ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

Enhancing Password Storage Through the Integration of Cryptarithmetic Techniques and Hash Functions

Jakkapong Polpong[1], Sompond Puengsom[2], Noppasak Tantisattayanon[3] and Phisit Pornpongtechavanich[4]

ABSTRACT

The utilization of internet-based applications offers numerous benefits and significantly enhances the daily lives of individuals. To access these systems, users must provide unique username and password credentials, verifying their identity as the legitimate owners of the data. However, there are instances where unauthorized individuals, including hackers, gain access to these systems by illegitimately exploiting these credentials. This study aims to enhance the system's efficiency by introducing a novel algorithm named Cryptarithmetic_shield. This algorithm combines the Cryptarithmetic technique with a hash function to create a secure encryption system. The research results indicate that password data encrypted using the Cryptarithmetic_shield algorithm, in conjunction with the hash functions MD5, SHA1, SHA256, SHA512, CRC32, and RIPEMD160, effectively prevents decoding from Dictionary attack and Rainbow Table Attack methods, achieving up to 100% effectiveness.

Article information:

Keywords: Cryptarithmetic, Authentication, Hash Function, Password Security, Secure Password Storage

Article history:
Received: August 19, 2023
Revised: December 7, 2023
Accepted: February 8, 2024
Published: April 6, 2024 (Online)

DOI: 10.37936/ecti-cit.2024182.253861

1. INTRODUCTION

In contemporary times, individuals and organizations are increasingly aware of the potential risks associated with utilizing applications or sharing personal information online [1-3]. To safeguard personal information, a standard known as Confidentiality, Integrity, and Availability (CIA) has been established [4-5]. It is crucial that online transaction platforms, such as e-commerce and e-banking, adhere to CIA standards for security purposes [6].

Currently, there are several ways to access mobile applications and websites, including text passwords, biometrics, and hardware tokens. However, text password access remains the most commonly used and the least secure option [7]. In many applications and websites, a user's username and password are often set similarly [8-9]. If a hacker gains access to a user's username and password, they can use techniques such as Brute-force and dictionary attacks to crack the password [10-11]. Once a password has been decoded, it is displayed in plaintext and can be used to attempt unauthorized access to websites and applications [12-13].

Consequently, various techniques and strategies have been developed to prevent hacking attacks. Hashing and encryption are methods utilized to store passwords that provide optimal security and are challenging to decode [14-15].

Hash functions are currently the preferred method for storing keys, with MD5, SHA1, SHA256, AES, and DES being the most widely utilized [16-18]. If users employ easily guessable passwords (characters and numbers) with the aforementioned hashing methods, the resulting hashed passwords cannot be guaranteed to be secure against the decryption techniques of a Rainbow Table Attack (where attackers compare the hash values of stolen passwords with entries in a Rainbow Table; if they match, it enables hackers to backtrack and find the original password) [19-20].

To combat this, the objective of this research is to combine "cryptarithmetic" with hash functions, such as MD5, SHA1, SHA256, SHA512, CRC32, and RIPEMD160, to improve data security and increase decoding complexity (protecting against Dictionary attacks and Rainbow Table attacks).

[1,2,3,4] The authors are the Department of Information Technology, Faculty of Industry and Technology, Rajamangala University of Technology Rattanakosin, Hua Hin, Thailand, E-mail: Jakkapong.pol@rmutr.ac.th, Sompond.pue@rmutr.ac.th, Noppasak.tan@rmutr.ac.th and Phisit.kha@rmutr.ac.th
[4] Corresponding author: Phisit.kha@rmutr.ac.th

This research article is organized into five sections, which consist of an introduction, background and literature review, proposed method, results and discussion, and conclusions and future work.

2. LITERATURE SURVEY

In this section, the methods for enhancing password security, the concepts of Cryptarithmetic, and the functioning of hash functions are explained. Additionally, pertinent research on these subjects is provided. Passwords can be improved and made more secure in various ways. Bhana and Flowerday [21] utilized Shannon's entropy theory approach to enhance password security, employing three design philosophies: security, memory, and typing. Kävrestad *et al.* [22] presented a range of password-creation strategies derived from a sample of the Swedish police through a series of semi-structured interviews with forensic experts.

Cryptarithmetic is a mathematical game that involves equations between characters and numbers, where identical characters have the same numerical value. This topic has been studied in various works [23-24]. By replacing the characters in the message with numbers, the objective is to conceal the answers to the mathematical puzzles used to calculate the characters included within the numbers [25-26], as shown in Fig. 1 and Fig. 2.

		S	E	N	D
+		M	O	R	E
=	M	O	N	E	Y

Fig.1: *An Example of a Cryptarithmetic problem.*

Fig. 1 shows a classic puzzle by Dudeney [27], published in the July 1924 issue of Strand Magazine, as an example [28].

```
  9567
+ 1085
--------------
 10652
```

Fig.2: *An appropriate solution to the puzzle is shown in Fig. 1.*

Fig. 2 shows the calculation process for a cryptarithmetic puzzle, with the solution being the word "MONEY." In this puzzle, each character is assigned a numerical value: O = 0, M = 1, Y = 2, E = 5, N = 6, D = 7, R = 8, and S = 9. The puzzle is then solved by performing arithmetic operations on these values to find the correct answer.

The issue was popularized in the May 1931 issue of Sphinx, a Belgian journal of recreational mathematics [29]. "Cryptarithmetic" was published by Kraitchik [30]. Yang [31] developed a program that utilizes cryptarithmetic and other techniques to determine the solution to a given problem. Abbasian and Mazloom [32] employed a genetic algorithm to solve cryptarithmetic problems in parallel, which is similar to the research carried out by Minhaz and Singh [33] on the Classical CryptArithmetic Problem (CAT + RAN = AWAY). Furthermore, Fontanari [34] utilized Cryptarithmetic with Social Learning Heuristic to identify the optimal grouping.

A hash function, also referred to as a hash, is a mathematical function that transforms plaintext or data, accessible to anyone who can read and under-

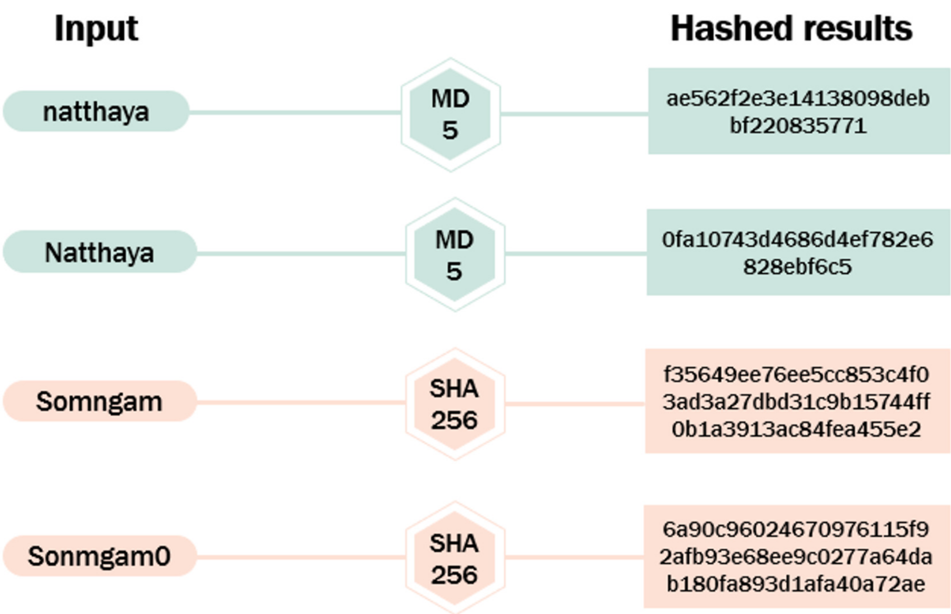

Fig.3: *Encryption using a hash function algorithm.*

stand it, into a fixed-size output. The purpose of this transformation is to make it practically impossible for humans to understand the original plaintext. Hash operations are one-way operations, meaning that once a password is entered into the hash algorithm, the password's plain text format cannot be recovered, as shown in Fig. 3 [35-36].

Fig. 3 shows how to use hash functions with MD5 and SHA256 algorithms to transform input plain-text data into a fixed-length character set. Regardless of the input size, SHA-256 requires 256 bits, MD5 requires 128 bits, and so on. The hashing operation produces different results if the message differs by just one character or if both uppercase and lowercase characters are used.

The length of the output produced by hash functions varies depending on the type of hash function used. MD5 generates a 128-bit string, SHA1 generates a 160-bit string (40 characters), SHA256 generates a 256-bit string (64 characters), and SHA512 generates a 512-bit string (128 characters). Longer output lengths require more calculation time and storage space to store the result value. However, longer output lengths can also increase the security of the decryption against Brute-Force and Rainbow Table attacks [37-39].

Applying hash functions in combination with other techniques can increase security and make data more difficult to decrypt. For example, Pal *et al.* [40] used four stages of bit interspersing and a 4D-hyperchaotic system with hashing functions to protect the rights of images. Almahmoud et al. [41] introduced a new hash function called HashComb that uses Distance-Preserving and Multi-Hash techniques for calculation with basic hash libraries, aiding in analyzing hierarchical spacing. Polpong and Wuttidittachotti [42] developed the SXR algorithm, which combines three equations (ratio, XOR, and Replace) with hash functions to enhance password security. Upadhyay et al. [43] studied the effects of 16 hash functions in applications using the Public Key Cryptography Standard (PKCS) and Hash-based message authentication code (HMAC). As a result, only 50% of input parameters and hash functions could satisfy the Strict Avalanche Criterion (SAC) and Bit Independence Criteria (BIC).

To generate a new result using the Cryptarithmetic technique, two parameters are required. Therefore, the researcher assumed that the username data could be set as the first parameter and the password as the second parameter, which would then be calculated using cryptarithmetic. To increase security, the resulting value would be hashed before storing it in the database.

3. PROPOSED METHOD

The proposed technique involves three steps. Step 1 consists of creating a pattern using unique characters to define the numerical value of each character in the username and password, as shown in Fig. 4. Step 2 optimizes the algorithm's performance using the cryptarithmetic technique, as shown in Fig. 5. Finally, Step 3 applies a hash function to the results obtained from Step 2, as shown in Fig. 6.

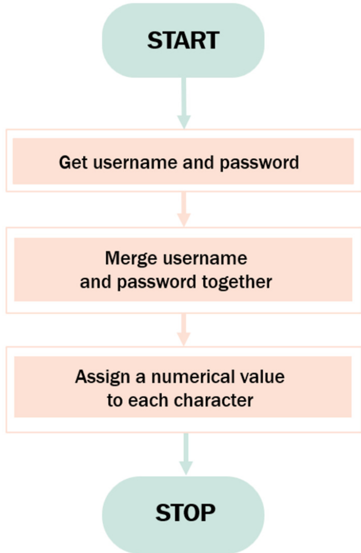

Fig.4: *The process of creating characters.*

In the first step (Fig. 4), a username and password are collected, and a numerical value is assigned to each character, with duplicate characters being counted only once, as shown in Fig. 7.

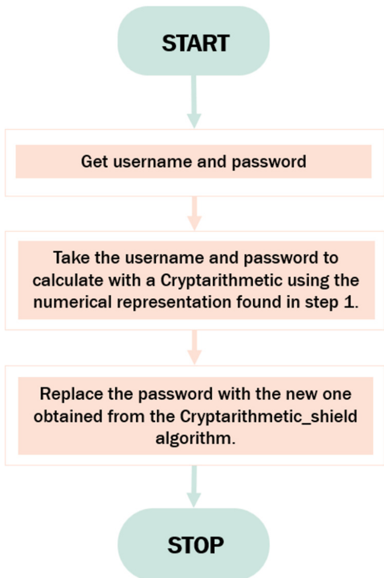

Fig.5: *The process of the Cryptarithmetic_shield algorithm.*

Next, in Step 2 (Fig. 5), the Cryptarithmetic shield algorithm is applied, using the values obtained in Step 1 as the algorithm's input. The resulting output is shown in Fig. 10.

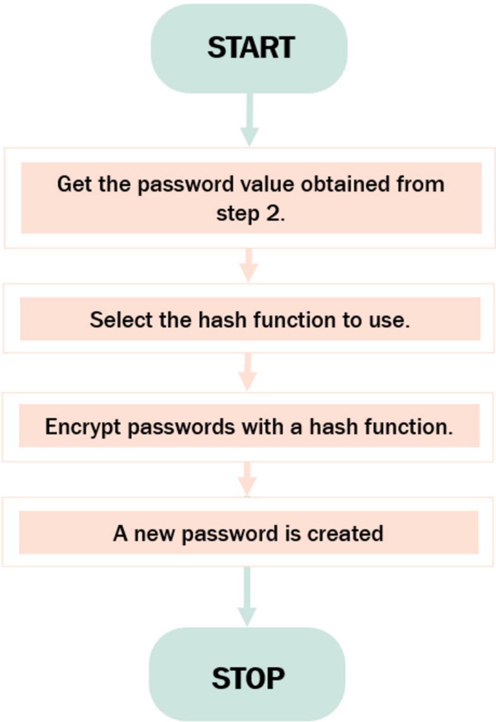

Fig.6: *The process of hash functions.*

Finally, in Step 3 (Fig. 6), the password generated using the Cryptographic Shield algorithm is combined with a hash function. The resulting data is stored in the database, as demonstrated in Fig. 11.

3.1 Algorithm Cryptarithmetic_shield

In Step 1, the username "Natthaya13" and password "Somngam08" are established. The combined username and password for Step 2 is "Natthaya13Somngam08".

When applying the Cryptarithmetic method, as shown in Fig. 7, each duplicate character is counted only once. For example, in the given sample, the letter "a" appears four times, but only one numerical value is assigned to it. The duplicate characters are removed sequentially from left to right, and each character is given a unique numeric value. The results are stored in a dictionary-based array named "cryptarithmetic dic," which holds this data.

The "cryptarithmetic_dic" array uses a list-based data storage format where each element has a unique key and a corresponding value, as explained in Fig. 8. The key can be a string or a numerical value, and each member must have a distinct key. The value can be of any data type, and the value of each member can be duplicated. The key associated with each character is used to determine its value.

Fig. 8 shows the characters associated with the numbers obtained in Fig. 7, which are stored as a dictionary for use in the Cryptarithmetic method. Additionally, the numerical values of the letters "S" and "n" are shown in Fig. 8, along with their corresponding vital values. The character "S" has a numerical value of 7, and the character "n" has a numerical value of 10.

In the second step, the Cryptographic_shield al-

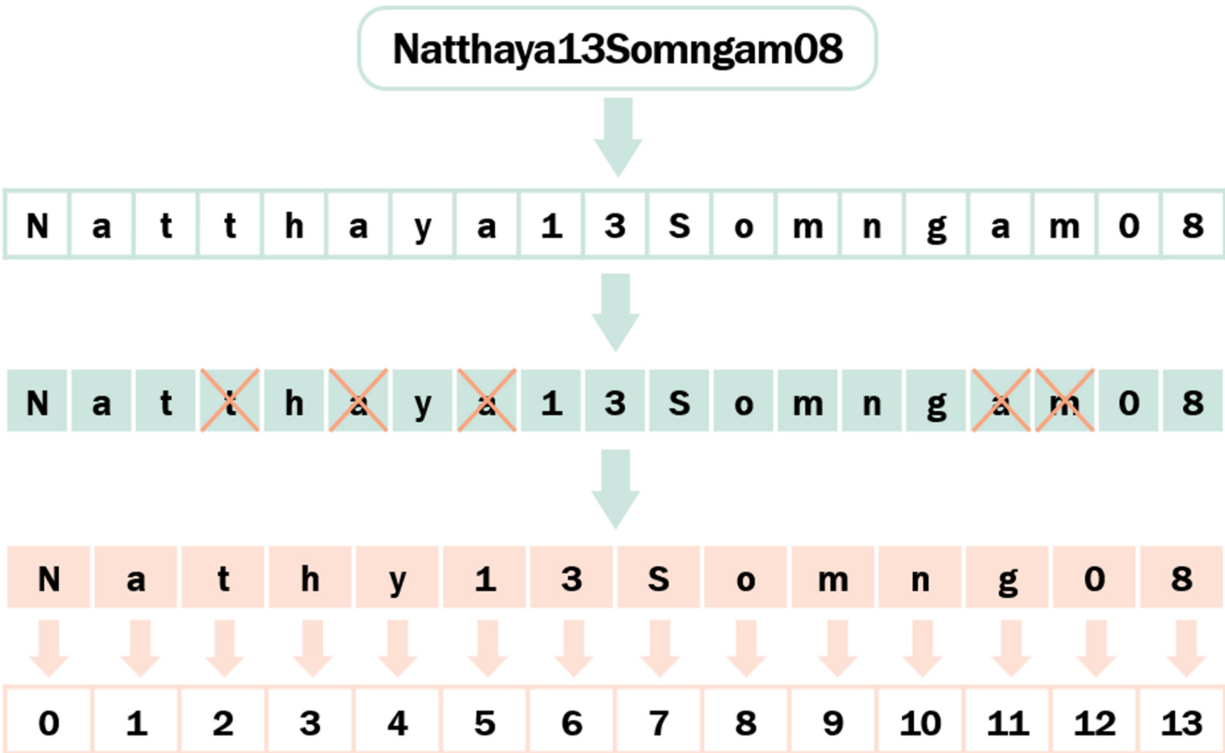

Fig.7: *Identifying and assigning values to each character.*

Key	Value
N	0
a	1
t	2
h	3
y	4
1	5
3	6
S	7
o	8
m	9
n	10
g	11
0	12
8	13

Fig.8: *Accessing dictionary members.*

gorithm calculates the username (Natthaya13) and password (Somngam08). If the character count of the username (10 characters) and the character count of the password (9 characters) are different, the first character of the string should be added after the last character to make the character count equal. For example, if the password "Somngam08" has nine characters and the username has 10, a new password with the same number of characters should be built: "Somngam08S" (shown in Fig. 9).

If the username has 12 characters but the password only has 9, the algorithm includes the first three characters of the password (Som). For example, if the password is Somngam08, the new password generated would be Somngam08Som.

The Cryptarithmetic_shield algorithm applies to the referenced numbers from the cryptarithmetic_dic data generated in step 1, proceeding from left to right. The first character of the username and password is "N" and "S," respectively, and the maximum value of the Value in the cryptarithmetic_dic variable is modulated into the outcome (in this case, 13). For example, the sum of "h" (3) and "g" (11) is 14 (3 + 11), and mod 13 equals 1, or "a" (as shown in Fig. 10).

Fig. 10 shows the encryption of the username and password data using the Cryptarithmetic_shield algorithm, with the new password generated by the method being "Smg0at8818."

The final step is to hash the new password generated using the Cryptarithmetic_shield algorithm (the user can choose which hashing technique to use). The resulting value is saved to the database, as shown in Fig. 11.

Fig. 11 provides an example of a new password being hashed with MD5 and SHA1. The length of the code string varies depending on the hash function chosen.

The entire operational process of the algorithm presented, known as Cryptarithmetic_shield, will be illustrated in "Figure 12."

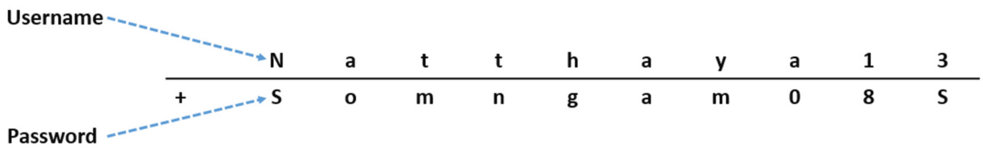

Fig.9: *The placement of strings when the number of characters in the username and the password are different.*

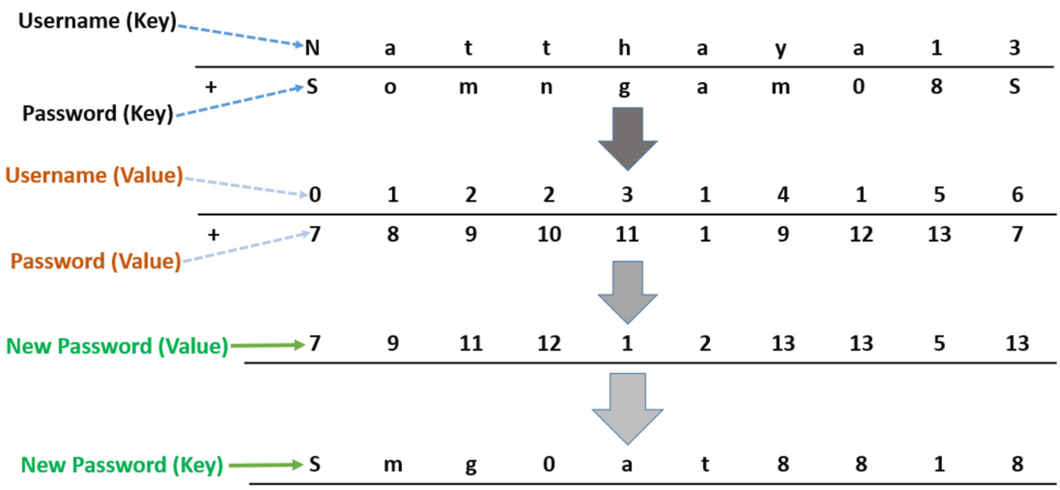

Fig.10: *Cryptarithmetic shield algorithm functions.*

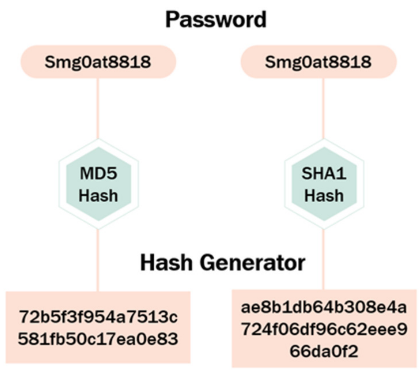

Fig.11: *Execute hash function.*

Fig. 12 comprises three components: 1. Input, responsible for receiving values such as username, password, and hash function; 2. Process, tasked with aiding in the processing and computation of the algorithm's equations; 3. Output, representing the results obtained from the operational process of the Cryptarithmetic_shield algorithm.

4. RESULTS AND DISCUSSION

In this research, 1000 passwords were selected, including the most commonly used passwords in 2022, and created based on "Password Entropy and Password Quality" [44]. The efficiency, time, and attack defense efficacy of the Cryptarithmetic_shield algorithm were then tested.

4.1 Dataset

The dataset included the most popular usernames [45], passwords that were switched out with lines between popular passwords [46], and the passwords that the researcher generated (a total of 1,000 passwords), as shown in Table 1.

Table 1: *Example of data used in performance testing.*

ID	Username	Password
1	ยศกร	password
2	Alex	123456
3	Maria	89VDsio$
4	Jessica	123456789
5	Alessandro	EFIJAPZQ
6	Valentina	guest
7	Alexander	R38svF^0m
8	Caroline	o8,Vqs]2^y
9	Sylvie	5Lt]8+dI4H
10	Oscar	111111

Table 1 shows the sample dataset used for the test, which includes popular usernames and passwords with passwords composed of alphanumeric characters (0–9), lowercase characters (a–z), uppercase characters (A–Z), and special characters (! @ ♯ $ % & ∗ () _ + [] :>?).

4.2 Time Efficiency

The time efficiency of the Cryptarithmetic_shield algorithm was compared to ordinary hash functions (operating system: Windows 11 64-bit, Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, RAM: 16GB, Python 3.11). Six hashing methods were tested, namely MD5, SHA1, SHA-256, SHA-512, CRC32, and RIPEMD160. The outcomes of 100 tests were averaged, and the research results are shown in Table 2.

According to Table 2, the usernames "ยศกร" and "password" were used to compare efficiency. The Cryptarithmetic_shield algorithm was found to take slightly longer than the original hash algorithm because it requires the creation of a new password before it can hash. The incremental time difference for MD5 is 0.085 ms, for SHA1 is 0.767 ms, for SHA-256 is 0.807 ms, for SHA-512 is 1.136 ms, for CRC32 is 1.273 ms, and for RIPEMD160 is 1.289 ms. Also, when deployed, this difference in the time frame is not noticeable to humans [47].

4.3 Resistance to Attacks

This section discusses the comparison of attack prevention. The results of the study are shown in Table 3.

The effectiveness of the Cryptarithmetic_shield algorithm in combination with six hashing techniques for defense against Dictionary attacks (which can decrypt passwords to plaintext) is also shown in Table 3. It is demonstrated that the algorithm can effectively thwart attacks. Furthermore, verification against a rainbow table reveals that none of the hash values generated by the algorithm correspond to entries in the rainbow table. This contrasts with the direct hashing of commonly used passwords, where almost every hashed password can be traced back to that same rainbow table. (In the future, researchers plan to subject this algorithm to various contemporary attack methods for further evaluation).

4.4 Cryptarithmetic_shield Algorithms

This section provides the pseudocode for the Cryptarithmetic_shield algorithm. The algorithm requires input of the username, password, and hashing method to operate.

Input: Username = U
Password = P
Select hash function = S_h

Output: Hashed password using Cryptarithmetic_shield algorithm.

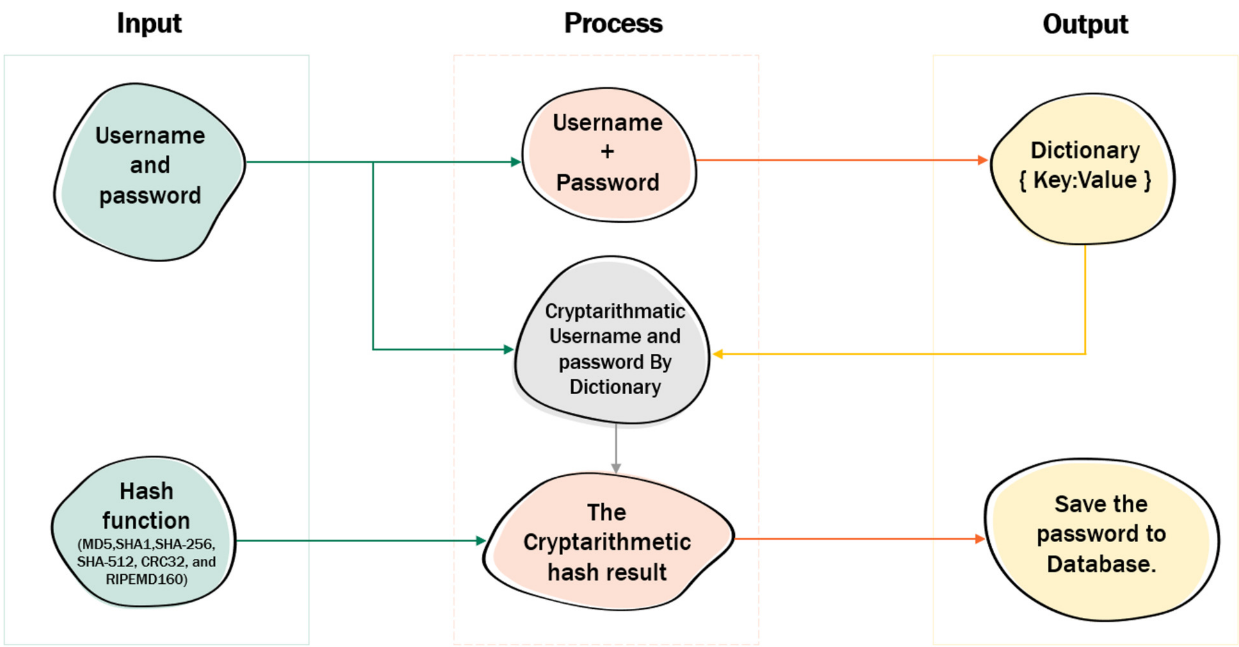

Fig.12: *Algorithm Cryptarithmetic_shield.*

Table 2: *Performance Comparison (average time).*

Algorithms	Hash Original	Cryptarithmetic_shield	Hash Original (ms)	Cryptarithmetic_shield (ms)	The difference (ms)
MD5	5f4dcc3b5aa765d61d8327deb882cf99	d0375ec9d74d835cf3525b50197db916	31.247	31.332	0.085
SHA1	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	941faddd996643780883cfdea9ee8d55d16e250f	33.249	34.016	0.767
SHA256	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8	76128e92e90f3662c7f5138ba03a7db69e635ee660513d63ee17b1e95d4b5050	39.906	40.713	0.807
SHA512	b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7785e5976ec049b46df5f1326af5a2ea6d103fd07c95385ffab0cacbc86	21cd912ec69ff1878ba5dde7d3bd9b799a5554f649acc9884c73a1ccff5eab0a9572f420430dd21ed813e934abf65ae7c48cc6b19616e0f02f3655a599ca678c	46.874	48.011	1.136
CRC32	901924565	2365005922	15.620	16.893	1.273
RIPEMD160	108f07b8382412612c048d07d13f814118445acd	acac5a72cc3519969a993f106d808deaec5ddbe2	15.624	16.914	1.289

Cryptarithmetic_shield Algorithms

Start

```
D ← {...}                          ###Dictionary variable
N_b ← 0                            ###Numerical value
i ← 1
M ← U + P                          ###Merge U and P together
for All M do
    if M_i ∉ D then                ###Verify that letters are not unique in D
        D[M_i] = N_b;              ###Add M_i to key and add to N_b value
        N_b ← N_b + 1;
    else
        Next;
    end
end for
if U(length) < P(length) then
    Tmp_number ← P(length) - U(length)
    U ← U + U[0:Tmp_number]
end
if P(length) < U(length) then
    Tmp_number ← U(length) - P(length)
    P ← P + P[0:Tmp_number]
end
while U(length) >= i do            ###Check if the last letter of U or P
    R ← D[U_i] + D[P_i]            ###Add the value of U and P obtained from D
    A_I ← D.get(R)                 ###Find the key that matches the R value and store it in A_I
    C_sh ← C_sh + A_I              ###Take the result of A_I and place it after C_sh
    i ← i + 1
end while
h ← S_h(C_sh)                      ###Hash C_sh
```

Stop

Notation

D	=	Dictionary variable
N_b	=	Numerical value
A_i	=	Result Value of the character D[R]
R	=	Result from the calculation
C_{sh}	=	Result value from the algorithm

Table 3: *Dictionary Attack and Rainbow Table Attack.*

Algorithms with hash function	defense against Dictionary attack (%)	defense against Rainbow table attack (%)
MD5	100%	100%
SHA1	100%	100%
SHA-256	100%	100%
SHA-512	100%	100%
CRC32	100%	100%
RIPEMD160	100%	100%

5. CONCLUSIONS AND FUTURE WORK

In this research, the combination of cryptarithmetic methods and hashing techniques has been proposed to enhance the security of applications and websites.

The results demonstrate that applying hash functions and Cryptarithmetic can potentially enhance the security of passwords and other private data against hackers. Even if a dictionary attack or brute force is used, the passwords will be encrypted and cannot be transformed into plaintext. Hence, similar attacks can be prevented with data encrypted via this algorithm.

Subsequent research endeavors could concentrate on evaluating the algorithm using a more extensive dataset and exploring various attack methodologies. Additionally, it would be beneficial to apply the algorithm in contexts such as financial transactions or identity verification procedures as experimental subjects.

FUNDING

This work is funded by Rajamangala University of Technology Rattanakosin revenue budget for the fiscal year 2023 with research contract number C39/2566.

ACKNOWLEDGEMENT

Thanks to Rajamangala University of Technology Rattanakosin for supporting this study.

AUTHOR CONTRIBUTIONS

Jakkapong Polpong studied relevant research documents, planning work, research design, conduct research, Algorithm development, analyze the results, criticize the result and co-write article. Phisit Pornpongtechavanich studied relevant research documents, planning work, research design, conduct research, gather information, analyze the results, criticize the result, conclusion and write main article. Sompond Puengsom studied relevant research documents, planning work, research design, criticize the result and conclusion. Noppasak Tantisattayanon studied relevant research documents, planning work, research design, criticize the result and conclusion. All authors had approved the final version.

References

[1] P. S. Rani and S. B. Priya, "Security-Aware and Privacy-Preserving Blockchain Chameleon Hash Functions for Education System," *ECTI-CIT Transactions*, vol. 17, no. 2, pp. 225–234, Jun. 2023.

[2] M. N. Hoque, P. Chakraborty and M. H. Seddiqui, "The Challenges and Approaches during the Detection of Cyberbullying Text for Low-resource Language: A Literature Review," *ECTI-CIT Transactions*, vol. 17, no. 2, pp. 192–214, Apr. 2023.

[3] C. Mike, "Today's Information Security Manager," in *CISM Certified Information Security Manager Study Guide*, Wiley, pp.1-30, 2022.

[4] S. Barakovic and J. B. Husic, "Cyber hygiene knowledge, awareness, and behavioral practices of university students," *Information Security Journal: A Global Perspective*, Taylor & Francis, pp.1-24, 2022.

[5] V. S. Basie and v. S. Rossouw, "Cybersecurity and information security – what goes where?," *Information and Computer Security*, vol. 26 No. 1, pp. 2–9, 2018.

[6] A. O. Aljahdalic, S. Banafee and T. Aljohani, "URL filtering using machine learning algorithms," *Information Security Journal: A Global Perspective*, Taylor & Francis, 2023.

[7] T. P. Fowdur and S. Hosenally, "A real-time machine learning application for browser extension security monitoring," *Information Security Journal: A Global Perspective*, Taylor & Francis, 2022.

[8] H. -M. Ying and N. Kunihiro, "Decryption of Frequent Password Hashes in Rainbow Tables," *2016 Fourth International Symposium on Computing and Networking (CANDAR)*, pp. 655-661, 2016.

[9] D. P. Joseph and P. Viswanathan, "SDOT: Secure Hash, Semantic Keyword Extraction, and Dynamic Operator Pattern-Based Three-Tier Forensic Classification Framework," in *IEEE Access*, vol. 11, pp. 3291-3306, 2023.

[10] Laatansa, R. Saputra and B. Noranita, "Analysis of GPGPU-Based Brute-Force and Dictionary Attack on SHA-1 Password Hash," *2019 3rd International Conference on Informatics and Computational Sciences (ICICoS)*, pp. 1-4, 2019.

[11] L. Bošnjak, J. Sreš and B. Brumen, "Brute-force and dictionary attack on hashed real-world passwords," *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1161-1166, 2018.

[12] RG. Rittenhouse and JA. Chaudhry, "A Sur-

vey of Alternative Authentication Methods," *International Conference on Recent Advances in Computer Systems (RACS 2015)*, pp. 179-182, 2015.

[13] I. Neforawati and D. Arnaldy, "Message Digest 5 (MD-5) Decryption Application using Python-Based Dictionary Attack Technique," *2021 4th International Conference of Computer and Informatics Engineering (IC2IE)*, pp. 424-428, 2021.

[14] B. Savege, "A Guide to Hash Algorithms," *GIAC Practical Repository*, SANS Institute, available at: https://www.giac.org/paper/gsec/2853/guide-hash-algorithms/104822 (accessed 10 September 2022).

[15] B. A. Forouzan, "Cryptography and Network Security,"*McGraw-Hill*, Inc., pp. 1-480. Available at: http://almuhammadi.com/sultan/books_2020/Forouzan.pdf (accessed 10 September 2022).

[16] R. Rivest, "RFC1321: The MD5 Message-Digest Algorithm," *RFC Editor.*, available at: https://dl.acm.org/doi/book/10.17487/RFC1321 (accessed 10 September 2022).

[17] A. A. A B, A. Gupta and S. Ganapathy, "A New Security Mechanism for Secured Communications Using Steganography and CBA," *ECTI-CIT Transactions*, vol. 16, no. 4, pp. 460–468, Oct. 2022.

[18] E. M. Michail, G. Athanasiou, F. Theodoridis, A. Gregoriades and E. C. Goutis, "Designand Implementation of Totally-Self Checking SHA-1 and SHA-256 Hash Functions Architectures," *Microprocessors and Microsystems*, vol. 45, pp. 227-240, 2016.

[19] G. Brose, "Rainbow Tables," *Encyclopedia of Cryptography and Security*, Springer, pp. 1021-1022, 2011.

[20] L. Zhang, C. Tan and F. Yu, "An Improved Rainbow Table Attack for Long Passwords," *Procedia Computer Science*, vol. 107, pp 47-52, 2017.

[21] B. Bhana and S. V. Flowerday, "Usability of the login authentication process: passphrases and passwords," *Information and computer security*, vol. 30 no. 2, pp 280–305, 2022.

[22] J. Kävrestad, F. Eriksson and M. Nohlberg, "Understanding passwords a taxonomy of password creation strategies," *Information and computer security*, vol. 27, no. 3, pp. 453–467, 2019.

[23] S. Widodo, U. Najati and P. Rahayu, "A Study of Cryptarithmetic problem-solving in elementary school," *Journal of Physics: Conference Series*, vol. 1318, pp. 1-8, 2019.

[24] V. Goel, "A comparison of design and nondesign problem spaces," *Artificial Intelligence in Engineering*, vol.9, no.1, pp. 53-72, 1994.

[25] B. Averbach and O. Chein, "Problem Solving Through Recreational Mathematics," *Part of: Dover Books on Mathematics*, available at: https://archive.org/details/Problem.Solving.Through.Recreational.Mathematics/mode/2up (accessed 10 September 2022).

[26] D. St. P. Barnard, "Adventures in Mathematics," *Pelham Books*, London, 2003.

[27] H. E. Dudeney, "The Strand Magazine," in *Strand Magazine*, vol.68, pp. 97 and 214, 1924.

[28] S. K. Namdeo, G. S. Kumar and P. Kuyate, "Crypto-Arithmetic Problem using Parallel Genetic Algorithm (PGA)," *International Journal of Modern Engineering Research (IJMER)*, vol.2, no.5, pp. 3275-3280, 2012.

[29] B. Averbach, O. Chein and H. C. Wolfe, "Mathematics: Problem Solving Through Recreational Mathematics," *Physics Today-PHYS TODAY*, vol. 34, available at: https://www.researchgate.net/publication/243387866_Mathematics_Problem_SolvingThrough_Recreational_Mathematics (accessed 10 September 2022).

[30] M. Kraitchik, "Mathematical Recreations," *Dover Publications*, Inc. NEW YORK, 2019.

[31] F. Yang, "Solving Cryptarithmetic Puzzles by Logic Programming," *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, pp. 1389-1394, 2020.

[32] R. Abbasian and M. Mazloom, "Solving Cryptarithmetic Problems Using Parallel Genetic Algorithm," *2009 Second International Conference on Computer and Electrical Engineering*, IEEE, pp. 308-312, 2009.

[33] A. Minhaz and A. V. Singh, "Solution of a Classical Cryptarithmetic Problem by using parallel genetic algorithm," *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization*, pp. 1-5, 2014.

[34] J. F. Fontanari, "Solving a cryptarithmetic problem using a social learning heuristic," *2014 IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB)*, pp. 65-70, 2014.

[35] M. I. Mihailescu and S. L. Nita, "Cryptography and Cryptanalysis in MATLAB," *Creating and Programming Advanced Algorithms*, pp. 83–102, 2021.

[36] L. Huang, Q. Yang and W. Zheng, "Online Hashing," *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, vol. 29, no. 6, pp. 2309-2322, 2017.

[37] V. Thing and H. Ying, "Enhanced Dictionary Based Rainbow Table," *Information Security and Privacy Research*, Springer, pp.513-524, 2012.

[38] E. İ. Tatli, "Cracking More Password Hashes

with Patterns," in *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 8, pp. 1656-1665, 2015.

[39] H. Kumar, S. Kumar, R. Joseph, D. Kumar, S. K. S. Singh, A. Kumar and P. Kumar, "Rainbow table to crack password using MD5 hashing algorithm," *IEEE Conference on Information and Communication Technologies*, pp. 433-439, 2013.

[40] S. Pal, A. Mahanty, A. Pathak, J. Karmakar, H. Mondal and M. Mandal, "A Novel Image Encryption Technique with Four Stage Bit-Interspersing and A 4D-Hyperchaotic System," *ECTI-CIT Transactions*, vol. 17, no. 1, pp. 105–116, Mar. 2023.

[41] A. Almahmoud, E. Damiani and H. Otrok, "Hash-Comb: A Hierarchical Distance-Preserving Multi-Hash Data Representation for Collaborative Analytics," in *IEEE Access*, vol. 10, pp. 34393-34403, 2022.

[42] J. Polpong and P. Wuttidittachotti, "Authentication and password storing improvement using SXR algorithm with a hash function," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 6, pp. 6582-6591, 2020.

[43] D. Upadhyay, Gaikwad, M. Zaman and S. Sampalli, "Investigating the Avalanche Effect of Various Cryptographically Secure Hash Functions and Hash-Based Applications," in *IEEE Access*, vol. 10, pp. 112472-112486, 2022.

[44] W. Ma, J. Campbell, D. Tran and D. Kleeman, "Password Entropy and Password Quality," *2010 Fourth International Conference on Network and System Security*, pp. 583-587, 2010.

[45] NordPass, "The 200 Most Popular Usernames of All Time," available at: `https://nordpass.com/blog/all-time-most-popular-usernames/` (accessed 10 January 2023).

[46] NordPass, "Top 200 most common passwords," available at: `https://nordpass.com/most-common-passwords-list/` (accessed 10 January 2023).

[47] R. B. Miller, "Response time in man-computer conversational transactions," *The roceedings of the fall joint computer conference*, part I, pp. 267-277, 1968.

Jakkapong Polpong is a Dr. in the Department of Information Technology at the Faculty of Industry and Technology, Rajamangala University of Technology Rattanakosin Wang Klai Kangwon Campus, Thailand. In 2010, he graduated with a bachelor's degree majoring in Computer Science. Later, in 2014, he earned a Master's degree in Information Technology from the Faculty of Information Technology. Finally, in 2020, he completed his Ph.D. in Information Technology from the Faculty of Information Technology at King Mongkut's University of Technology North Bangkok. His research interests include Web Applications, VoIP Quality Estimation, Computer Programming/Coding, Security, and Authentication. (e-mail: jakkapong.pol@rmutr.ac.th).

Sompond Puengsom is currently a lecturer in the Department of Information Technology at the Faculty of Industry and Technology, Rajamangala University of Technology Rattanakosin Wang Klai Kangwon Campus, Thailand. In 2005, he graduated with a bachelor's degree majoring in Information Systems from Rajamangala University of Technology Rattanakosin. Later, in 2013, he earned a Master's degree in Computer and Information Science from Silpakorn University. His research interests include Web Applications, VoIP Quality Estimation, Computer Programming, and Coding. (e-mail: sompond.pue@rmutr.ac.th).

Noppasak Tantisattayanon is an Assistant Professor and Dr. in the Department of Information Technology at the Faculty of Industry and Technology, Rajamangala University of Technology Rattanakosin Wang Klai Kangwon Campus, Thailand. He graduated with a Bachelor's degree in Business Administration (Marketing) from Srinakharinwirot University, a Master's Degree in Computer Technology from King Mongkut's University of Technology North Bangkok, and a Ph.D. in Computer Education from King Mongkut's University of Technology North Bangkok. His research interests include programming, e-commerce, system analysis, and design. (e-mail: noppasak.tan@rmutr.ac.th).

Phisit Pornpongtechavanich is an Assistant Professor and Dr. in the Department of Information Technology at the Faculty of Industry and Technology, Rajamangala University of Technology Rattanakosin Wang Klai Kangwon Campus, Thailand. In 2012, he received his Bachelor of Technology in Information Technology from RMUTR_KKW. He obtained a scholarship to study in Thailand and then earned a Master of Science in Information Technology from King Mongkut's University of Technology North Bangkok (KMUTNB) in 2014. Finally, in 2020, he completed his Ph.D. in the Division of Information and Communication Technology for Education from King Mongkut's University of Technology North Bangkok (KMUTNB). His research interests include Security, Deep Learning, Artificial Neural Networks, Deep Neural Networks, VoIP quality measurement, QoE/QoS, 3G/4G/5G/6G, mobile networks, multimedia communication, Computer Programming, and Coding (e-mail: phisit.kha@rmutr.ac.th).