Hyper BCK-hashing algorithm: employing encoding system based on logical algebra in enhancing the secure hash algorithms

Hussein A. Jad*
Informatics Research Institute City of Scientific Research and Technological Applications (SRTA City)
New Borg ElArab City, Alexandria
Egypt
hussein.aligad@gmail.com

Samy M. Mostafa
Department of Mathematics
Ain Shams University
Roxy, Cairo
Egypt
samymostafa@yahoo.com

Mokhtar A. Abdel Naby
Department of Mathematics
Ain Shams University
Roxy, Cairo
Egypt
abdelnaby@hotmail.com

Bayumy A. B. Youssef
Informatics Research Institute City of Scientific Research and Technological Applications (SRTA City)
New Borg ElArab City, Alexandria
Egypt
bbayumy@gmail.com

Mona S. Kashkoush
Informatics Research Institute City of Scientific Research and Technological Applications (SRTA City)
New Borg ElArab City, Alexandria
Egypt
mkashkoush@srtacity.sci.eg

Ashraf A. Taha
Informatics Research Institute City of Scientific Research and Technological Applications (SRTA City)
New Borg ElArab City, Alexandria
Egypt
ashraftaha1968@gmail.com

∗. Corresponding author

Abstract. Cryptographic algorithms perform essential functions to generate data from digital form to comprehensible patterns such that the permitted user is the only one who can understand the message. In this study, we propose Hyper BCK-Hashing (HBCK-HASHING) Algorithm based on a hyper BCK-valued function and hash function (SHA-2). It targets to enhance the Secure Hash algorithms (SHA-2) with an algorithm of hyper BCK-valued function which based on the redundant encoding to maximize the security level of the cryptographic process of n-ary block codes (U) through maximize the quantity of information with the fewest number of visible characteristics. The redundant encoding based on making a unique - identified HBCK-algebra (H) for n-ary block codes (U) with applying the hyper BCK-valued function on (H) to generate n-ary block codes(U_H). In addition, we perform the computational Secure Hash algorithms on (U_H) to map the size of n-ary block codes (U_H) into a fixed size. The proposed algorithm was evaluated by using the avalanche effect parameter in comparison with the Secure Hash algorithm (512 and 256). Experimental outcomes indicate that the HBCK-HASHING algorithm shows a significant-high.

Keywords: Hyper BCK-algebras, N-ary block codes, secure Hash algorithm(SHA-2), avalanche effect.

1. Introduction

1.1 Logical algebras and its applications on block codes and Hyper structure approach

Logic algebra indicates a conveying for characteristics and conditions from logic to algebra. Logic algebra fulfills methods for the main assignment of artificial intelligence in elucidating the basics of keeping a computer simulates a human in dealing with data. There are numerous attempts to study emerging characteristics of logic algebras like [1, 2, 3, 4]. Recently, there are abundant research papers studied the relationship between logical algebras and block codes. Block codes mirror an essential class of error-correcting codes Which considered effective to encode data in blocks. Error-control codes allow increasing the security of data transmission over noisy communication channels. Luis Hernandez Encinas [5] introduced the notion of R_0- valued function with related features and examined the generating of binary block codes by R_0- valued function. Cristina Flaut [6] examined the relationships between binary block-codes and Hilbert algebras. Also, she suggested other characteristics associated with Hilbert algebras. Samy M. Mostafa et al. [7, 8, 9] offered an efficient method to produce a KU-algebra from binary block code and introduced the notion of KU- valued function with producing binary code from KU- function. Also, they constructed codes by soft sets PU-valued functions. A.B. Saeid et al. [10] presented an algorithm to generate BCK-algebra from n-ary block code. Numerous applications of Hyper structures are employed in pure and applied sciences. Hyper structures approach adapted to the logical algebraic structure BCK-algebra and consisted the concept of Hyper BCK-algebra.

Y. B. Jun et al. [11] clarified that the generalization of BCK-algebra is Hyper BCK-algebra. Authors defined Hyper BCK-algebra and studied some relevant properties. Atamewouetsafacksurdive et al. [12] stated the concept of hyper BCK-function with some properties related, and generated binary codes by the hyper BCK-function through an algorithm allows constructing a hyper BCK-algebra from binary block code.

In the following, we introduce some results associated with hyper BCK-algebras and algebraic structures applications in coding theory that will be applied effectively in the study.

1.2 Secure Hash Algorithms (SHA-512 and SHA-256)

National Institute of Standards and Technology (NIST) announced Secure Hash Algorithms which indicates SHA. It was developed in 1993 as a federal information processing standard. [14]. After discovering a few weaknesses, an insecure hash algorithm called SHA-0 was withdrawn. SHA-1 procedure has a hash value of 20 bytes (160 bits). SHA-2 is a more powerful version than its ancestors (SHA-0, SHA-1). SHA-256 is a member of the SHA-2 group, yielding alike functionality with more security like SHA-384 and SHA-512. It is an iterative and one-way function. SHA-512 is a member of SHA with a message digest 512- bit of length less than 2128. When the length of any message less than 2128 bits is an input to a hash algorithm, the result is a fixed message digest size (512). Also, SHA-256 is a version of SHA with a 256- bit message digest of length less than 264. When the length of any message less than 264 bits is input to a hash algorithm, the result is a fixed message digest size (256). These algorithms allow the purpose of information's integrity. Any modification in the message will make a modified message digest with a high probability [15].A cryptographic hash function directs to ensure different features, which provides high value for message safety. The hash function requires to satisfy the following features [17, 18]:

1. Compression: hash function maps the input message of uncertain finite-size to a value of fixed size.

2. Security of calculation: the hash value of an input message is simple to compute.

3. Pre-image resistance (one-way): it is obstinate to obtain only one input message which hashed to a determined hash value.

4. Weak collision resistance: it is obstinate to detect other messages that have an equal hash value.

5. Strong collision resistance: it is obstinate to detect two separated input messages hashed to the alike hash value.

Currently, countless applications through unrestricted networks require end-to-end protected connections to support authentication and data privacy [1]. Consequently, Cryptography algorithms are necessary for information security.

One of the cryptography algorithms families that the encrypter and decrypter utilize the same secret key is Symmetric-Key Cryptography. These algorithms depend on the agreement on a key from the sender and receiver before transferring their information. These algorithms use a unique key for encryption and decryption. Some popular patterns of Symmetric-Key encryption algorithms are Advanced Encryption Standard, Data Encryption Standard, Rivest Cipher 5, 3DES, Blowfish, etc.

1.3 Applications of Secure Hash Algorithms

To create a protected cryptographic process, the described algorithm must be trusted, time-examined, and peer-reviewed extensively. A hash function is an algorithm that receives input data and forms a data digest. In this paper, we utilized SHA-2 (SHA-256 and SHA-512). One of the most important reasons for using SHA-2 in our implementation (SHA-256 and SHA-512) is, providing more outcomes (512b and 256b sequentially) than SHA-1 (160b), such that the increased output intensity of SHA-2 is the main reason behind attack defense. Next, present the most vital applications of SHA.

1.3.1 BlockChain Technology

Blockchain technology is an extremely and advanced invention. It empowers digital data to remain distributed but not replicated [19]. Blockchain controls the modern crypto-currency named Bitcoin (digital gold). The expression of "blockchain" indicates structures of data, systems, or networks. It is a listing of ordered blocks, every block includes transactions and communicated to prior one, carrying the hashed value from prior block. Consequently, the transaction history cannot be removed without removing the contents of chain [20]. This is the main reason for saving blockchain from hackers.

Information stored on the blockchain, encrypted by applying HASH functions [22]. Bitcoin utilizes SHA-256. It is one of the most secure functions since every encrypted data give a fully different hash value. The encryption level is a firm such that brute attacks demand various endeavors and still find different input values. Blockchain has principal features as follows [21].

1. Decentralization. Third parties are not needed to confirm activities. Agreement algorithms are employed to keep data on blockchain networks.

2. Persistency. Valid Transactions are quickly, and invalid transactions are not accepted. Therefore, it is infeasible to remove transactions that have already happened.

3. Anonymity: On a blockchain network, the user communicates with others through a produced address. So, the real identification of the user is not represented during the communication.

4. Auditability. Every transaction on the blockchain network indicates the prior transaction. So, each transaction is confirmed easily and followed.

1.3.2 Internet of Things(IoT)

Internet of Things is great employment of the Internet to manage devices that are utilized daily, identified (things) through sensors within the Internet. IOT is defined as a network system of associated various devices (things) that empower us to interact using the protocol of machine-to-machine transmission [23].

Multiple safety vulnerabilities have been identified in the associated devices. Many users have concerned about safety issues, they worry about their data being removed or stolen, or misused [24].Advanced Encryption System (AES-256), SHA-1. SHA-256, etc. are security tools employed in IoT systems to secure the data [25]. IoT is a principle for future Internet development. IoT has managed and the base of emergent technologies like WoT defined as the Web of Things [26]. WOT technology is designed to perform our lives simpler and best. The accelerated growth of IoT led to appear various obstacles, like the vulnerability to cyber-attacks [24, 30]. It is difficult to make safe IoT devices because several security systems are broken to make IoT devices small in size, easy to use, and cheap. One technique that can be arranged to increase the security of IoT is the utilization of blockchain technology [27, 28, 29, 31]. Ronglin Hao et al. [32] an algebraic fault attack on the SHA-256 compression function introduced under the word-oriented random fault pattern. Throughout the attack, the automated Segmentation, Targeting and Positioning (STP) Model is employed, which forms binary representations for the word-based operations in the SHA-256 compression function and then requests a Satisfiability Problem (SAT) solver to resolve the equations. M. Sumathi et al. [33] announced a software framework for the implementation of data security algorithms. AES, RC5 and SHA algorithms have been used in this investigation and examined their implementations in Quartus – II software. They designed the encryption and decryption using Verilog HDL and simulated using ModelSim. With these algorithms, SHA-256 is more cooperative for preparing long data and it produces extraordinary security. The system meets all conditions and the results confirmed its reliability for data transmission. Fırat Artuğer and Fatih Özkaynak [34] offered a new technique to improve the performance of chaos-based substitution box structures. Substitution box structures have a special role in block cipher algorithms since they are the only non-linear elements in substitution permutation network designs. The analysis outcomes explain that the recommended approach can increase the performance standards. The quality of these results is that chaos-based designs may give chances for other applications in addition to the arrest of side-channel attacks.

2. HBCK-HASHING Algorithm

We describe the steps of HBCK – HASHING algorithm, initiated by the step of preparing N-ary block codes (U) as input message to generate square associated matrix of U by using specific notations. Next, we describe a multiplication operation $i \circ j = \beta_{ij}$ towards making HBCK-algebra $(H, \circ, 0)$. Subsequently, we construct N-ary block codes U_H with code words of length q for every HBCK-valued function such that U_H have U inside with redundancy. Moreover, we apply the steps of the secure hash algorithms (SHA-2), starting from the step of Appending bits, Length, and Initialize hash buffer step. Then, divide the message into blocks. Lastly, output the final value as a cipher text.

Step 1. Pre-processing the input N-ary block codes $U=\{d_1,\ldots,d_m\}$. Consider a finite set $L'_n= \{1,2,\ldots, n\text{-}1\}$. After lexicographic order, ascending order U of length q. Let $d_i = d_{i1}, d_{i2}, \ldots, d_{iq}, d_{ij} \in L'_n$ and d_{ij} ordered descending.

Step 2. Constructing the associated matrix $T \in t_r(L_n)$ of hyper BCK-algebra of U. We generate an associated matrix T of N-ary codes U such that $T \in t_r(L_n)$, $r = m + q + 1$. we define the following equation 2.1:

$$\begin{cases} \beta_{s0} = s, \beta_{0t} = 0, s \in \{0, 1, 2, \ldots, r-1\}, \\ \beta_{st} = 0, \text{if } s \leq t, \\ \text{for } q<s \leq r-1, \text{we suppose } \beta_{st} = d_{(q+i)}; \\ \qquad i \in \{1, 2, \ldots, m\}, j \in \{1, 2, \ldots, q\}, \\ \text{for } q<s \leq r-1, q<t<r-1, s>t, \beta_{st} = 1. \end{cases} \tag{2.1}$$

If T is the related matrix of U defined on L'_n and $L_r = \{0, 1, 2, \ldots, r-1\}$ is a non-empty set. Then, by using the previous schemes 2.1, we defined on L_r the following operation $i\circ$ j = β_{ij}.

Step 3. Applying the HBCK-valued function on T to get U_H. We construct N-ary block codes $L_r = \{d_0,d_1,\ldots,d_r\}$ with length q for every HBCK- function such that U_H have U inside with redundancy. Suppose that we have the following:

Finite hyper BCK-algebra (H ,$\circ$,0) with elements (n), finite non-empty set (L) and L_n as a finite set, where $H = \{r_0,r_1,\ldots,r_{n-1}\}, L = \{a_0, a_1,\ldots,a_{m-1}\}$.

The map $f : L \to H$ is a hyper BCK-function, and the generalized function cutted of f is

$$f_{rj} : L \to L_n; r_j \in H, f_{r_j}(a_i) = k$$
$$\Leftrightarrow r_j \circ f(a_i) = (a_i) = \begin{cases} [0, r_k], \\ (0, r_k], \quad \forall r_j, r_k \in H, a_i \in L \\ \{r_k\}. \end{cases} \tag{2.2}$$

$k, j, i \in \{0, 1, 2, \ldots.., n-1\}$.

We suppose $\forall r \in H$, the generalized cut function f_r: L $\to L_n$. Every generalized cut fun, we construct the following code word d_r, with digits belong to the set $L_n$as the following:

$$d_r = d_0, d_1, \dots, d_{m-1}, d_i = j, j \in L_n \Leftrightarrow f_r(a_i) = j; r \circ f(a_i) = \begin{cases} [0, r_j], \\ (0, r_j], \\ \{r_j\}. \end{cases} \tag{2.3}$$

Enlightenment:

1. $(L_r, \circ, 0)$ is a unique identified hyper BCK-algebras since it was obtained by using T, which was a unique identified by U.
2. Let $d_X = \{X_1, X_2, \dots, X_q\}, d_Y = \{Y_1, Y_2, \dots, Y_q\} \in U_H$. We define the relation of order $\leq_c$ on U_H by the following $d_x \leq_c d_y \Leftrightarrow x_i \leq y_i, i \in \{1, 2, \dots, q\}$.
3. On H we define the following:

$$x \circ y = \begin{cases} \theta, & \text{if } x \leq_c y, \forall x, y \in H, \\ (\theta, y], & \text{if } x >_c y, y \neq 0, x, y \in H, \\ \{X\}, & \text{if } y = 0, \\ \{\theta\}, & \text{if } x = 0. \end{cases} \tag{2.4}$$

Where, it gets a hyper BCK-algebra structure.Next steps, we have an exchange between applying SHA-512 or SHA-256, in case of selecting one of them. The following stages concerning applying steps of SHA-512.

Step 4. Appending bits on U_H. It consists of a single 1-bit accompanied by the required amount of 0-bits so that its range is matching to 896 modulo 1024 [range = 896(mod 1024)]. Padding is always added to the N-ary block codes U_H, even if U_H is already of the desired range.

Step 5. Appending length on U_H. A block of 128 bits [unsigned 128-bit integer].

Step 6. Initialize hash buffer. Buffer of 512-bit is utilized to operate in-between and last result of HBCK-HASHING algorithm. Registers of eight 64-bit (a, b, c, d, e, f, g, h) represents the buffer. These records are initialized to the next 64-bit integers (hexadecimal values): a = 6A09E667F3BCC908, b = BB67AE8584CAA73B, c = 3C6EF372FE94F82B, d = A54FF53A5F1D36F1, e = 510E527FADE682D1, f = 9B05688C2B3E6C1F, g = 1F83D9ABFB41BD6B, h = 5BE0CD19137E2179.

Step 7. Divide the message into blocks of 1024-bit with 80 rounds. The module of 80 rounds is identified g. Every round income the input of 512-bit buffer (H_i), and appraises the fillings of the buffer. The value of the eightieth round is joined to the input to the first round (H_{i-1}) to create Hi, the increase is made separately for every of the eight-word in the buffer with each of the similar words in H_{i-1} using addition modulo 264.

Step 8. Output the final desired cipher text.

3. Structure of HBCK-HASHING Algorithm

To understand the proposed HBCK-HASHING algorithm, it is essential to present the model of construction as shown in Figure 1. This model shows the structure of the proposed algorithm through HBCK-valued function as a pre-processing stage that applied on the associated matrix T of the input N-ary block codes U. This function changes the input N-ary block code U to N-ary block codes U_H with redundancy. It aims to maximize the quantity of information with the fewest number of visible characteristics during enlarging the size of U from n×m to r× r with the same length q [35].

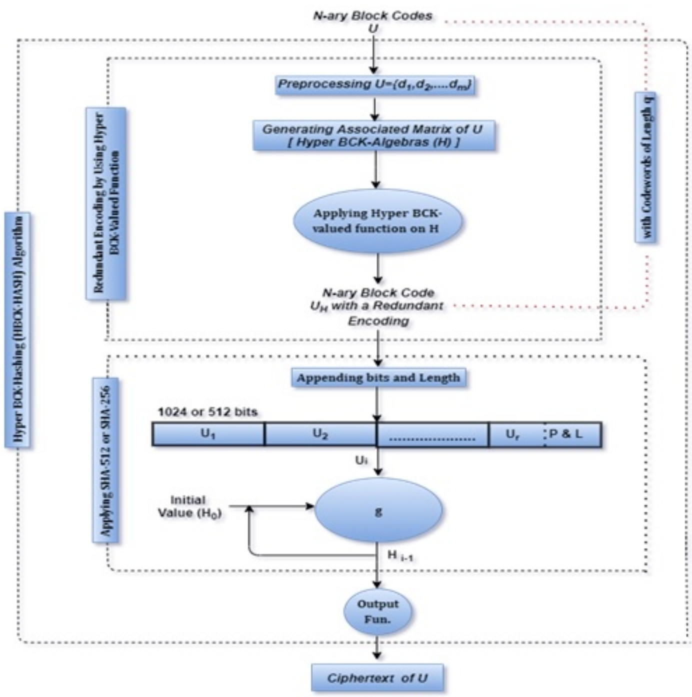

Figure 1: Model of HBCK-HASHING construction

Besides, the structure of the model demonstrates the subsequent steps which including adding padding and length to the N-ary block codes U_H with dividing the U_H into blocks of 1024-bit (in case of using SHA-512) and512-bit (in case of using SHA-256) to get the cipher text value of the N-ary block codes U. The compression Function g, in the construction model, represents

$$g : \{0,1\}^s \times \{0,1\}^{|U_i|} \to \{0,1\}^s. \tag{3.1}$$

Receives an input code H_i (i = 0, ..., r -2) of size S bits and U_i (i = 0, ..., r-1) of size U_i bits, to get the renewed cipher text variable H_i (i = 1, ..., r-1) of size S bits. Consequently, to support the rule of input code of uncertain length, the construction requires padding to transform the input code into a padded code of length a multiple of U_i bits. Simple padding makes unsafe constructed Cipher text. So that, the construction utilizes a padding function, which attaches the value of code length S at the end of U_i to produce the expanded code U_i.

$$H^i = H^{i-1} + g_U^i(H^{i-1}), \tag{3.2}$$

where, g is the compression function of SHA, + is word by word edition mod 264 and H is the cipher text of U.

4. Evaluation parameter

We evaluated the strength of HBCK-HASHING by calculating Avalanche Effect for every N-ary block codes. It has computed over small changes on the plaintext that contains 20 digits. These should provide a meaningful difference in cipher text. Particularly, changing an only bit in the plaintext, fixing the key, should change every bit in cipher text with probability (¿ 50%) ([16]).

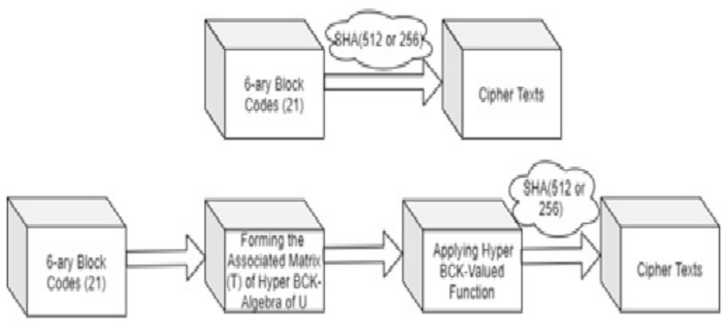

Figure 2: Process of Cryptographing (21) 6-ary block codes

We selected (21) 6-ary block codes with ascending ordered after lexicographic order and descending ordered for bits of each block inside the 6-ary block code. Each block code with4 code words of length 5. i.e., 20 bits in each 6-ary block code. Figure2 shows the process of Cryptographing (21) 6-ary block codes by using SHA-512 or SHA-256 directly, and also with applying HBCK-HASHING to compare the cipher texts and calculate the avalanche effect as an evaluation parameter. We implement HBCK-HASHING on P1 in the case of picking up SHA-512 through constructing a unique identified HBCK-algebra(H)and applying the function of HBCK f: L → H given by

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

to generate 6-ary block codes with a redundant encoding $U_H =$ 00000, 10000, 11000, 11100, 11110, 43221, 53321, 54321, 5431,as stated by step 1, step 2, and step 3 of HBCK-HASHING algorithm. In addition, we implement the Secure Hash Algorithm 512 on U_H to generate the first cipher text of P1. On the same way, we implemented HBCK-HASHING algorithm on another (20) 6-block codes by using the same HBCK-valued function f: L $\rightarrow$ H , and calculated the Avalanche Effect as shown in 4. To calculate the Avalanche Effect of (21) 6-ary block codes, we compared the cipher text of them after applying HBCK-HASHING, as shown in 5 with the cipher texts of (21)6-ary block code after applying the secure hash algorithm 512, as shown in 5, through the division of Number of flipped bits in the cipher text after applying HBCK-HASHING over number of bits in the cipher texts, as shown in 4.

6-ary Block Codes	No. of Flipped Bits in Cipher Texts of 6-ary Block Codes after Applying HBCK-HASHING	No. of Total Bits in Cipher Texts	Avalanche Effect(%)
P1	121	128	94.53125
P2	117	128	91.40625
P3	118	128	92.1875
P4	120	128	93.75
P5	124	128	96.875
P6	117	128	91.40625
P7	119	128	92.96875
P8	124	128	96.875
P9	119	128	92.96875
P10	115	128	89.84375
P11	114	128	89.0625
P12	119	128	92.96875
P13	115	128	89.84375
P14	116	128	90.625
P15	122	128	95.3125
P16	121	128	94.53125
P17	121	128	94.53125
P18	121	128	94.53125
P19	121	128	94.53125
P20	121	128	94.53125
P21	124	128	96.875

Table 1: Value of Avalanche Effect of (21) 6-ary block codes after applying HBCK-HASHING in case of using SHA-512.

we perform HBCK-HASHING algorithm, in the case of picking up SHA-256, on P1 and compared the cipher text of P1 after applying HBCK-HASHING, as shown in 5 with the cipher text of the same 6-ary block codes(P1) after the implementation of SHA-256, as shown in 5. Further, we measured the

Avalanche Effect, as shown in 4. On the same way, we implemented HBCK-HASHING algorithm on another (20) 6-block codes by using the same hyper BCK-valued function f: L → H, and calculated the Avalanche Effect as shown in 4. To calculate the Avalanche Effect of (21) 6-ary block codes, we compared the cipher text of them after applying HBCK-HASHING, as shown in 5with the cipher texts of (21)6-ary block code after applying SHA-512, as shown in 5, through the division of Number of flipped bits in the cipher text after applying HBCK-HASHING over number of bits in the cipher texts, as shown in 4.

6-ary Block Codes	No. of Flipped Bits in Cipher Texts of 6-ary Block Codes after Applying HBCK-HASHING	No. of Total Bits in Cipher Texts	Avalanche Effect(%)
P1	60	64	93.75
P2	60	64	93.75
P3	60	64	93.75
P4	62	64	96.875
P5	58	64	90.625
P6	60	64	93.75
P7	60	64	93.75
P8	60	64	93.75
P9	59	64	92.1875
P10	62	64	96.875
P11	63	64	98.4375
P12	59	64	92.1875
P13	61	64	95.3125
P14	57	64	89.0625
P15	62	64	96.875
P16	60	64	93.75
P17	61	64	95.3125
P18	60	64	93.75
P19	60	64	93.75
P20	60	64	93.75
P21	60	64	93.75

Table 2: Value of Avalanche Effect of 6-ary block codes (U) after applying HBCK-HASHING in case of using SHA-256.

5. Experimental results and analysis

In the following, we have promising results regarding the algorithm of HBCK-HASHING, in case of using SHA-512. 5 shows cipher texts of (21) 6-ary block codes after applying the algorithm and 5 shows cipher texts of (21) 6-ary block codes after applying SHA-512 only.we calculate the Avalanche Effect of (21) 6-ary block codes by computing number of flipped bits in cipher texts, as shown in 4 and representing the values of Avalanche Effect on a graph of (21)6-ary block codes, as shown in Figure 3. We noticed that the maximum value of Avalanche Effect was 96.875% in P5, P8 and P21, where the number of flipped bits in the cipher texts increased to 124, and the least value of Avalanche Effect was 89.0625% in P11, where the number of flipped bits decreased to 114. Addition, the trending line of all values of Avalanche Effect lies between 92% and 94% as shown in Figure 4.

The increasing of Avalanche Effect probabilities lead to increase the security level and the complexity of break through the system.

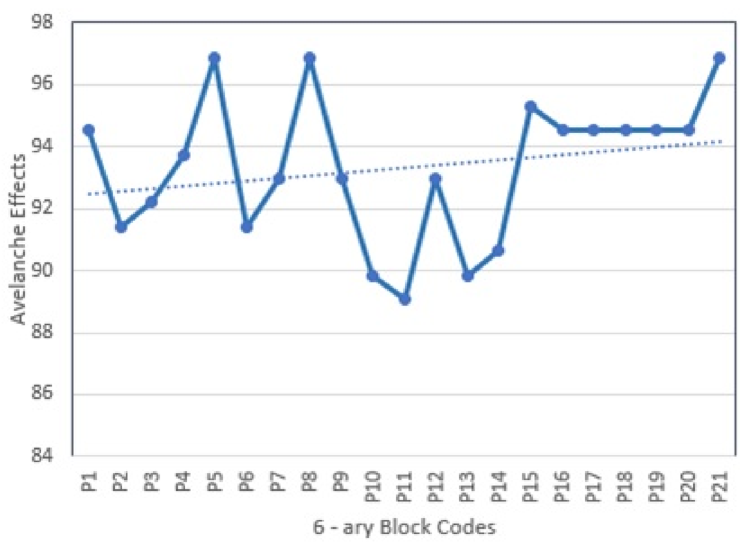

Figure 3: Avalanche Effect of 6-ary block codes (U) after applying HBCK-HASHING in case of using 512

Similarly, in the case of joining SHA-256 with HBCK-HASHING. 5 shows cipher texts of (21) 6-ary block codes after HBCK-HASHING, in case of using SHA-256, and 5 shows cipher texts of (21) 6-ary block codes subsequent implementing SHA-256 only. After computing the Avalanche Effect of (21) 6-ary block codes , as shown in 4 , and representing the values of Avalanche Effect on a graph of (21) 6-ary block codes, as shown in Figure 4. In the case of attaching SHA-256, especially in 4,the highest percentage of Avalanche Effect is 98.4375 in P11, wherever the quantity of flipped bits in the cipher texts following utilizing HBCK-HASHING raised to 63, and the smallest percentage of Avalanche Effect was 89.0625 in P14, wherever the number of flipped bits reduced to 57. In addition, the trending range of all values of Avalanche Effect rest between 93% and 95% as shown in Figure 4.

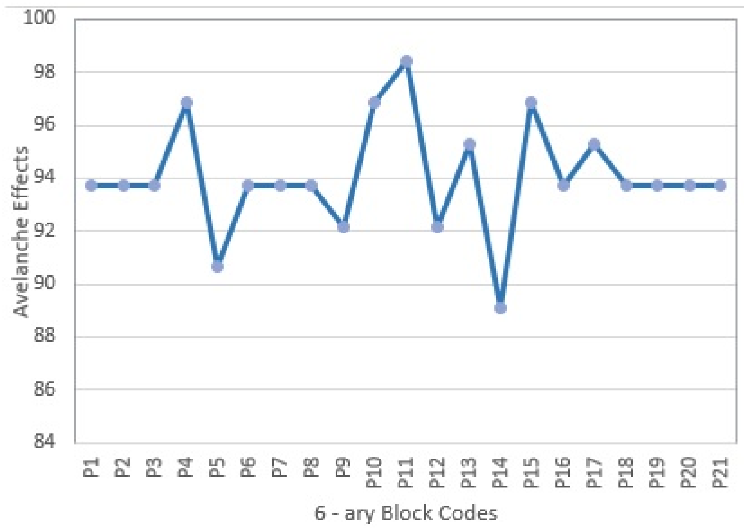

Figure 4: Avalanche Effect of 6-ary block codes (U) after applying HBCK-HASHING in case of using SHA-256.

	6-ary Block Codes (U)	6-ary Block Codes with a Redundant Encoding (U_H)	Cipher Texts of 6-ary Block Codes with a Redundant Encoding by Using SHA (512)
P1	4322153321 5432155431	000001000011000 111001111043221 533215432155431	dc228680e90ec2f6a285518e5ee23e5611b0872 bb20f05d559d524aa1dbf2c474ea259eaa917c74 5cf12c68ec1408f40854e4fbc76cbc7e1e3ffa416 1178463b
P2	**5**322153321 5432155431	000001000011000 1110011110**5**3221 533215432155431	5e779c9152f7af033cec0d01bab8a74954c448bf 43d5c3be58187a1c77c29cb489f3466b95892ff0 43d5c3be58187a1c77c29cb489f3466b95892ff0 8b54e3e03
P3	4**4**22153321 5432155431	000001000011000 11100111104**4**221 533215432155431	a9bab493ff75ff08506e0670a8252065aed839ed 78184a70fee3bd285d0e274b796eb991bf3ef666 58cd262511790e3f928532ada54e4a8e5cbda12 2ae4da427
P4	43**5**2153321 5432155431	000001000011000 111001111043**5**21 533215432155431	b921b1265dd55fdc4e461c4be657afb1bc3c796 b5ccd4678c848b81c96e6dcb691afb50e190043 e1dc504882094a8fc4c1c14aaaa131ab133cb222 73bc0d51af
P5	432**4**153321 5432155431	000001000011000 1110011110432**4**1 533215432155431	182922bf9a7fcdadf82ec275fd0d83586989c78e 58f864e49ecf944eb9c46fdbb4914f574f52eacd2 a6416acdc68b64f442d561f04c59b7476f3def1d 749d45c
P6	4322**5**53321 5432155431	000001000011000 11100111104322**5** 533215432155431	1dc0c82129a07348c123e97c6e4c912c6dd1ea9 53dfbd76a8f9eaee28f31c58394c50d9bbe3ac80 57ac2008b5fef45ff04146343a1671cd2a1c6c2b a608aba86
P7	43221**6**3321 5432155431	000001000011000 111001111043221 **6**33215432155431	5e228a9948c55bb44900c811758ed4bbde9cbd2 1484d16ae706a2993cf1b7dc2304ee182cf76060 85f8bf973e0a44d1646d23ae34c752b9c5140de 1c8498615
P8	432215**4**3321 5432155431	000001000011000 111001111043221	7bf8d3a918bf70eab23308c379e64671468c9dd 102b1fdc401d098f3111550e05e8df82cb373f0d

		5**4**3215432155431	60838c84fd60f7bffe4c9bf625512ee0649687ec
			d2e7fbf08
P9	4322153**6**21	000001000011000	87e21431a04822155db9e3595b2e5998fc52747
	5432155431	111001111043221	9cd4a8c08b3600de59b2ad24f4c6ab55df03b29c
		53**6**215432155431	6bbff88dcc355b3bdbef62e9c8e30d651ba387cc
			8ac3da53c
P10	43221533**4**1	000001000011000	d615749b41bd75d9967accf44e8ca754ed93a47
	5432155431	111001111043221	7a8274c35620f976466ca985893ffbdddfdefcc1
		533**4**15432155431	8440ebd3012c2e0830fbbe158e7a4fc0aec33a05
			df073cb89
P11	432215332**5**	000001000011000	b8795773db70c087ab42aa679724a6e72201b14
	5432155431	111001111043221	1772cb8b4c530f31cd87c3571bf91e3169d617b
		5332**5**5432155431	69d45749c92b7bd599c65bb4765740fee97f0c6
			ac691e7190
P12	4322153321	000001000011000	1a3f80d12198d196290f83653aae952e2bed62a
	3432155431	111001111043221	b5b6f70b6334b5d20c6629e247de28ad657bb87
		53321**3**432155431	54a331f17a2c9e23f2a067c3ec9ba2ead740b022
			9b3828f148
P13	4322153321	000001000011000	726b1e4b5ec8a9f803726336c9784b02cec50ad
	5**5**32155431	111001111043221	72dbd57a13853526b449d8e05c1bba4c779206f
		533215**5**32155431	20bc7a00a79bc36807b0a2dfebd59a0756237c3
			d19fba54f2
P14	4322153321	000001000011000	d7e30f3505cc4281a35666379e4cd11cab617bfc
	54**2**2155431	111001111043221	726f6b705c2cfef33ac18b4723aa5dd330a77c5c
		5332154**2**2155431	8874b1328155a4d2b18007ce7c26683be82f331
			7af6382fc
P15	4322153321	000001000011000	7c9cf687d470ba75efdf63ebd4a3797249f28fce0
	543**5**155431	111001111043221	c9af5a46081e16a7652741ac597f5531ebbafd69
		53321543**5**155431	13d3619895bf6c837e071ac6ecc90538d40c6c3
			185ef084
P16	4322153321	000001000011000	5d37540ddae4f63eb8b2ecc60735e07150dd5cc
	5432**3**55431	111001111043221	5ca2d723514041543d7804af93e5b681978fa4e
		533215432**3**55431	bedf1305fef5d4624804d17039ee52ca67027886
			dfa42e409e
P17	4322153321	000001000011000	3f88ff4f061ffed2caab9226c377bafb8a83c2c92
	54321**6**5431	111001111043221	7487650deeb38c9e0b021c6c0eba016bc0d1d10
		5332154321**6**5431	4e9e0f9ea0c3a6cdeaf4449f58d15368c6cc1393
			bbd822d5
P18	4322153321	000001000011000	51a56ddffaea5df9ab884e07ff5d5ba12591bd2d
	543215**4**431	111001111043221	4883cbdadf0d0a018b0eff4b5799232dfcd92ab4
		53321543215**4**431	0f743ddc615c4c8507fdc84ce67aa83aee7939b6
			48ad5897
P19	4322153321	000001000011000	6d3e924bee5c38e20c01aa7b2a37321821a8a1c
	5432155**3**31	111001111043221	eb353b7b08bf40ab756fc87c24c2eda761c1efcb
		533215432155**3**31	4a0d59bd2676243db2f244f2e0469dc2477a4a5
			56b735b2fa
P20	4322153321	000001000011000	8e9b1debb8fc55df828969595cbd69c1a53571d
	54321554**2**1	111001111043221	138d5f4221c5b2d4416afad15b34504b86a18fd
		5332154321554**2**1	b4921beee6b94673b244f170b4406be9c2a85e5
			6bae6b2936
P21	4322153321	000001000011000	1d6598a24e464832be5da6fb679272c5f35916b
	543215543**2**	111001111043221	b0c612715236275fc4af33cdafb8d46e595f5fa7
		53321543215543**2**	04ec1655448918c4ca3bb1ac1c7d855cf9131a9f
			f49c2a8b4

Table 3: Cipher texts of 6-ary block codes by using HBCK-HASHING algorithm in the case of using SHA-512

	6-ary Block Codes (U)	Cipher texts
P1	4322153321	8dff689bfca583e6734665c695ce8db3163909380af6bbd72d6d716da
	5432155431	ff7f5c5ec9e913ea89b630be957eedbc20246e7ee3d07345d7fc526f81
		49cd72391d73d
P2	**5**322153321	8ac6818ec798fd2511525516b2ebadd6434d485d5fff70b6657befb67
	5432155431	f5c7b1d547e0c07a7225236392729046cd617ea5e1418d12c1b9041a
		e9beb7cfe99f205
P3	4**4**22153321	3ef9b78088b2ac2109f363fd3c81e1bb7d413c1c4055b72ffce42e772
	5432155431	239c916bc5a151ebc37822da23e741be300529152703d81a484f8b11
		acdddc653518e1c
P4	43**5**2153321	872f5ade38a10c8998e07bf29556ebbb239e4cb4e5f3f2c09a30b4f2f0
	5432155431	7443daa26abe1f0cce4b5c360c68c53db221231b2c95a10294225047
		46b82a27fbe73d
P5	432**4**153321	56dd4f42da49f779b73faec92f62fe23556d21e70376f08cfa390c1fd1
	5432155431	3446b5205c75a29351508778512e06fe53373913cdffcc9899633f983
		9c4384417bfcb
P6	4322**5**53321	d0e0c0ab89bce182aee4736f053d013ce209911bce82d8810b9812c7
	5432155431	d308e5e2cc464ae241c898e22cc6a7a8f9d10f2ea4724ccb36b0cd344
		74175e6177e98c2
P7	43221**6**3321	4f26287098b573d4e56c8dfba84afe1d778d5432939b9ec89cd7629bb
	5432155431	bbfb4026f998122ad8e1435acffc496ce828ee2f4b6eccbb09f260ed76
		e6a7d0eebc79d
P8	432215**4**321	ecaeaf191c7375c85982365a31a4544215d2a18cb1f6a52695fa8b0c8
	5432155431	2a9213e00a36303fc17221e4c6121a8a168c5ba642144c400b653e1fc
		a6644019b5c39f
P9	4322153**6**21	2ec54eb37462d87c01608e009b460aed68aa243e5cd41d0fb807d05fb
	5432155431	3dde000a3da2ab02be006dfb2c2b9592bc0981c0ac9ae599d26bf0b1a
		cbc084f97b5505
P10	43221533**4**1	6ced46ed192444df99e8def15733ef9f1daea1107037fb0d184045765
	5432155431	49a9e7c1adbb5dd97898fda9744ec732a226aca533bd4b2da9a281f5d
		cdc97649f4b35f
P11	432215332**5**	dc99a1583fb7f1eca60212a4aadc6cd2a2064904636c93b8acfc6abd4
	5432155431	25e6f31492d216bcbf0425ec51b2bc524486e096d6e5506bac8e7d4c9
		2427cb4a881fa7
P12	4322153321	bce45b51b6b57c003815439c1ceb938df4fdc4ef565ddb01211b045f5
	3432155431	53cb364d85e986a4cfab9d30c405403c816ba37935b6cf77412397f6c
		dc64f431fe0387
P13	4322153321	b201af458c161044b203fe38ad2be39fb649a0943c2e43e65b2f1cf8b
	5**5**32155431	d277d77eb85146220b00a36bc8e726560a6e804e046f331f79aa6533
		8c5829175ce22fc
P14	4322153321	dffb5af0dc513e321caef73feeede9fc7204420b278b4365f70addb7abe
	54**2**2155431	1b1471fa4e508f733eb3cdb161ea84c40b8f41c4e58c6651541353e2d
		089d7212d780
P15	4322153321	37f794ec81ff5963f329ab167a6e2a9210fdd77ac5222bcf3e578a8f14
	543**5**155431	0816bebb5efbb32dd7f0bf84498d9eba0cfbf3def9512eed9b0d82303e
		3c2a23ec9819
P16	4322153321	9a762eba83a1249a2842ca5b668a26e9522a3118fef745c29b716e3db
	5432**3**55431	c37402842e7ea8ec6324ccabc2c41386a459563452a6d5c1af831c84a
		cd26d3f5a032f6
P17	4322153321	3b2e53f31d329878685ccc9a11972dc65fad1b872d520b1d10ba499f9
	54321**6**5431	2a54fbafdc879ec135f041f936ad3b3a5f518acfb780441b99527e183b
		4ca2cb6b66b62
P18	4322153321	d2a5b248e9aec7ad19a100832f3555d8accb70f98b2befb092e2def9aa
	543215**4**431	a90e2d501f6b25ed6275d573e36b9700ab133fd860d87bf563b31cc7
		50d67acfca620f
P19	4322153321	5e231249bad007f2008330441281f8d7c4e5f212a6b22be64948470b
	5432155**3**31	a1147cc33dd4849e1278c8ca214aa5e13d2daabd5c5d8e75ffefdf7335
		390205e1a99139
P20	4322153321	2c958aa35c81a73ceaa79fe89e9326099e7643289646a8638f6045156
	54321554**2**1	775f5cd2ccd0018b742aba1c8151c6bfc91458f1d9f15b1b23da4c96b

P21	4322153321 543215543**2**	2e2f78046baf77 025de2e020e7f67796e31c417cd6921416620c4d0fb31815883c98e5f f8e633655e6e74f8246811a9c591f2aceab9b9219175dd5bd738f578c 69e4a1f4a75d6a

Table 4: Cipher texts of 6-ary block codes by using SHA-512

	6-ary Block Codes (U)	6-ary Block Codes with a Redundant Encoding (U_H)	Cipher Texts of 6-ary Block Codes with a Redundant Encoding by Using SHA (256)
P1	4322153321 5432155431	000001000011000 111001111043221 533215432155431	c443e90b301d4b313ed9f15135550a8f52cfcd1 e1f271b50df3ee20651c39c31
P2	**5**322153321 5432155431	000001000011000 11100111105**5**3221 533215432155431	1a5f25ab34f89787fb650b632d523969b889967 6ff67e0c9545bb1b211e9f90b
P3	4**4**22153321 5432155431	000001000011000 1110011110**4**4221 533215432155431	8e24467b41ec0b64ae9301c90d97d0b6a1ba4f1 a234b9c4e9898cbff8a56fc7d
P4	43**5**2153321 5432155431	000001000011000 1110011110 43**5**21 533215432155431	15e571900d38fe7cefd68226891c3ee95067253 c975e831aea6201c85d3a44f1
P5	432**4**153321 5432155431	000001000011000 11100111104 32**4**1 533215432155431	6579148a6c847e16ccfd06cb4d6aaa2e117dac6 97e18dcade7e9c52f7cd0efc4
P6	4322**5**53321 5432155431	000001000011000 11100111104322**5** 533215432155431	b56ce6bf6f383011285fb14170553112108d94f 8a836a325cf2a29f6f20a46a7
P7	43221**6**3321 5432155431	000001000011000 111001111043221 **6**33215432155431	b5ba82dbefb13902140c9f29214defc17d5d34b f4e5c202f053e72a54eb1687d
P8	432215**4**3321 5432155431	000001000011000 111001111043221 5**4**3215432155431	e8b36d7c026609584b524b1af624e3c2d43ea8a 29dd28eb03ad9046744a36bd1
P9	4322153**6**21 5432155431	000001000011000 111001111043221 53**6**215432155431	638371974ee28e81a6dcede48c739111dcc41ef 4946de646615b15dda4206fbd
P10	43221533**4**1 5432155431	000001000011000 111001111043221 533**4**15432155431	edebbc31e7d02f45d459fdedf6ab6cfcf47e4761 7d47b8c7a3ad5d1d37f968d1
P11	432215332**5** 5432155431	000001000011000 111001111043221 5332**5**5432155431	fbebdf9b610ca90e3e47d6098557f3d623b34e3 9f4d9d7c5699cbcc8e0157d49
P12	4322153321 **3**432155431	000001000011000 111001111043221 53321**3**432155431	54a7a28ca927b7c6bbf8df27572d887a7fcd994 436f89543818dc533b96d192f
P13	4322153321 5**5**32155431	000001000011000 111001111043221 533215**5**32155431	4c1ea6391bcb8ae9580b2d9cc4cb9e1c8a85778 b6c3413e0fcb3c42a784544ff
P14	4322153321 54**2**2155431	000001000011000 111001111043221 5332154**2**2155431	e9faf3a45b983203a57a81fd1092447b083ffc0e d8d90cc1e07b785d7c0c701f
P15	4322153321 543**5**155431	000001000011000 111001111043221 53321543**5**155431	0472c022839276a1c8a9c2e06cce663e57a6985 d00163e49de43bb49531f0c68
P16	4322153321 5432**3**55431	000001000011000 111001111043221 533215432**3**55431	cd8034033603933af5d5f15b8d06e06ebc6fef0 65498d115eb90a15ce41b376f
P17	4322153321 54321**6**5431	000001000011000 111001111043221 5332154321**6**5431	d75972576f0b7168e053332a824f8010aaaff16 59592642e2177a87a214171a7

P18	4322153321	000001000011000	c8c49737513ae92869caee869864ad73cd38552
	5432154**4**31	111001111043221	bb0e93a0d9a4c0e278b6cedf6
		53321543215**4**431	
P19	4322153321	000001000011000	7b63c9f8810434bdcc388f1acb0bb80cbd98dd8
	5432155**3**31	111001111043221	8a418ebbae3f12cb104a8899d
		533215432155**3**31	
P20	4322153321	000001000011000	cf1460cf18c4f7cdc94420cf87390133b24bc074
	54321554**2**1	111001111043221	5085c6b01e87130072edaeae
		5332154321554**2**1	
P21	4322153321	000001000011000	75a6a221dbb141d6ff31a9224508c4db28b5f03
	543215543**2**	111001111043221	e4251d6dc6c856632157ed98b
		53321543215543**2**	

Table 5: Cipher texts of 6-ary block codes by using HBCK-HASHING algorithm in the case of using SHA- 256

	6-ary Block Codes (U)	Cipher texts
P1	4322153321	615cdff092a7b8bc9eead08549ea60c12776e2dc9cd9818
	5432155431	c187e29b6f16f685e
P2	**5**322153321	3f8f2609bc1a7c2c1efea2efa82fd744c4f407f312af611cb
	5432155431	e87b3d0f1a0371d
P3	4**4**22153321	247324ce62b2922a4472bcc1e532d74e3cfb029d955a0ff
	5432155431	390ac62f83c22c94e
P4	43**5**2153321	575d73533b24b2d6c1be9d2844d3e42c174e7ee4ba43eff
	5432155431	df1d5839f7594b71f
P5	432**4**153321	9557f68e556c735534fe1631188b540582e4d49a7cdca03
	5432155431	c222377c6f11a530e
P6	4322**5**53321	476451d83f3e15f940687fab0ecffc6ca6664b0809241d4
	5432155431	9e3a1909e57e48376
P7	43221**6**3321	0e69e71b6a96cc120ca694a4d2732e3a0643ef919d56888
	5432155431	d77a4847c09737b9a
P8	432215**4**321	57481836aabf4dbe63dad0e49642bc74d0edaa972baac8e
	5432155431	33777475bd8d095c2
P9	4322153**6**21	52ffbc60a39bddf7ac2e5ad9e2d443b8cff807595df3d6d9
	5432155431	91895a3ba4b3904a
P10	43221533**4**1	92983647186b1f3fee7651fb9fb110553c9df676e5bf5adf
	5432155431	e17964aff4ec2ec1
P11	432215332**5**	53aa730d5d39758401b968b6e8e1e0ce731172fd975eae
	5432155431	58816338ba149698be
P12	4322153321	16ce0b01b68906d484fdf732ad63bf4f05e111e238413c3
	3432155431	31685685a79354db7
P13	4322153321	3b24ca7468d9baf78affb2ad02d0182056e260b5d6c827f
	5**5**32155431	097285a5d741dbf8e
P14	4322153321	d1b2f630399cc0e1cd89d1495957ef9862aec559f236b8c
	54**2**2155431	191122ea57a01ffd5
P15	4322153321	6768bb0ce52c1004fc38d5ceff78245932a78783963fad0
	543**5**155431	003e4db5c8291d345
P16	4322153321	c696a68e8731c21dc5c34356865a79bd25c5788730aefe8
	5432**3**55431	df754f0783c95a90b
P17	4322153321	53d111ae10641bf1ede879fdb569b3c7c1d03768fadf267
	54321**6**5431	0e4e79a496a03f8ff
P18	4322153321	6f0f830a57f359cd03b13874cb9864e646f8dfe2f236315e
	543215**4**431	c7bda9e2c5ea9b0a
P19	4322153321	625d8332bafcc1f8130449a89ac0bd9a040b04181719ec1
	5432155**3**31	6c988ca842ac498bf
P20	4322153321	339f041f012a6b016ae073158c7f5ff15ac594a495abf20d
	54321554**2**1	a142a61721301eed
P21	4322153321	e2a00587de1c3434e014082ce6e2da337daad81b5e91e5
	543215543**2**	5bab430deaf80c0aa1

Table 6: Cipher texts of 6-ary block codes by using SHA-256

References

[1] R. Karri, P. Mishra, *Minimizing the secure wireless session energy*, Journal of Mobile Network and Applications (MONET), 8 (2002), 177-185.

[2] M.A. Ahmed, E.A. Amhed, *Fuzzy BCK-Algebras*, Journal of Applied Mathematics and Physics, 8 (2020), 927-932.

[3] Y.B.Jun, X.L.Xin, *On derivations of BCI-algebras*, Information Sciences, 159 (2004), 167-176.

[4] S. Mostafa, F. Kareem, H.A. Jad, *Intersectional (/alpha, A)-soft new-ideals in PU-algebras*, Journal of New Theory, 13 (2016), 38-48.

[5] L.H. Encinas, *Codes generated by R0-algebra valued functions*, Applied Mathematical Sciences, 9 (2015), 5343-5352.

[6] C. Flaut, *Some connections between binary block codes and Hilbert algebras*, in Recent Trends in Social Systems: Quantitative Theories, Springer International Publishing Switzerland, 2017, 249-256.

[7] S. Mostafa, B. A.B. Youssef, H.A. Jad, *Efficient algorithm for constructing KU-algebras from block codes*, International Journal of Engineering Science Invention, 5 (2016), 32-43.

[8] S. Mostafa, B.A.B. Youssef, H.A. Jad, *Coding theory applied to KU-algebras*, Journal of New Theory, 6 (2015), 43-53.

[9] S. Mostafa, F. Kareem, H.A. Jad, *Brief review of soft set and its application in coding theory*, Journal of New Theory, 33 (2020), 95-106.

[10] A.B. Saeid, C. Flaut, Š. Hošková-Mayerová, M. Afshar, M. K. Rafsanjani, *Some connections between BCK-algebras and N-ary block codes*, Soft Comput, 22 (2018), 41–46.

[11] Y.B. Jun, L. Xin, M.M. Zahedi, R.A. Borzoei, *On hyper BCK-algebras*, Italian Journal of Pure and Applied Mathematic, 8 (2000), 127-136.

[12] A. T. Surdive, N. Slestin, L. Clestin, *Coding theory and hyper BCK-algebras*, Journal of Hyper structures, J. Hyperstructures, 7 (2018), 82-93.

[13] S. Mostafa, M. Abd-Elnaby, B.A.B. Youssef, H.A. Jad, *Algorithm for encoding N-ary block codes by using the hyper function*, Advances in Mathematics: Scientific Journal, 10 (2021), 339-351.

[14] W. Stallings, *Cryptography and network security: principles and practice*, 5th ed. Pearson, 2011, 342-345.

[15] J. Kurose, K. Ross, *Computer networking: a top-down approach featuring the internet 3/E,* Pearson Education India, 2005.

[16] Menezes, L. Bernard, *Network security and cryptography*, Wadsworth Publishing Company Incorporated, 2012.

[17] B.K. Kim, S.J. Oh, S.B. Jang, Y.W. Ko, *File similarity evaluation scheme for multimedia data using partial hash information*, Multimed Tools Appl, 76 (2017), 19649-19663.

[18] M. Bellare, R. Canetti, H. Krawczyk, *Keying hash functions for message authentication*, Annual International Cryptology Conference, Springer, New York, 1996, 1-15.

[19] S. Nakamoto, *Bitcoin: a peer-to-peer electronic cash system*, Manubot, 2019.

[20] X. Xu, I. Weber, M. Staples et al., *A taxonomy of blockchain-based systems for architecture design*, 2017 IEEE International Conference on Software Architecture (ICSA), 2017.

[21] Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang, *An overview of blockchain technology: architecture, consensus, and future trends*, 2017 IEEE International Congress on Big Data (Big Data congress), 2017.

[22] J. L. Carter, M. N. Wegman, *Universal classes of hash functions*, Journal of Computer and System Sciences, 18 (1979), 143-154.

[23] L. Coetzee, J. Eksteen, *The internet of things-promise for the future? An introduction*, 2011 IST-Africa Conference Proceedings, IEEE, 2011.

[24] X. Wang, J. Zhang, E. M. Schooler, M. Ion, *Performance evaluation of attribute-based encryption: toward data privacy in the IoT*, 2014 IEEE International Conference on Communications (ICC), IEEE, 2014.

[25] Katagi, Masanobu, S. Moriai, *Lightweight cryptography for the internet of things*, Sony Corporation, 2008, 7-10.

[26] V. Shirley, R. Pamidi, *Web of things*.

[27] M. Singh, A. Singh, S. Kim, *Blockchain: a game changer for securing IoT data*, 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), IEEE, 2018.

[28] G. Pulkkis, J. Karlsson, M. Westerlund, *Blockchain-based security solutions for iot systems*, Internet of Things A to Z: Technologies and Applications, 2018, 255-274.

[29] Conoscenti, Marco, A. Vetro, J. C. Martin, *Blockchain for the Internet of things: a systematic literature review*, 2016 IEEE/ACS 13th International Conference of Computem Systems and Applications (AICCSA), IEEE, 2016.

[30] Kamble, Ashvini, Sonali Bhutad, *Survey on Internet of Things (IoT) security*, issues 418, solutions, 2018, 2nd International Conference on Inventive Systems and Control (ICISC), IEEE, 2018.

[31] Kshetri, Nir, *Can blockchain strengthen the internet of things?*, IT Professional, 19 (2017), 68-72.

[32] Ronglin Hao, B. Li, B. Ma, L. Song, *Algebraic fault attack on the SHA-256 compression function*, International Journal of Research in Computer Science, 4 (2014), 1-9.

[33] M. Sumathi, D. Nirmala, R. I. Rajkumar, *Study of data security algorithms using Verilog HDL*, International Journal of Electrical and Computer Engineering (IJECE), 2015, 1092-1101.

[34] F. Artuger and F. Özkaynak, *A novel method for performance improvement of chaos-based substitution boxes*, Journal Of Symmetry, 2020.

[35] E.R. Tufte, *The visual display of quantitative information*, Cheshire, CT: Graphics Press, 1983.

Accepted: September 22, 2021