Hashing Techniques

Division Algorithm

The division method maps keys to hash table indices:

$$H(K) = K \pmod{M}$$

where H signifies the hash function, K indicates the key value, and M denotes the length of the hash table.

Mid-Square Technique

JOURNAL OF ENGINEERING AND COMPUTATIONAL INTELLIGENCE REVIEW (JECIR)
ISSN (Online): 3105-4811 ISSN (Print): 3106-969X
Volume 4, Issue 1, 2026
https://jecir.com/

Squares the key value and takes the middle digits as the hash value:

$$H(K^2) = L$$

Folding Hash Method

Uses addition to add numbers together:

$$H(K) = K_1 + K_2 + K_3 + \ldots + K_n$$

Multiplication Hash Method

Ensures value of A is $0 < A < 1$, multiplies key with A, separates fractional part, multiplies by M: $H(K) = \text{floor}(M \times (kA \bmod 1))$

IV. METHODOLOGY: PERFORMANCE ANALYSIS OF HASH FUNCTIONS

A. Average Successful Search Length

The average successful search length is affected by factors such as dataset size, distribution of elements, and search algorithm authenticity [7]. In a properly-designed cryptographic hash function like SHA-256, the average successful search length for finding a preimage is expected to be close to 2^256.

We analyse hash functions from the H1 class of universal hashing functions, which takes the form:

$$h_{a,p}(x) = ((a \cdot x + b) \bmod p) \bmod m$$

B. Effect of Prime Numbers

To analyse the effect of prime numbers on hash performance, we used the H1 class universal hash function:

$$h(x) = ((cx + d) \bmod p) \bmod m$$

where c and d are chosen randomly, p is a large prime number, and m is the table size.

V. Results and Analysis

A. Performance Analysis with Different Key Sets

Case 1 examined the same hash function with four different key sets:

$$h(x) = (((519x + 703) \bmod 36373) \bmod 13)$$

Key Set K1 = {936, 748, 996, 815, 864, 867, 730, 971}

Slot	0	1	2	3	4	5	6	7	8	9	10	11	12
Keys	-	936	-	815	748 867	971	-	996 864	-	730	-	-	-

Collisions: At slot 4 (748, 867) and slot 7 (996, 864)

Average Search Length: (1+1+1+1+2+2+1+1)/8 = 1.25

Key Set K2 = {772, 841, 738, 907, 876, 932, 713, 816}

Slot	0	1	2	3	4	5	6	7	8	9	10	11	12
Keys	713	738	816	907	841	932	-	772	876	-	-	-	-

Average Search Length: 1 (No collisions)

Key Set K3 = {757, 861, 929, 744, 755, 797, 808, 836}

Slot	8	10
Keys	757, 929, 744, 797, 836	861, 755, 808

Average Search Length: (1+1+2+3+2+4+3+5)/8 = 2.625

Key Set K4 = {715, 952, 899, 754, 860, 913, 992, 781}

Slot	11
Keys	715, 952, 899, 754, 860, 913, 992, 781

Average Search Length: (1+2+3+4+5+6+7+8)/8 = 2.5

TABLE I
AVERAGE SEARCH PERFORMANCE OF h(x) OVER DIFFERENT KEY SETS

Key Set	Average Successful Search Length	Performance
K1	1.25	Good
K2	1.00	Perfect

JOURNAL OF ENGINEERING AND COMPUTATIONAL INTELLIGENCE REVIEW (JECIR)
ISSN (Online): 3105-4811 ISSN (Print): 3106-969X
Volume 4, Issue 1, 2026
https://jecir.com/

K3	2.625	Average
K4	2.50	Worst

TABLE II
SEARCH PERFORMANCE OF DIFFERENT h(x) OVER K

Hashing Function	Average Search Length	Performance
$h_1(x)$	1.00	Perfect
$h_2(x)$	1.625	Good
$h_3(x)$	3.00	Bad
$h_4(x)$	4.50	Worst

B. Effect of Prime Numbers on Hash Function Performance

Using key set K = {737, 945, 763, 815, 867, 841, 893, 789, 919, 997} with c = 453 and d = 657:

h(x) = ((453x + 657) mod p) mod 13

TABLE III
PERFORMANCE OF h(x) BY VARYING p

p Value	Overflow Records	Average Search Length	(cx+d)/p	Search Performance
226,637	9	5.5	1.47	Worse
170,689	8	4.6	1.95	Bad
156,967	9	5.5	1.1	Worse
128,761	8	3.0	2.59	Average
10,687	0	1.0	31.3	Perfect
7,477	2	1.2	44.7	Good

The analysis reveals that when the ratio (cx+d)/p decreases, the number of collisions increases. When the prime number is too large, all numbers of experience collisions.

C. Comparative Analysis of Hashing Techniques

TABLE IV
COMPARISON ON BASIS OF AVERAGE LENGTH

Subset Size (r)	3	4	5	6	7	8	9	10
No. of Subsets (nCr)	1,140	4,845	15,504	38,760	77,520	125,970	167,960	184,756
Mid-Square Method	1.12	1.19	1.25	1.31	1.38	1.44	1.50	1.57
Modulo Method	1.05	1.08	1.10	1.13	1.16	1.18	1.21	1.23
Multiplication Method	1.13	1.19	1.25	1.31	1.38	1.44	1.50	1.57
Decimal to Base 11	1.06	1.09	1.12	1.16	1.19	1.22	1.25	1.28
Universal Hashing	1.07	1.11	1.15	1.18	1.22	1.26	1.29	1.33

The comparison shows that the average search performance of the division remainder method is superior to universal hashing. As the subset size changes from 3 to 10, the average successful search length is calculated for all subsets for each hash function.

TABLE V
COMPARISON ON BASIS OF PERFECT DISTRIBUTION (PERCENTAGE)

Subset Size (r)	3	4	5	6	7	8	9	10
Mid-Square Method	65.96	42.13	21.96	8.79	2.41	0.34	0	0
Modulo Method	84.20	69.35	52.01	34.67	19.81	9.15	3.05	0.55
Multiplication Method	67.63	50.09	33.69	20.52	10.79	4.65	1.46	0.25

JOURNAL OF ENGINEERING AND COMPUTATIONAL INTELLIGENCE REVIEW (JECIR)
ISSN (Online): 3105-4811 ISSN (Print): 3106-969X
Volume 4, Issue 1, 2026
https://jecir.com/

Decimal to Base 11	81.40	64.75	46.31	29.05	15.40	6.49	1.94	0.31
Decimal to Base 15	59.65	36.12	18.78	8.31	3.04	0.88	0.18	0.02
Universal Hashing	78.77	60.56	41.49	24.75	12.43	4.96	1.41	0.22

TABLE VI
COMPARISON ON BASIS OF WORST DISTRIBUTION (PERCENTAGE)

Subset Size (r)	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Mid-Square Method	1.93	0.31	0.04	0	0	0	0	0
Modulo Method	0	0	0	0	0	0	0	0
Multiplication Method	2.68	0.91	0.33	0.11	0.03	0.01	0.001	0.0001
Decimal to Base 11	0.18	0	0	0	0	0	0	0
Decimal to Base 15	3.51	0.62	0.08	0.01	0	0	0	0
Universal Hashing	0.37	0.02	0	0	0	0	0	0

D. Analysis of Results

From Table V, which shows the percentage of perfect distribution for different hash functions, we observe that for r = 3 (1,140 subsets), the division remainder method achieved the highest perfect distribution rate at 84.20%, compared to universal hashing at 78.77%. Perfect distribution means the resulting successful length is 1 (no collisions).

Table VI shows the percentage of worst distribution across different hash functions. For r = 3, 22 out of 1140 subsets had all collisions, giving (22/1140) × 100 = 1.93% for the mid-square method. The modulo method demonstrated the lowest worst distribution percentage at 0%.

VI. DISCUSSION

A. Prime Number Effect

The analysis demonstrates that the choice of prime number significantly influences the performance of universal hashing functions. When using the H1 class universal hash function h(x) = ((cx + d) mod p) mod m:

1. **Large prime numbers** (p = 226,637, 156,967) resulted in poor performance with high average search lengths (5.5) and numerous overflow records
2. **Moderate prime numbers** (p = 128,761) yielded average performance with search length 3.0
3. **Small prime numbers** (p = 10,687, 7,477) achieved excellent performance with search lengths of 1.0 and 1.2 respectively

The ratio (cx+d)/p was found to be inversely correlated with performance—smaller ratios led to more collisions and poorer performance.

B. Comparative Performance

The comparison of different hashing techniques revealed:

1. **Division Remainder Method**: Demonstrated the highest percentage of perfect distribution (84.20% for r=3) and the lowest percentage of worst distribution (0% across all subset sizes)
2. **Universal Hashing**: Showed good performance with 78.77% perfect distribution for r=3, but was outperformed by the division remainder method
3. **Decimal to Base 11**: Performed well with 81.40% perfect distribution for r=3
4. **Mid-Square and Multiplication Methods**: Showed lower perfect distribution percentages compared to other methods

VII. CONCLUSION

This study examined the average search length of hash functions using different methods. By considering comprehensive subsets of the Universe, we analysed all possible subsets to understand how hash functions distribute data properly.

The key findings of this study are:

1. **Division-Remainder Method Superiority**: The division-remainder method performed best in terms of distributing subsets with fewer collisions, resulting in a smaller average successful search. This method had the highest percentage of perfect distributions across all subset sizes.
2. **Prime Number Effect**: A smaller value for prime number 'p' improved the performance of the Universal Hashing function. The performance significantly improved when the ratio (cx+d)/p increased, with the best performance achieved at p = 10,687 (average search length = 1.0).
3. **Performance Classification**: The H1 class universal hash function can exhibit performance ranging from perfect to good, bad, or worst depending on key selection and parameters.
4. **Comparative Analysis**: Among the evaluated methods, the modulo (division remainder) method achieved the highest perfect distribution percentage (84.20% for r=3) and lowest worst distribution percentage (0%), making it the most reliable for minimizing collisions.

This extensive analysis serves as a valuable resource for understanding the significance of hash functions and their performance characteristics under different conditions. Future work could explore additional hash function families and their performance under various operational scenarios.

References

[1] W. Stallings, *Network and Internetwork Security: Principles and Practice*. Prentice-Hall, Inc., 1995.
[2] S. M. Naser, "Cryptography: From the ancient history to now, it's applications and a new complete numerical model," *International Journal of Mathematics and Statistics Studies*, vol. 9, no. 3, pp. 11-30, 2021.
[3] E. Swathi, G. Vivek, and G. S. Rani, "Role of hash function in cryptography," *Int. J. Adv. Eng. Res. Sci. (IJAERS)*, 2016.
[4] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 2018.
[5] H. F. Al-Layla, F. N. Ibraheem, and H. A. Hasan, "A review of hash function types and their applications," *Wasit Journal of Computer and Mathematics Science*, vol. 1, no. 3, 2022.
[6] G. M. Sridevi, M. V. Ramakrishna, and D. V. Ashoka, "Comprehensive performance study of hashing functions," *Computer Science Journal of Moldova*, vol. 31, no. 2, 2023.
[7] E. Unsal, "A review of hashing algorithm," in *1st International Conference on Frontiers in Academic Research*, 2023, pp. 544-550.
[8] National Institute of Standards and Technology (NIST), "Secure Hash Standard," FIPS Publication 180-1, 1995.
[9] P. Gauravaram, "Cryptographic hash functions: Cryptanalysis, design and applications," Ph.D. dissertation, Queensland University of Technology, 2007.
[10] U. Iqbal and Y. Bhutto, "Digital transformation through artificial intelligence and advance business analytic in American operational management," *Journal of Theoretical and Applied Econometrics*, vol. 3, no. 1, pp. 37-50, 2026.
[11] National Institute of Standards and Technology (NIST), "Secure Hash Standard," FIPS Publication 180-2, 2002.
[12] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for hash functions MD4, MD5, HAVAL-128, and RIPEMD," School of Mathematics and System Science, Shandong University, Jinan, China, 2004.
[13] U. Iqbal, "AI-enhanced network optimization for electric vehicle charging infrastructure expansion in the United States using graph theory and demand analytics," *Journal of Engineering and Computational Intelligence Review*, vol. 2, no. 2, pp. 112-129, 2024.
[14] S. Chang, R. Perlner, W. Burr, M. Sonmez, J. Kelsey, S. Paul, and L. Bassham, "Third-round report of the SHA-3 cryptographic hash algorithm competition," *NIST Interagency/Internal Report (NISTIR)*, National Institute of Standards and Technology, 2012.
[15] E. Swathi, "Role of hash functions in cryptography," *International Journal of Advanced Engineering Research and Science*, pp. 10-13, 2016.

[16] K. Rajeshwaran and K. A. Kumar, "Cellular automata based hashing algorithm (CABHA) for strong cryptographic hash function," in *Proc. IEEE International Conference on Electronics, Computer and Communication Technology (ICECCT)*, 2019.

[17] N. Kheshaifaty and A. Gutub, "Preventing multiple accessing attacks via efficient integration of captcha crypto hash functions," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 20, no. 9, pp. 16-28, 2020.

[18] 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-document," *IEEE Access*, pp. 80290-80304, 2020.

[19] P. William and A. Choubey, "Assessment of hybrid cryptographic algorithm for secure sharing of textual and pictorial content," in *Proc. International Conference on Electronic Renew System (ICEARS)*, Tuticorin, India, 2022, pp. 918-922.

[20] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, 1976.

[21] B. V. Rompay, "Analysis and design of cryptographic hash functions, MAC algorithms and block ciphers," Ph.D. thesis, Electrical Engineering Department, Katholieke Universiteit, Leuven, Belgium, 2004.

[22] U. Iqbal, "AI-driven predictive maintenance for US smart manufacturing: Deep learning models for equipment failure prediction and operational resilience," *Journal of Engineering and Computational Intelligence Review*, vol. 3, no. 1, pp. 114-138, 2025.