The Application of Logical Operations and Tabular Transformations in the Base Accents of Hash-Function Algorithms

Akbarov Davlatali Egitaliyevich, Umarov Shukhratjon Azizjonovich

Kokand state pedagogical institute, Fergana branch of the Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Fergana, Uzbekistan

sh.umarov81@mail.ru

Abstract

In the article the stages of transformation of the hashing processes on the structure of the general functional model of the algorithm without the key hash function are defined. The foundations of the application of logical operations and table replacements with the construction of models of basic transformations of the algorithm are revealed. The basics of methods and methods for modeling basic transformations have been developed. Their fundamental applications for creation with new transformations without key hash function algorithms are substantiated.

Keywords:

without key algorithm hash functions, logical operations, table replacements, uniform distribution, crypto-resistant keys, main basic transformations, pseudo-random sequence, keys of stages (rounds), intermediate value, hash value.

Introduction

Electronic data is confirmed by an electronic digital signature, this electronic digital signature acquires the status of an electronic document if it has a model of the mechanism for verifying the identity and authentication of the Signer - the author of which has not changed-by ensuring that the data is not denied authorship. Hash-function is an expression from the algorithm of the reflection of compression of electronic data of a finite length in bits or byte units to a previously recorded - fixed small-length Block[1], [2], [3], [4].

Materials and Methods

In modeling hash-function algorithms, the application of tabular replacement accents for elements consisting of logical operands and bit-combinations in which the values in the maple table are evenly distributed is a solution to the issue. In this case, the basis of co-operation in the modeling of Hash-function base reflections is developed, based on their application in the creation of new adaptive algorithms, in the improvement of computational techniques and technologies in accordance with the development [5].

Main Text (Review only)

Iterative method in the construction of hash-functions algorithms takes into account the possibility of creating collisions on the basis of the method of collisions of texts, in order to prevent it, the length of information and the sum of control are added to the end of hash information. For this purpose, the following stages are calculated and hash processes are carried out [4],[5],[6]:

1-step. Add filling bits. Hashed data length is optional, the data length is divided into blocks with 2^k, k=8, 9, …$< \infty$ bits. If the length of the last block is smaller than 2^k bits, then 2^k bits are filled with such a sign as zero or space (“probel”).

2-step. Add information length. The value of 2^k bits of the data length given in Step 1 is attached. The block denoting the length of the information to be determined by the number of bits is generated by calculating the length of the information to be hashed [7].

3-step. Add control sum. 2^k bit block indicating the control sum of the given information is attached. The block denoting the sum of the control is generated by calculating the sum of the values of all blocks in the decimal counting system according to $\mathrm{mod}\, 2^{2^k}$.

4-step. Processing of data into 2^k - bit blocks. The data is divided into 2^k bit blocks. Let the number of these blocks be equal to *N*. These blocks are M_1, M_2, ..., M_N is defined as. In the execution of algorithm accents on the first block of data, $T=M_1$ on the block (based on the RC-4 algorithm or the compression table) K-key is generalized, that is, by taking the *T* – Block as the key, $0 \le S_i \le 255$ is a linear-filled S_i -byte massive elements with 256-byte satisfying the condition that $N\times 2^k$ bit:

$$K(N\times 2^k)=K_1(2^k)...K_N(2^k)$$

In the execution of the algorithm accents on the next blocks, the H_i -intermediate result, which is formed from the implementation of the algorithm accents on the first block, is generated by the support of the multi-valued *S* -block or any one-sided accents, which is the opposite of the $S(H_i)=D$ - Intermediate Result, and the key generalization is defined as $T=D$. Then the processing is carried out by dividing the data into 256-bit blocks, with the main-base reflections. When the key generated from the text that needs to be hashed is unknown, the algorithm is considered desirable that most of the key accents have a one-sided property[3], [4].

The main-base accents of the general functional model in the quality of one of the supporters of the creation of hash-function algorithms without a crypto-resistant key can be expressed as follows [5],[8]:

According to the composition of this general functional model, the accents of the hash process are carried out as follows (Figure 1):

1. expanded text to be hashed $M=M_1M_2...M_n$ is a generation of round keys by blocks (i - the beginning of the cycle on);

2. text block and multiple execution of the base reflection on the generated round keys (according to j-cycle);

3. the result determined by h-variable which determines the value of a variable as a hash value $H_i=h$, $H=H_i$;

4. determining whether this result is an intermediate result or a final result, that is, checking this i<n+2 condition;

5. if there is no intermediate result ($i<n+2$ is not executed), then the hash-value H is determined by going to the next stage, or rather if there is an intermediate result ($i<n+2$ is executed), this result block is reflected without a line (for example, through an S-Block), and the beginning of the process of generating more phase keys from the Here is an intermediate hash-value with the block to be hashed in the queue $*_6=\oplus, *_t$, t=9,10,11,12 is reflected on the basis of a tabular substitution of logical actions or combinations of bits $D=S(H_i)$.

6. hashing process is completed and its value is determined by *H*.

A general functional model of the creation of a hash-function algorithm based on the implementation of logical operations and tabular substitution, in which the values of the main-base accents of the maple table are evenly distributed, is created (Figure 2). The information that needs to be hashed is allocated to bits-blocks, the

incomplete block to bits is filled with an agreed character, for example, a space. The number of these blocks is n. This block is M_1, M_2,..., M_n and M_{ny} the control log is in bit units, M_{mu} the data length is taken in bytes: $M=M_1M_2 \dots M_n M_{ny}M_{mu}$. Enhanced text M is blurred in a serial iterative mode.

T block is the key generalization K(N× 2^k)=K_1(2^k)...K_N(2^k bit) for the implementation of the main logical and tabular accents on the block several times (cycle j-on). For this

$$T = t_1t_2t_3t_4\ t_5t_6t_7t_8\ \text{K}\ t_{249}t_{250}t_{251}t_{252}\ \ t_{253}t_{254}t_{255}t_{256} = = T1T2...T64,$$

$$T\beta = t_{4(\beta-1)+1}(\beta)t_{4(\beta-1)+2}(\beta)t_{4(\beta-1)+3}(\beta)t_{4(\beta-1)+4}(\beta),\ \beta = 1,2,...,64;$$

half bytes are expanded into bytes.

This is a tabular replacement in each column and row from "0" to "15":

1) From column 0- $T1 = t_1(1)t_2(1)t_3(1)t_4(1) = (t_1t_2t_3t_4)_2 = (T1)_{10}$ calculated, this is the number of the row $Q1 = q_1(1)q_2(1)q_3(1)q_4(1) = (q_1q_2q_3q_4)_2 = (Q1)_{10}$ identified, instead of half byte $T1$ bytes are put $q_1q_2q_3q_40000$;

2) From column 1- $T2 = t_5(2)t_6(2)t_7(2)t_8(2) = (t_5t_6t_7t_8)_2 = (T2)_{10}$ calculated, this is the number of the row $Q2 = q_5(2)q_6(1)q_7(1)q_8(1) = (q_5q_6q_7q_8)_2 = (Q2)_{10}$ identified, instead half byte $T2$ bytes are put $q_5q_6q_7q_80001$ ... and so on.

Such a replacement for 16 half-bytes in the queue is also carried out according to the row:

1) From row 0- $T17 = t_{65}(17)t_{66}(17)t_{67}(17)t_{68}(17) = (t_{65}t_{66}t_{67}t_{68})_2 = (T17)_{10}$ calculated, this is the number of the column $U1 = u_1(1)u_2(1)u_3(1)u_4(1) = (u_1u_2u_3u_4)_2 = (U1)_{10}$ identified, instead half byte $T17$ bytes are put 0000 $u_1u_2u_3u_4$;

2) From row 1- $T18 = t_5(18)t_6(18)t_7(18)t_8(18) = (t_{69}t_{70}t_{71}t_{72})_2 = (T18)_{10}$ calculated, this is the number of the column $U2 = u_5(2)u_6(2)u_7(2)u_8(2) = (u_5u_6u_7u_8)_2 = (U2)_{10}$ identified, instead half byte $T18$ bytes are put 0001 $u_5u_6u_7u_8$ ... and so on.

16) From row 15- $T32 = t_{125}(32)t_{126}(32)t_{127}(32)t_{128}(32) = (t_{125}t_{126}t_{127}t_{128})_2 = (T32)_{10}$ calculated, this is the number of the column $U16 = u_{61}(16)u_{62}(16)u_{63}(16)u_{64}(16) = (u_{61}u_{62}u_{63}u_{64})_2 = (Q16)_{10}$ identified, instead half byte $T32$ bytes are put $1111u_{61}u_{62}u_{63}u_{64}$.

In this way $T33$, $T34$, ..., $T64$, half bytes are also exchanged for bytes by column and rows, the 256 bit block will be expanded to a 512 bit block. This expansion can be implemented in 512 bit blocks and can be continued with 1024 bit blocks and so on (table 1).

Table 1

5	13	6	11	1	10	15	8	0	4	7	9	2	12	3	14
8	7	2	14	15	3	11	6	1	12	13	10	5	4	9	0
14	2	13	4	12	7	1	11	6	9	0	5	3	10	8	15
0	14	9	12	3	13	7	4	15	6	5	1	11	2	10	8
3	10	7	2	4	12	9	1	14	13	15	8	0	5	11	6
2	3	1	8	0	14	5	9	12	11	6	7	10	15	13	4
10	4	14	15	9	5	8	2	11	0	1	3	12	6	7	13
11	9	10	1	6	4	13	15	3	5	14	0	8	7	2	12
1	0	3	7	13	11	10	12	9	14	4	6	15	8	5	2
4	8	11	9	14	6	2	5	10	3	12	15	7	13	0	1
9	12	15	0	2	1	14	10	5	8	11	13	4	3	6	7
6	11	8	13	7	9	0	3	4	15	10	2	14	1	12	5
15	1	0	5	10	8	3	7	13	2	9	12	6	14	4	11
12	5	4	10	11	2	6	13	8	7	3	14	1	0	15	9
7	15	12	6	5	0	4	14	2	10	8	11	13	9	1	3
13	6	5	3	8	15	12	0	7	1	2	4	9	11	14	10

By completing the block bits and logical operations that need to be $T=T*_l Kj$, l=6,9,10,12 expression is obtained, and by completing the tabular combinations on the L, more hashed and distribution accents are performed, such as $h=E_{Kj}(T)$ (table 2).

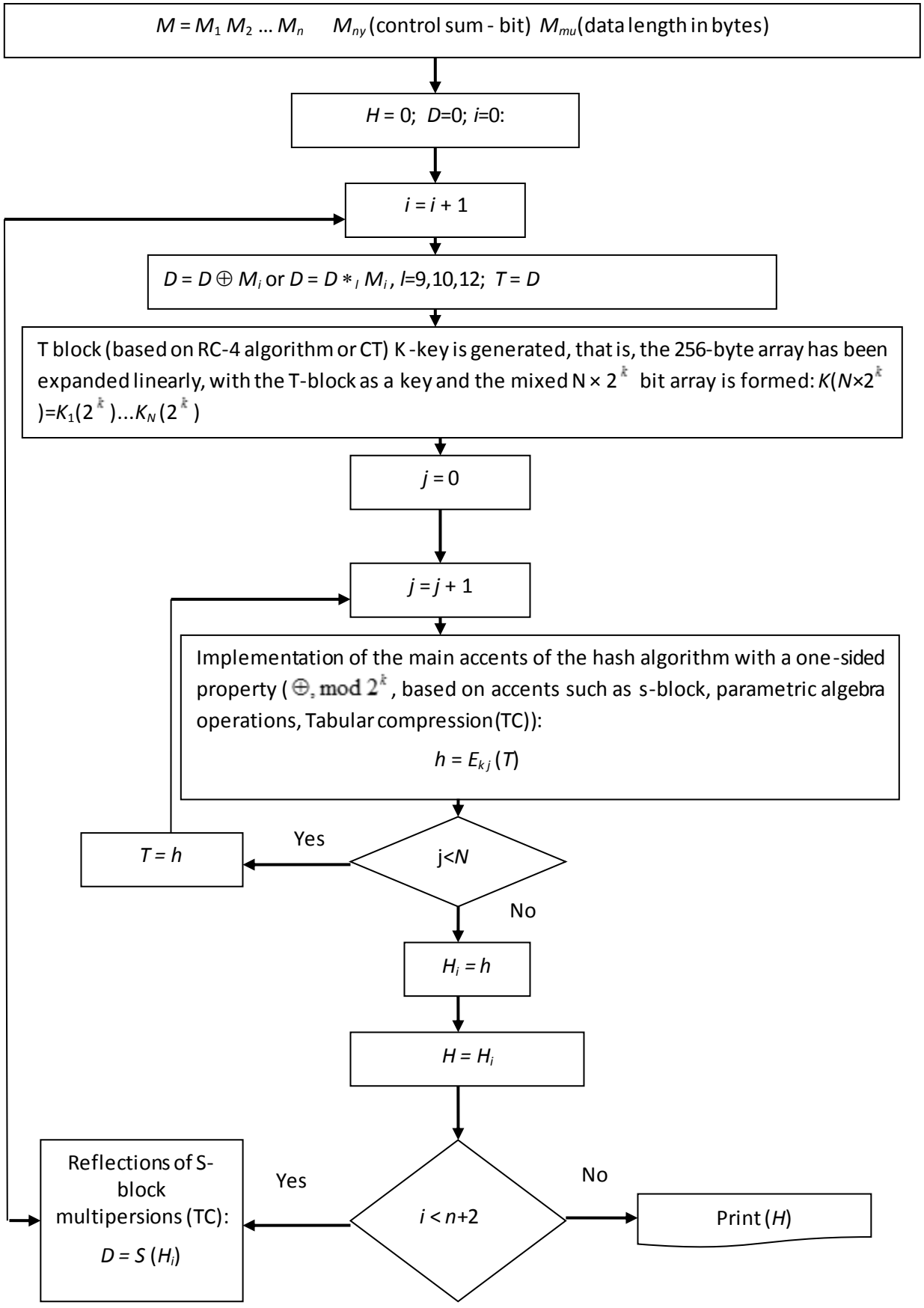
Table 2

$t \oplus k = = t *_6 k$	t/k	0	1
	0	0	1
	1	1	0

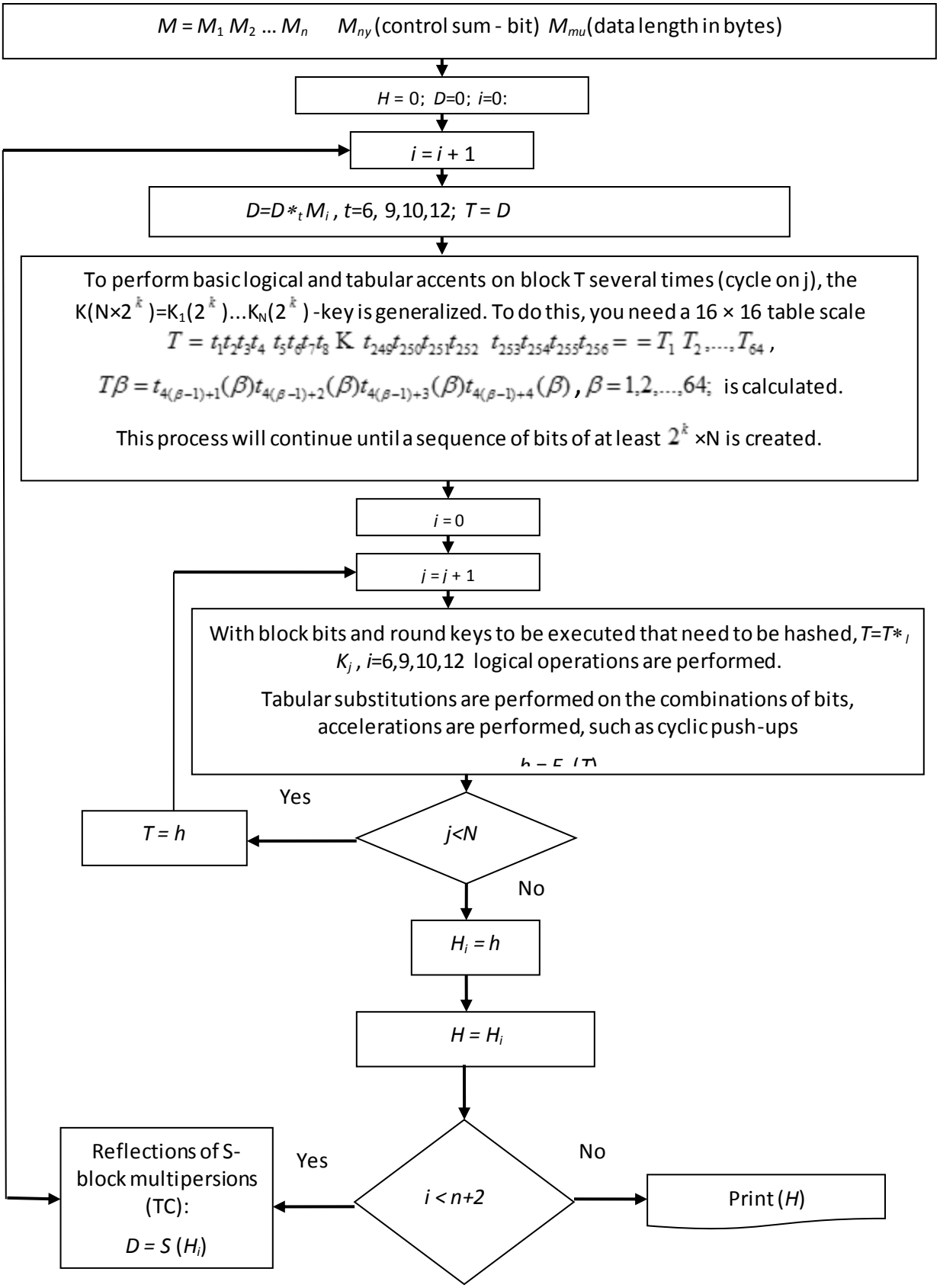
$t *_9 k$	t/k	0	1
	0	1	0
	1	0	1

$t *_{10} k$	t/k	0	1
	0	1	0
	1	1	0

$t *_{12} k$	t/k	0	1
	0	1	1
	1	0	0



1-picture. Functional scheme for creating crypto-stable, keyless hash-function algorithms



2-picture. Functional scheme for the creation of a hash-function algorithm with a flat distributed table substitution Pairs of bits let the tabular substitutions are given in table 3.

Table 3

$x *_{(2\times2)} k$	x/k	00	01	10	11
	00	10	11	00	01
	01	11	00	01	10
	10	00	01	10	11
	11	01	10	11	00

Hashed T block $t_\gamma t_{\gamma+1}$ is carried out with the performance of substitutions on bit pairs $k_\gamma k_{\gamma+1}$ corresponding to bit pairs and key block *Kj*. In this the result of the reflection of the combination of bits on the collar at the intersection of rows $t_\gamma t_{\gamma+1}$ and column $k_\gamma k_{\gamma+1}$ on the exchange table is obtained as $z_\gamma z_{\gamma+1}$.

For three-bit combinations, such as two bit pairs, a swap table can be entered (Table 4). The result of the basic-base reflection is obtained h = E_{Kj}(T) and is followed by steps 3-6.

Table 4

$x *_{(3\times3)} k$	x/k	000	001	010	011	100	101	110	111
	000	110	111	000	001	010	011	100	101
	001	111	000	001	010	011	100	101	110
	010	000	001	010	011	100	101	110	111
	011	001	010	011	100	101	110	111	000
	100	010	011	100	101	110	111	000	001
	101	011	100	101	110	111	000	001	010
	110	100	101	110	111	000	001	010	011
	111	101	110	111	000	001	010	011	100

Conclusions

According to the general functional model of creating a keyless hash-function algorithm, the stages of implementation of hash process mirroring were determined. It was shown the possibility of creating many new keyless crypto-resistant hash-function algorithms with the selection of key-based accents and their combination. The prospect of creating hardware-software and hardware devices related to Information Protection, which is solved by the use of hash-function, as well as the use of logical actions and replacement tables of bit-combinations as the main-base reflections, proved its ownership in economic, scientific and technical activities.

References

1. **Akbarov D.E.** Cryptographic methods of ensuring information security and their application-Tashkent. 2009 – p. 432.

2. **Vostrov G.** Dynamic processes of hash function formation in a system of finite fields = Динамічні процеси формування хеш-функцій в системі кінцевих полів / G. Vostrov, O. Ponomarenko // ELECTROTECHNIC AND COMPUTER SYSTEMS = Електротехнічні та комп'ютерні системи. Науково-технічний журнал. – 2018. – № 28(104). – С. 233-239.

3. **Akbarov D. E., Umarov Sh.A., Madaminov E.** Means of checking necessary conditions - criterion of crypto stability of symmetric block encryption algorithm. \\Descendants of Mohammed al Khwarizmi. Scientific-practical and information-analytical journal. №4(6). 2018 y. p. 49-52.

4. **Акбаров Д.Е., Умаров Ш.А.** Алгоритм хеш-функции с новыми базовыми преобразованиями.\\ Вісник Національного технічного університету України "Київський політехнічний інститут". Серія: Приладобудування. 2016. № 51 (1). С. 100-108. DOI: https://doi.org/10.20535/1970.51(1).2016.78112

5. **Akbarov D.E., Umarov Sh.A., Tojiboyev I.** Applications of logical operations and tablet replacement while creating perfectly perfect wrenches. \\Scientific-technical journal of Fergana Polytechnic Institute. №4. 2018. p. 17-23.

6. **Акбаров Д.Е., Умаров Ш.А.** Новый алгоритм блочного шифрования данных с симметричным ключом. \\Вісник Національного технічного університету України "Київський політехнічний інститут". Серія: Приладобудування. 2016. № 52 (2). С. 82-91.

7. DOI: https://doi.org/10.20535/1970.52(2).2016.92963

8. **Карондеев А. М.** Сложение по модулю 2^n в блочном шифровании \\ ПДМ. Приложение, 2015, № 8, 62–63. DOI: https://doi.org/10.17223/2226308X/8/22

9. **Логачев О. А., Сальников А. А., Ященко В. В.** Булевы функции в теории кодирования и криптографии. М.: МЦНМО, 2004.