A New Modified MD5-224 Bits Hash Function and an Efficient Message Authentication Code Based on Quasigroups

Umesh Kumar and V. Ch. Venkaiah

Abstract In this paper, we have proposed (i) a hash function and (ii) an efficient message authentication code based on quasigroup. We refer to these as QGMD5 and QGMAC, respectively. The proposed new hash function QGMD5 is an extended version of MD5 that uses an optimal quasigroup along with two operations named as QGExp and QGComp. The operations quasigroup expansion (QGExp) and the quasigroup compression (QGComp) are also defined in this paper. QGMAC is designed using the proposed hash function QGMD5 and a quasigroup of order 256 as the secret key. The security of QGMD5 is analyzed by comparing it with both the MD5 and the SHA-244. It is found that the proposed QGMD5 hash function is more secure. Also, QGMAC is analyzed against the brute-force attack. It is resistant to this attack because of the exponential number of quasigroups of its order. It is also analyzed for the forgery attack, and it is found to be resistant. In addition, we compared the performance of the proposed hash function to that of the existing MD5 and SHA-224. Similarly, the performance of the proposed QGMAC is compared with that of the existing HMAC-MD5 and HMAC-SHA-224. The results show that the proposed QGMD5 would take around 2 μs additional execution time from that of MD5 but not more than SHA-224, while QGMAC always takes less time than that of both the HMAC-MD5 and the HMAC-SHA-224. So, our schemes can be deployed in all the applications of hash functions, such as in blockchain and for verifying the integrity of messages.

Keywords Cryptography · HMAC-MD5 · HMAC-SHA-224 · Latin square · MD5 · QGMAC · QGMD5 · Quasigroup · SHA-224

U. Kumar (✉) · V. Ch. Venkaiah
School of Computer & Information Sciences, University of Hyderabad, Hyderabad, India
e-mail: kumar.umesh285@gmail.com

V. Ch. Venkaiah
e-mail: vvcs@uohyd.ernet.in

D. P. Agrawal et al. (eds.), *Cyber Security, Privacy and Networking*, Lecture Notes in Networks and Systems 370, https://doi.org/10.1007/978-981-16-8664-1_1

1 Introduction

These days the need for securing a message has been increasing and with that there has been a tremendous need for new hashing techniques and message authentication codes. In cryptography, two types of hash functions are used: (1) hash function without a key (or simply a hash function) and (2) hash function with a key (or HMAC)

1.1 Hash Function Without a Key

A hash function takes an arbitrary length input message and produces a fixed length hash value, called the message digest or checksum. It detects the integrity of a message which is sent by a sender. The properties of the cryptographic hash function (H) are given in [12, 14]

Various cryptographic hash functions exist in the literature [3, 8]. Of these, MD5 is still a widely used hash function because it is one of the hash functions requiring the least number of operations. Of late, many articles are published showing that the MD5 is not secure because the length of the hash-value is too short. So, it is vulnerable to brute force birthday attacks [15], and a collision can be found within seconds with a complexity of around 2^{24} [18]. It is also vulnerable to pre-image attacks and can be cryptanalyzed using dictionary and rainbow table attacks [5, 19]. Various researchers have analyzed the MD5 algorithm against these attacks and tried to modify it [2, 11]. However, no amendment has yet been proven to be fully effective at resolving the vulnerability and therefore remains a challenge to address the problem against MD5 attacks.

1.2 Hash Function with Key or HMAC

The output of HMAC is used to verify both the authenticity and the data integrity of a message when two authorized parties communicate in an insecure channel. It is also used in Internet security protocols, including SSL/TLS, SSH, IPsec. HMAC uses a hash function (H) and a secret key (k) shared between the sender and the receiver. The properties of the HMAC are given in [12, 14].

The security of the proposed schemes is studied by verifying the basic properties of hash function and message authentication code. It is heartening to note that the schemes not just meet the requirements but rather surpass them. Initially, our schemes start with an optimal quasigroup of order 16. Later on, we would like to use optimal quasigroups of order 256.

The paper is organized as follows: Next section gives a brief overview of quasigroup, optimal quasigroup, and MD5. The proposed algorithm including the QGExp and QGComp operations is discussed in Sect. 3. The performance of the QGMD5 and QGMAC algorithms and its comparison with that of MD5, SHA-224, HMAC-MD5, and HMAC-SHA-224 are discussed in Sect. 4. The security analysis of the proposed QGMD5 and QGMAC is discussed in Sect. 5. The concluding remark is given in Sect. 6.

2 Preliminaries

2.1 Quasigroup

Definition-1:A quasigroup Q=(Z_n, *) is a finite nonempty set Z_n of non-negative integers along with a binary operation '*', satisfying the following properties:

(i) If x,y $\in Z_n$ then x*y $\in Z_n$ (Closure property).
(ii) For $\forall$ x,y $\in Z_n$, $\exists$ unique a,b $\in Z_n$ such that $x * a = y$ and $b * x = y$.

Example 1: Table 1 is an example of a quasigroup of order 3 over the set Z_3={0,1,2}.

Note that for $x = 2$ and $y = 1$, $a = 0$ and $b = 1$ are the unique elements of Z_3 such that $x * a = y$ and $b * x = y$, where $*$ denotes the quasigroup operation of order 3. It is true for all $x,y \in Z_3$.
Observe that in a quasigroup, every element appears exactly once in each row and once in each column. Such a table is also called a Latin square [1]. So, the number of quasigroups is the same as that of the Latin squares and the number of quasigroups increases rapidly with its order [17]. In fact, the number is given by the following inequality [9].

$$\prod_{\ell=1}^{n}(\ell!)^{\frac{n}{\ell}} \geq QG(n) \geq \frac{(n!)^{2n}}{n^{n^2}}, \tag{1}$$

where $QG(n)$ denotes the number of quasigroups of order n. For $n = 2^k$, $k = 4, 8$ the bounds of the number are:

$$0.689 \times 10^{138} \geq QG(16) \geq 0.101 \times 10^{119}, \tag{2}$$

$$0.753 \times 10^{102805} \geq QG(256) \geq 0.304 \times 10^{101724}. \tag{3}$$

Table 1 Quasigroup of order 3

*	**0**	**1**	**2**
0	2	1	0
1	0	2	**1**
2	**1**	0	2

Table 2 Optimal Quasigroup of order 16

$*_2$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	8	0	11	2	9	14	7	6	13	3	15	4	5	10	12	1
1	11	2	8	0	7	6	9	14	15	4	13	3	12	1	5	10
2	2	11	0	8	6	7	14	9	4	15	3	13	1	12	10	5
3	10	5	1	12	3	13	4	15	14	9	6	7	0	8	2	11
4	9	14	7	6	8	0	11	2	5	10	12	1	13	3	15	4
5	0	8	2	11	14	9	6	7	3	13	4	15	10	5	1	12
6	12	1	5	10	15	4	13	3	7	6	9	14	11	2	8	0
7	1	12	10	5	4	15	3	13	6	7	14	9	2	11	0	8
8	14	9	6	7	0	8	2	11	10	5	1	12	3	13	4	17
9	7	6	9	14	11	2	8	0	12	1	5	10	15	4	13	3
10	3	13	4	15	10	5	1	12	0	8	2	11	14	9	6	7
11	6	7	14	9	2	11	0	8	1	12	10	5	4	15	3	13
12	5	10	12	1	13	3	15	4	9	14	7	6	8	0	11	2
13	4	15	3	13	1	12	10	5	2	11	0	8	6	7	14	9
14	15	4	13	3	12	1	5	10	11	2	8	0	7	6	9	14
15	13	3	15	4	5	10	12	1	8	0	11	2	9	14	7	6

2.2 *Optimal Quasigroups*

A quasigroup of order 2^k that consists of a collection of $k \times k$ bits optimal S-boxes is called an optimal quasigroup. Our hash function (QGMD5) uses 4×4 bits S-boxes to form an optimal quasigroup. The description of a 4×4 bits optimal S-box is given in [10]. Various approaches to generate the optimal S-boxes of 4×4 bits are given in [10, 13]. Not all such S-boxes are capable of forming the quasigroups because quasigroup is a mathematical object and has certain properties to be satisfied. We have used 16 S-boxes that are suitable for forming the quasigroup, and these are listed row-wise in Table 2.

2.3 *Brief Description of MD5*

MD5 is the most widely used hash function in cryptography. It is designed based on Merkle–Damgard construction. It takes variable length input (message M) and produces a fixed length 128-bit output (hash-value). Before starting the process, the whole message is divided into 512-bit fixed size blocks. If a message length is not a multiple of 512 bits, then it is padded as given in [16].

Now each 512 bits message block m is divided into sixteen 32-bit words (16 sub-blocks) as $m = m_0, m_1, \ldots, m_{15}$. The algorithm of MD5 has four rounds, and each round has 16 steps making 64 steps in total. These four round functions are defined by the following four nonlinear Boolean functions:

$$\begin{aligned} R_{(1,j)}(x, y, z) &= (x \wedge y) \vee (\neg x \wedge z), & 1 \leq j \leq 16 \\ R_{(2,j)}(x, y, z) &= (x \wedge z) \vee (y \wedge \neg z), & 17 \leq j \leq 32 \\ R_{(3,j)}(x, y, z) &= (x \oplus y \oplus z), & 33 \leq j \leq 48 \\ R_{(4,j)}(x, y, z) &= y \oplus (x \vee \neg z), & 49 \leq j \leq 64 \end{aligned} \tag{4}$$

where x, y, z are 32-bit words and $\wedge$, $\vee$, $\oplus$, and $\neg$ are AND, OR, XOR, and NOT operations, respectively. The $R_{(r,j)}$ is defined as the j^{th} step of round r, $1 \leq r \leq 4$ and $1 \leq j \leq 64$.

3 Proposed Schemes

In this section, we have proposed two schemes based on quasigroup: (i) a new hash function QGMD5: it expands the hash size of MD5 and converts 128 bits into 224 bits and (ii) a new message authentication code named here as QGMAC, which is based on the QGMD5. It expands the MD5-based message authentication code (MAC-MD5) to 224 bits. Both the expansions are done through a series of QGExp and QGComp operations. The underlying structure of both the QGMD5 and the QGMAC is similar. The only difference between the two is that the quasigroup used in QGMD5 is publicly known, while the quasigroup used in QGMAC is a secret key. Figure 1 depicts the workflow of both the QGMD5 and the QGMAC. In these schemes, at first, an arbitrary length message is divided into k fixed size blocks, each of which is 512 bits in size. If the length of a message is not a multiple of 512 bits, then the padding will be required, and it is padded as in the case of MD5 hash function [16]. Observe that each round, except the last round of the last block of MD5, is followed by a QGExp operation that inserts 96 bits and a QGComp operation that deletes 96 bits. The last round of the last block of MD5 is followed by only a QGExp operation. QGExp and QGComp are denoted by $\circledast$ and $\circledcirc$, respectively. Since our proposed schemes use quasigroups of orders 16 and 256, the functioning of QGExp and QGComp operations with these order quasigroups is explained separately in detail.

3.1 Quasigroup Expansion (QGExp) Operation

Let each byte of data be divided into two 4-bit integers. That is, a character (one byte data) x is represented as $x = x_1 x_0$, where x_0 and x_1 are 4-bit integers (hexadecimal

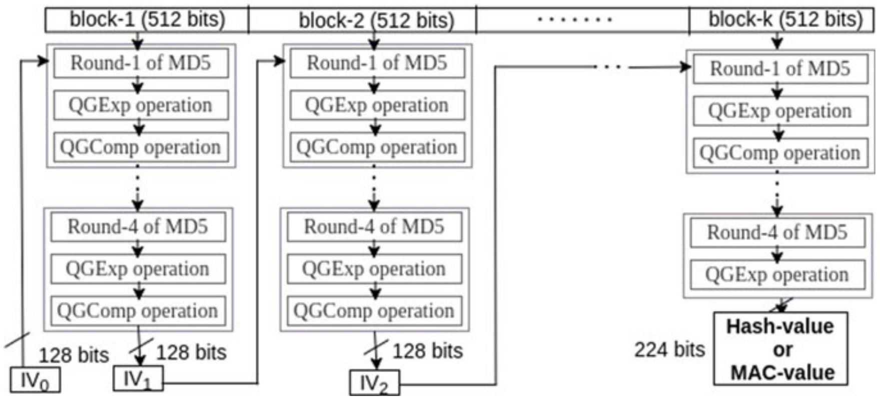

Fig. 1 Workflow of QGMD5 and QGMAC

digits or nibble values). The QGExp operation takes two bytes of data and produces a sequence of three bytes of data. For the quasigroup of order 256, it is defined as

$$x_1x_0 \circledast_1 y_1y_0 = (x_1x_0, y_1y_0, z_1z_0), \tag{5}$$

where z_1z_0=$x_1x_0 *_1 y_1y_0$, and $\circledast_1$ and $*_1$ are the QGExp operation and the quasigroup operation for the order 256, respectively. Note that z_1z_0 is the resultant element which is determined by looking up the element having the row index of x_1x_0 and the column index of y_1y_0 in the table representation of the quasigroup of order 256. And, for the quasigroup of order 16, it is defined as

$$x_1x_0 \circledast_2 y_1y_0 = (x_1x_0, y_1y_0, z_1||z_0), \tag{6}$$

where z_1=$x_1 *_2 y_1$, z_0=$x_0 *_2 y_0$, and $\circledast_2$ and $*_2$ are the QGExp operation and the quasigroup operation for the order 16, respectively, and || is the concatenation operation that concatenates two 4 bits and makes it as one block of 8 bits. Note that z_1 is determined by looking up the element having the row index of x_1 and the column index of y_1 in the table representation of the quasigroup of order 16. Similarly, z_0 is determined by looking up the element having the row index of x_0 and the column index of y_0 in the table representation of the quasigroup of order 16.

An application of the QGExp operation to a pair of sequences of elements is as follows:
Let A=$(a_1^1a_0^1, a_1^2a_0^2, \ldots, a_1^ta_0^t)$ and B= $(b_1^1b_0^1, b_1^2b_0^2, \ldots, b_1^tb_0^t)$, where $a_1^ia_0^i$ and $b_1^jb_0^j$ are byte values whereas a_0^i, a_1^i, b_0^j, and b_1^j are nibble (4 bits) values, for $1 \le i, j \le t$, then

$$(A \circledast_1 B)\ or\ (A \circledast_2 B) = ((a_1^1a_0^1, b_1^1b_0^1, r_1^1r_0^1), (a_1^2a_0^2, b_1^2b_0^2, r_1^2r_0^2), \ldots,$$
$$(a_1^ta_0^t, b_1^tb_0^t, r_1^tr_0^t))$$

where $r_1^j r_0^j = a_1^j a_0^j *_1 b_1^j b_0^j$, $*_1$ is the quasigroup operation of order 256 with respect to the QGExp operation $\circledast_1$ or $r_1^j r_0^j = (a_1^j *_2 b_1^j)||(a_0^j *_2 b_0^j)$, $*_2$ is the quasigroup operation of order 16 with respect to the QGExp operation $\circledast_2$ and $||$ is the concatenation operation.

Similarly if $A = ((a_1^{11}a_0^{11}, a_1^{12}a_0^{12}, \ldots, a_1^{1k}a_0^{1k}), (a_1^{21}a_0^{21}, a_1^{22}a_0^{22}, \ldots, a_1^{2k}a_0^{2k}), \ldots, (a_1^{t1}a_0^{t1}, a_1^{t2}a_0^{t2}, \ldots, a_1^{tk}a_0^{tk}))$ and $B = (b_1^1 b_0^1, b_1^2 b_0^2, \ldots, b_1^t b_0^t)$, where $a_1^{ij}a_0^{ij}$ is a byte value, a_0^{ij} and a_1^{ij} are nibble (4 bits) values for $1 \le i \le t$, $1 \le j \le k$, $b_1^l b_0^l$ is a byte value, b_0^l and b_1^l are nibble (4 bits) values for $1 \le l \le t$, then

$$\begin{aligned}(A \circledast_1 B)\ or\ (A \circledast_2 B) = ((&a_1^{11}a_0^{11}, a_1^{12}a_0^{12}, \ldots, a_1^{1k}a_0^{1k}, b_1^1 b_0^1, r_1^1 r_0^1),\\ (&a_1^{21}a_0^{21}, a_1^{22}a_0^{22}, \ldots, a_1^{2k}a_0^{2k}, b_1^2 b_0^2, r_1^2 r_0^2),\\ &\ldots,\\ (&a_1^{t1}a_0^{t1}, a_1^{t2}a_0^{t2}, \ldots, a_1^{tk}a_0^{tk}, b_1^t b_0^t, r_1^t r_0^t))\end{aligned}$$

where $r_1^j r_0^j = a_1^{jk} a_0^{jk} *_1 b_1^j b_0^j$, $*_1$ is the quasigroup operation of order 256 with respect to the QGExp operation $\circledast_1$ or $r_1^j r_0^j = (a_1^{jk} *_2 b_1^j)||(a_0^{jk} *_2 b_0^j)$, $*_2$ is the quasigroup operation of order 16 with respect to the QGExp operation $\circledast_2$ and $||$ is the concatenation operation.

3.2 Quasigroup Compression (QGComp) Operation

The QGComp operation compresses the partial hash-value (or MAC-value) of 224 bits into 128 bits. The resulting 128 bits are then fed into the next round of MD5 algorithm. The application of QGComp operation can be explained as follows: First it divides the 224 bits (28 byte) into 4 sub-blocks of 7 bytes each. It then operates on each of the 4 sub-blocks as follow: Let A=$(a_1^1 a_0^1, a_1^2 a_0^2, a_1^3 a_0^3, a_1^4 a_0^4, a_1^5 a_0^5, a_1^6 a_0^6, a_1^7 a_0^7)$ be a block of 7 byte, where $a_0^i a_1^i$ is a byte value, for $1 \le i \le 7$. Then, QGComp(A)=$(b_1^1 b_0^1, b_1^2 b_0^2, b_1^3 b_0^3, b_1^4 b_0^4)$, where $b_1^i b_0^i = a_1^i a_0^i *_1 a_1^{8-i} a_0^{8-i}$, $*_1$ is the quasigroup operatin of order 256 or $b_1^i b_0^i = (a_1^i *_2 a_1^{8-i})||(a_0^i *_2 a_0^{8-i})$, $*_2$ is the quasigroup operation of order 16 for $1 \le i \le 3$ and $b_1^4 b_0^4 = a_1^4 a_0^4$.

4 Implementation and Software Performance

The proposed schemes have been implemented in C++ on a system that has the following configuration: Intel(R) Core(TM) i5-2400 CPU @3.40 GHz processor with 4 GB RAM and 64-bit Linux operating system. The source code of QGMD5, MD5, SHA-224, QGMAC, HMAC-MD5, and HMAC-SHA-224 is run 10^3 times for the message M="The brown fox jumps over a lazy dog," and it calculated the average execution time in microseconds (μs). The C++ standard <*chrono*> library is used to

Table 3 Camparison of the average execution time for the message M in microseconds

	Hash functions			Message authentication codes		
Parameters	MD5	SHA-224	QGMD5	HMAC-MD5	HMAC-SHA-224	QGMAC
Avg. Exe. time (μs)	7.94	10.27	9.84	10.12	15.71	9.84

measure the execution time [6]. The performance of QGMD5 is compared with that of both MD5 and SHA-224 and the performance of QGMAC with that of both HMAC-MD5 and HMAC-SHA-224. The results of this analysis are presented in Table 3. Note that the average execution time of the proposed QGMD5 is 1.9 μs more than that of MD5 but not more than SHA-224. Also, note that the average execution time of the proposed QGMAC is always less than that of both HMAC-MD5 and HMAC-SHA-224. This is because the underlying structure of both QGMD5 and QGMAC is the same.

5 Security Analysis

5.1 Analysis of QGMD5

The proposed hash function was analyzed against the dictionary attack by subjecting its output to the online tools such as CrackStation [4] and HashCracker [7]. These tools are basically design to crack the hash-values of MD4, MD5, etc. They employ massive pre-computed lookup tables to crack password hashes. The proposed hash function is also analyzed and found to be resistant to various other attacks, including the brute-force attack. The strength of a hash function against the brute-force attack depends on the length of the hash-value produced by the hash function. The QGMD5 produces 224 bits hash-value instead of 128 bits, as in the case of MD5. Given an n bits hash-value brute-force attack to compute the pre-images (both first pre-image and second pre-image) requires 2^n effort, and to find a collision, it requires $2^{n/2}$ effort, where n is the size of the hash-value. Since the size of the hash-value of QGMD5 is 224 bits as against 128 bits of MD5, QGMD5 can be seen to be more secure than the MD5.

5.2 Collision Resistance

Collision resistance is an important property to test the security of a hash function because the space of messages and that of the hash values are related by a many-to-one mapping. This means different messages may have the same hash-value. For this test, we randomly choose two messages M and $M^{'}$, with hamming distance 1. We compute the hash values h and $h^{'}$ for each pair of messages M and $M^{'}$ and store in ASCII format (ASCII representation is a sequence of bytes in which each byte value lies from 0 to 255), then perform the following experiment [20]: Compare h and $h^{'}$ byte by byte and count the number of hits. That is, count the number of bytes that have the same value at the same position. In other words, compute

$$v = \sum_{p=i}^{s} f(d(x_p), d(x^{'}{}_p)), \text{ where } f(x, y) = \begin{cases} 1, & x = y \\ 0, & x \neq y. \end{cases} \tag{7}$$

The function $d(.)$ converts the entries to their equivalent decimal values and s denotes a number of bytes in a hash-value. Smaller v characterizes the stronger hash function against collision resistance.

Theoretically, for N independent experiments, the expected number of times v hits for an s bytes hash-value is calculated as follows:

$$W_N(v) = N \times Prob\{v\} = N \times \frac{s!}{v!(s-v)!}\left(\frac{1}{256}\right)^{v}\left(1 - \frac{1}{256}\right)^{s-v}, \tag{8}$$

where $v = 0, 1, 2, \ldots, s$. A collision will never happen if $v = 0$, and a collision will happen if $v = s$. For $N = 2048$, we computed, using equation (8), the expected values of $W_N(v)$ for $s = 16$ and $s = 28$ byte hash-values, compared these results with those of the experimental values of MD5, SHA-224, and QGMD5, and tabulated these findings in Table 4. From the entries in Table 4, we observe that the experimental results of QGMD5 not only coincide very well with the theoretical ones but also it has the better collision resistance than that of both MD5 and SHA-224.

Table 4 Results of expected and experimental

	Expected value of $W_N(v)$		Experimental value of $W_N(v)$		
v	$s = 16$	$s = 28$	MD5 ($s = 16$)	SHA-224 ($s = 28$)	QGMD5 ($s = 28$), Pro.
0	1923.69	1835.42	1912	1828	1841
1	120.70	201.54	130	212	199
2	3.55	10.67	6	8	8
$v \geq 3$	0	0	0	0	0

5.3 Avalanche Effect

One of the desirable properties of a hash function is that it should exhibit a good avalanche effect. That is, for a slight change in the input difference, there should be a significant difference in the output of the hash function. The proposed hash function is tested for this property, and the resulting values are compared with those of MD5 and SHA-224. Details of the test are as follows: The message M = "The brown fox jumps over a lazy dog" of 280 bits is randomly chosen, and 280 messages $(M_0, M_1, \ldots, M_{279})$ are generated by changing the i^{th} bit in M and $0 \leq i \leq 279$.

Let $h = H(M)$ be the hash-value of the original message M and $h_i = H(M_i)$ be the hash-values of the messages M_i for $0 \leq i \leq 279$. Since the size of hash-value of MD5 is 128 bits and it differs from that of SHA-224 and QGMD5, the hamming distance h_i from h is measured in percentage using the following formula:

$$HDP_i = \frac{D(h, h_i)}{NB(h)} \times 100\% \tag{9}$$

where HDP_i denotes the hamming distance of h_i from h in percentage for $0 \leq i \leq 279$, $D(h, h_i)$ denotes the hamming distance between h and h_i, and $NB(h)$ denotes the total number of binary digits in hash-value h. Table 5 shows the number of times the hamming distances (HDP_i) of the hash-values $h_0, h_1, \ldots, h_{279}$ from h lie in the specified range for the hash functions MD5, SHA-224, and QGMD5. Also given in the table is the average (mean) of these values. From these values, we can conclude that the avalanche effect of QGMD5 is better than that of both MD5 and SHA-224.

5.4 Analysis of QGMAC

The security of the proposed message authentication code QGMAC depends on the hash function QGMD5 as well as on the quasigroup of order 256 that is used. This is because the quasigroup used in QGMAC acts as a secret key. Since the number of quasigroups of order 256 is lower bounded by 0.304×10^{101724}, it follows that the

Table 5 Hamming distances for MD5, SHA-224, and QGMD5

Range of HDP_i	Number of hash pairs of MD5	Number of hash pairs of SHA-224	Number of hash pairs of QGMD5 (proposed)
35–44.99	41	19	16
45–54.99	206	238	246
55–64.99	33	23	18
	Avarage hamming distance		
Mean:	49.76%	49.97%	50.02%

probability of identifying the chosen quasigroup is close to zero. Hence, QGMAC is resistant to brute-force attack. Also, QGMAC is resistant to forgery attack. In forgery attack, an attacker chooses a fixed n number of different messages $(M_1, M_2, \ldots, M_n)$ and their corresponding MAC-values (authentication tags) $(h_1, h_2, \ldots, h_n)$ and tries to solve the following equations for the key k :

$$h_i = H_k(M_i),\ \ for\ 1 \le i \le n, \tag{10}$$

where, in our case, H is the QGMD5 and k is the quasigroup employed. This is because if the attacker can get the key, then the attacker can forge an authentication tag for any chosen message. But the above system of equations has as many solutions as there are quasigroups of order 256. Hence, determining the quasigroup makes it practically impossible. Therefore, the QGMAC is also resistant to forgery attack.

6 Conclusions

This paper has proposed an efficient method named QGMAC to compute the message authentication code of a message. This method is designed based on the concept called a quasigroup. This QGMAC uses the new hash function, named QGMD5, which is also proposed in this paper. The QGMD5 can be viewed as the extended version of MD5, and it uses the MD5 along with 16 optimal S-boxes of 4×4 bits that form an optimal quasigroup. Because of this, the relationship between the original message and the corresponding hash-value is not transparent. We have analyzed the QGMD5 by comparing it with both the MD5 and the SHA-244, including brute-force attack, collision resistance, and the avalanche effect. We observed that the QGMD5 is more secure than that of both MD5 and SHA-224. Also, the proposed QGMAC is analyzed against brute-force attack and forgery attack. We found that QGMAC is resistant to these attacks.

References

1. Denes J, Keedwell AD (1991) Latin squares: new developments in the theory and applications, vol. 46. Elsevier
2. Farhan D, Ali M (2015) Enhancement MD5 depend on multi techniques. Int J Softw Eng
3. Gupta DR (2020) A Review paper on concepts of cryptography and cryptographic hash function. Eur J Mol Clin Med 7(7):3397–408 Dec 24
4. https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm
5. https://en.wikipedia.org/wiki/Dictionary_attack
6. https://en.cppreference.com/w/cpp/chrono
7. https://www.onlinehashcrack.com
8. Ilaiyaraja M, BalaMurugan P, Jayamala R (2014) Securing cloud data using cryptography with alert system. Int J Eng Res 3(3)

9. Jacobson MT, Matthews P (1996) Generating uniformly distributed random Latin squares. J Combinator Des 4(6):405–437
10. Leander G, Poschmann A (2007) On the classification of 4 bit S-Boxes. In: Proceedings of the 1st international workshop on arithmetic of finite fields. Springer, Berlin, pp 159–176
11. Maliberan EV, Sison AM, Medina RP (2018) A new approach in expanding the hash size of MD5. Int J Commun Netw Inf Secur 10(2):374–379
12. Meyer KA (2006) A new message authentication code based on the non-associativity of quasigroups
13. Mihajloska H, Gligoroski D (2012) Construction of optimal 4-bit S-boxes by quasigroups of order 4. In: The sixth international conference on emerging security information, systems and technologies, SECURWARE
14. Noura HN, Melki R, Chehab A, Fernandez Hernandez J (2020) Efficient and secure message authentication algorithm at the physical layer. Wireless Netw 9:1–5 Jun
15. Paar C, Pelzl J (2009) Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media
16. Rivest R (1992) The MD5 message-digest algorithm. RFC:1321
17. Selvi D, Velammal TG (2014) Modified method of generating randomized Latin squares. IOSR J Comput Engi (IOSR-JCE) 16:76–80
18. Stevens M (2007) Master's Thesis, On collisions for MD5
19. Theoharoulis K, Papaefstathiou I (2010) Implementing rainbow tables in high end FPGAs for superfast password cracking. In: International conference on field programmable logic and applications
20. Zhang J, Wang X, Zhang W (2007) Chaotic keyed hash function based on feedforward-feedback nonlinear digital filter. Phys Lett A 362(5–6):439–448