Chapter IV

CHAPTER - IV

APPLICATIONS OF QUASIGROUPS IN CRYPTOGRAPHY

Cryptography is the field of encrypting information or digital data by using mathematics, computer science and engineering approaches. In today's world, cryptography is also often used to provide security in computer based systems or applications like e-business, e-marketing, e-science, e-government and e-signature.

At this point, two elements of the cryptography: **Encryption and Decryption** techniques have important roles on status of provided security levels. The encryption term can be defined as transforming information or digital data with a special function and the encryption key used by this function. On the other hand, decryption is defined as converting the encrypted information or digital data to its original, unencrypted form. In encryption works, two different encryption-key techniques are widely used. These are named as **"Public-Key Encryption"** and **"Private-Key Encryption"** respectively.

Public-Key Encryption:

Public-key encryption technique can also be called as "asymmetric encryption". In this encryption technique, the user, who wants to ensure a secure communication, needs two different keys. These keys are called as "private key" and "public key" respectively. Each key that the user can use has different roles during the communication. The public key is known by everybody. But the private key is known by only one user. The encryption process is performed by using the public key whereas the decryption process is performed with the private key. At this point, some mathematical equations are used to make the connection between public and private keys.

In other words, encrypted information or digital data can be decrypted by using the private key, which is connected with the public key that was used for encrypting the mentioned information or digital data. Because of this, it is impossible to decrypt the encrypted information or digital data with the help of other private keys. In this technique, it is too important that the user must hide his / her private key from other users. But he / she can share the public key with other people. Figure 4.1 represents a diagram that explains a typical communication session based on public-key encryption technique.

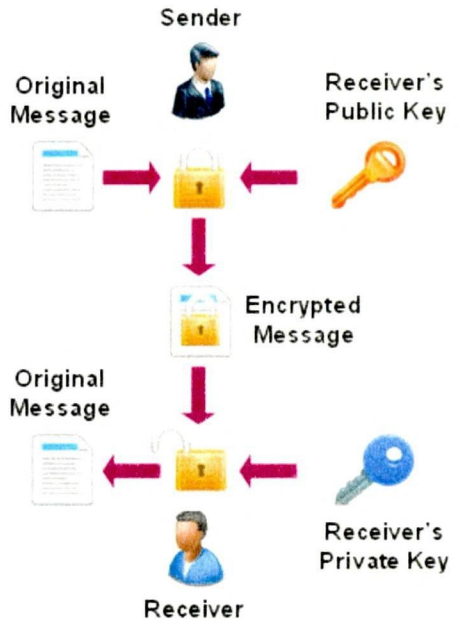


Figure 4.1

Private-Key Encryption:

Private-key encryption technique can also be called as "symmetric encryption". In this encryption technique, only one key is used for encrypting or decrypting the information or digital data. The private-key encryption technique comes with two different approaches: "block encryption" and "row encryption". In block encryption systems, the original message is separated into fixed length blocks and each block is encrypted

individually. In this way, a block is matched with another fixed length block from the same alphabet. In designing of block codes, mixing and diffusion techniques are used and these techniques are applied by using "permutation" and "linear transformation" operations respectively. At this point, strength of the related block encryption algorithm is set by S boxes, number of loops, using keys in XOR operations, block length and key length. Using random key is also another important factor to improve strength of the applied algorithm. The other approach: row encryption is a new form of permutation algorithms which were used in the past. Row encryption technique needs a long key data. Because of this, transition files with feedback feature are used to produce a half-random key. The encrypted message content is created by performing XOR operations with the produced key on the original message. At this point, the receiver must produce the same key in order to decrypt the encrypted message. Figure 4.2 represents a diagram that explains a typical communication session based on private-key encryption technique.

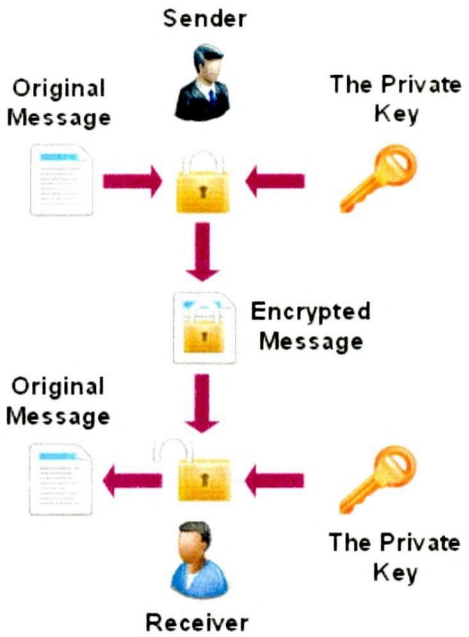


Figure 4.2

SECTION - 4.1

BASIC ENCRYPTION AND DECRYPTION USING QUASIGROUPS

Basic Encryption and Decryption Using Quasigroups:

Suppose that we are given a quasigroup with the operation * as follows:

*	**0**	**1**	**2**	**3**
0	2	0	3	1
1	1	3	0	2
2	3	1	2	0
3	0	2	1	3

Table 4.1.1

The following convention is normally used:

Leader – Any of the symbols of Quasigroup can act as a leader. Here we have four choices: 0, 1, 2, 3 for the leader. The leader decides which row would contribute to the encryption process.

Key – The leader and the quasigroup itself (with the unique arrangements of elements) may be used as keys during the encryption process.

Encryption Scheme:

Let the plain message be represented by $M=(x_1, x_2, ..., x_n)$. We choose a leader L and pass on this secret information as a key to the receiver. Assuming that the quasigroup generation mechanism is available to the receiver, encryption is performed as follows

$$E_L(x_1, x_2, \ldots x_n) = (y_1, y_2, \ldots y_n)$$

where $y_1 = L * x_1$ (1)

and $y_i = y_{i-1} * x_i$ (for all $i > 1$)

For example, if M = (3 0 2 1 2 3 3 1), then $x_1 = 3$, $x_2 = 0$, $x_3 = 2$ and so on. Then, with the Leader as L = 0, the quasigroup given in Table 4.1.1, and the encryption formula given above, we have

$$y_1 = L * x_1 = 0 * 3 = 1$$

$$y_2 = y_1 * x_2 = 1 * 0 = 1$$

$$y_3 = y_2 * x_3 = 1 * 2 = 0 \text{ and so on.}$$

Thus, the encrypted message $E_L(M) = (1\ 1\ 0\ 0\ 3\ 3\ 3\ 2)$.

Since there are various options for choosing a leader, the encryption scheme can be made stronger by using different leaders for encryption of different blocks of the message. The Quasigroup can also be changed periodically by permuting its rows and columns to increase the complexity of encryption scheme.

Decryption Scheme:

For each quasigroup operation '*' we can associate a new quasigroup operation 'o' defined by:

$$x \bullet y = z \text{ iff } x * z = y \qquad (2)$$

The "dual" operation '•' is used for decryption and the "inverse" quasigroup is generated using the above equation:

•	**0**	**1**	**2**	**3**
0	1	3	0	2
1	2	0	3	1
2	3	1	2	0
3	0	2	1	3

Table 4.1.2

Decryption is carried out with the following formula:

$$D_L(y_1, y_2, \ldots y_n) \quad (x_1, x_2, \ldots x_n)$$

where $x_1 \quad L \quad y_1$ (3)

and $x_i \quad y_{i\ 1} \quad y_i$ (for all i 1)

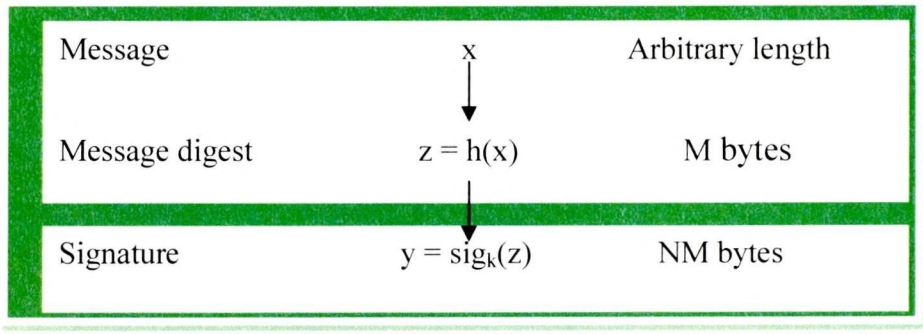
By performing calculations using (3) and the previous quasigroup, the encrypted message is converted back to the original plain message.

$$D_L(E_L(M)) = (3\ 0\ 2\ 1\ 2\ 3\ 3\ 1).$$

SECTION - 4.2

APPLICATIONS OF QUASIGROUPS TO HASHING

Hash functions are a special kind of (public) functions which are used in cryptography for different cryptographic purposes, like signature schemes, message authentication codes, etc. From an input message of arbitrary length, they produce an output message digest of a specified size (M bytes) [**Byte** is a unit of digital information]. The obtained message digest can be used for signing large documents, for checking the validity of packages sent over insecure networks and so on.



Definition: 4.2.1 [59]

A function H ( ) that maps an arbitrary length message M to a fixed length hash value H(M) is a **One Way Hash Function (OWHF),** if it satisfies the following properties:

1. The description of H ( ) is publicly known and should not require any secret information for its operation.

2. Given M, it is easy to compute H (M).

3. Given H (M) in the range of H ( ), it is hard to find a message M for given H (M), and given M and H (M), it is hard to find a message M' ($\neq$ M) such that H (M') = H (M).

Definition: 4.2.2 [59]

A function H ( ) that maps an arbitrary length message M to a fixed length hash value is a **Collision Free Hash Function (CFHF),** if it satisfies the following properties:

1. The description of H ( ) is publicly known and should not require any secret information for its operation.

2. Given M, it is easy to compute H (M).

3. Given H (M) in the range of H ( ), it is hard to find a message M for given H (M), and given M and H (M), it is hard to find a message M' ($\neq$ M) such that H (M') = H(M).

4. It is hard to find two distinct messages M and M' that hash to the same result (H (M) = H (M')).

Definition: 4.2.3 [51]

For a quasigroup $(Q, \cdot)$, a **Hash Function** H_Q ( ) can be defined as

$$H_Q(q_1 q_2 \cdots q_n) = ((\ldots(a \cdot q_1) \cdot q_2 \cdots) \cdot q_n$$

where a is a fixed element from Q.

Quasigroup Transformation QM2:

Let suppose that $a_1a_2\ldots a_m$ be a given string and

$$a_1'a_2'\ldots a_m' = d_l(a_1a_2\ldots a_m),$$

where $l = a_1 \oplus a_2 \oplus \ldots \oplus a_m$. Here $\oplus$ denotes the addition modulo 256. We consider the string,

$$z_1z_2\ldots z_{2m} = a_1a_1'a_2a_2'\ldots a_ma_m'$$

and define the sequence $g_1g_2\cdots g_{2m}$ on the following way:

$$g_i = z_i \oplus (z_{i-1} * z_i), \quad i = 1, 2, \ldots, 2m, \quad \text{where } z_0 = l.$$

We can denote the last operations on a short way as

$$g_1g_2\cdots g_{2m} = z_1z_2\ldots z_{2m} \oplus d_l(z_1z_2\ldots z_{2m}).$$

Note that $z_{2i+1} = a_i$ and $z_{2i} = a_i' = a_{i-1} * a_i$ for $i = 1, \ldots, m$. We want to find another sequence $y_1y_2\ldots y_m$ $(\neq a_1a_2\ldots a_m)$ such that

$$g_1g_2\cdots g_{2m} = y_1y_1'y_2y_2'\ldots y_my_m' \oplus d_{l_1}(y_1y_1'y_2y_2'\ldots y_my_m'),$$

where $l_1 = y_1 \oplus y_2 \oplus \ldots \oplus y_m$. In order to find such a sequence $y_1y_2\ldots y_m$ we have to solve a system of the following kind:

$$\begin{aligned}
&l_1 = y_1 \oplus y_2 \oplus \ldots \oplus y_m \\
&y_1 \oplus (l_1 * y_1) = g_1 \\
&(l_1 * y_1) \oplus (y_1 * (l_1 * y_1)) = g_2 \\
&y_2 \oplus ((l_1 * y_1) * y_2) = g_3 \\
&(y_1 * y_2) \oplus (y_2 * (y_1 * y_2)) = g_4 \\
&\vdots \qquad\qquad (1) \\
&(y_{m-3} * y_{m-2}) \oplus (y_{m-2} * (y_{m-3} * y_{m-2})) = g_{2m-4} \\
&y_{m-1} \oplus ((y_{m-3} * y_{m-2}) * y_{m-1}) = g_{2m-3}
\end{aligned}$$

$$(y_{m-2} * y_{m-1}) \oplus (y_{m-1} * (y_{m-2} * y_{m-1})) = g_{2m-2}$$
$$y_m \oplus ((y_{m-2} * y_{m-1}) * y_m) = g_{2m-1}$$
$$(y_{m-1} * y_m) \oplus (y_m * (y_{m-1} * y_m)) = g_{2m}$$

We can try to find a solution of the previous system starting from the last equation. From all 256^2 possible pairs $(y_{m-1}, y_m) \in A^2$, we have to find all of them which satisfy the last equation of the system. Given any solution (y_{m-1}, y_m) of the last equation, from the equation before the last, we can find y_{m-2} on the unique way. The next equation upward is a control equation for checking if y_{m-2}, y_{m-1}, y_m are well chosen. From the next equation y_{m-3} is uniquely determined, and the equation next upward of it, is a control equation again and so on. Actually, at first we can consider the subsystem:

$$\begin{aligned} &y_i \oplus ((y_{i-2} * y_{i-1}) * y_i) = g_{2i-1} \qquad i = 2, 3, \ldots m \\ &(y_{m-1} * y_m) \oplus (y_m * (y_{m-1} * y_m)) = g_{2m} \end{aligned} \tag{2}$$

where $y_0 = l_1$ is taken to be the leader in the quasigroup transformation.

Choosing the solution (y_{m-1}, y_m) of the last equation, the other unknowns y_i, $i = m-2, m-3, \ldots 1, 0$ are uniquely determined, but we have to check if they satisfy the following equations:

$$\begin{aligned} &y_0 = y_1 \oplus y_2 \oplus \ldots \oplus y_m \\ &y_1 \oplus (y_0 * y_1) = g_1 \\ &(y_{i-1} * y_i) \oplus (y_i * (y_{i-1} * y_i)) = g_{2i}, \qquad i = 1, 2, \ldots m \end{aligned} \tag{3}$$

We note that for any given $g_1 g_2 \cdots g_{2m} \in A$ the system (2) has exactly 256 solutions.

Namely, if we denote $a = (y_{m-1} * y_m)$ and , $b = y_m * (y_{m-1} * y_m)$, the last equation in the system (2) can be rewritten as $a \oplus b = g_{2m}$ and it

has 256 solutions (for each of 256 different values of a, the values of b are uniquely determined). For obtained values of $a = y_{m-1} * y_m$ and $b = y_m * a$, we can compute y_{m-1} and y_m from the quasigroup (A,*) on a unique way. After that y_i for $i = m-2, \ldots, 2, 1$ are uniquely determined. This means that the system (2) has exactly 256 solutions. But, each of them has to satisfy m+2 equations in (3). We can choose m enough large such that the possibility to find another solution of the system (1) different than $a_1 a_2 \ldots a_m$, is enough small. In other words, if m is large enough, then the system of equations (1) has no solutions different than the given one.

Construction of Hash Function Using QM2:

Given message $a_1 a_2 \ldots a_r$, if $r < m$ it can be enlarged to a message $a_1 a_2 \ldots a_r 10 \ldots 0 = a_1 a_2 \ldots a_m$. So, let $r \geq m$. Now, we apply QM2 as follows. Let $j = r - m$ and

$$QM2(a_1 a_2 \ldots a_m) = g_1^1 \ldots g_{2m}^1$$
$$QM2(g_{m+2}^1 \ldots g_{2m}^1 a_{m+1}) = g_1^2 \ldots g_{2m}^2$$
$$QM2(g_{m+2}^2 \ldots g_{2m}^2 a_{m+2}) = g_1^3 \ldots g_{2m}^3$$
$$\vdots$$
$$QM2(g_{m+2}^j \ldots g_{2m}^j a_r) = g_1^{j+1} \ldots g_{2m}^{j+2}$$

Then we define a hash function H_Q by

$$H_Q(a_1 a_2 \ldots a_r) = g_1^{j+1} \ldots g_{2m}^{j+2}$$

which produces message digests of length 2m.

SECTION - 4.3

AUTHENTICATION SCHEMES USING QUASIGROUPS

Definition: 4.3.1

A **Message Authentication Code, or MAC,** is a function h_k, where k is a secret key, with the following properties:

1. Given a value k and an input x, $h_k(x)$ is easy to compute (ease of computation). The value of $h_k(x)$ is usually called the MAC-value of x.

2. For an arbitrary-length input x, $h_k(x)$ is of fixed length n. (compression).

3. Given a fixed number of input-output pairs $(x_i, h_k(x_i))$ for i = 1, 2..., m and any other input $x \notin \{x_1, \ldots, x_m\}$ it is computationally infeasible to compute $h_k(x)$ without knowledge of k (computation-resistance).

MAC is easy to compute with knowledge of the key and difficult to compute without knowledge of the key. Because of this, it is possible that with knowledge of the key, it is very easy to find collisions for the MAC. This situation does not contradict the definition of a MAC.

MACs are used often in cryptography to create a check digit (i.e. an authentication tag) or to authenticate a message. If Bob wants to make sure that the message he receives was really sent by Alice and that this message hasn't been altered or corrupted in transit, he can accomplish this by using a MAC. Alice and Bob simply need to decide on a MAC and exchange the secret key. Alice then computes the MAC-value associated with the message prior to sending it and appends this value to the end of the message. When Bob receives the message, he checks that the appended authentication tag is indeed the correct MAC-value and therefore verifies that the message has come from Alice (or from someone else with knowledge of the secret key)

and that the message has not been changed in transit. If Eve, who does not have knowledge of the secret key, wants to either substitute her own message or alter the message, she will be unable to do so because of the computation resistance property of the MAC. In fact, even if Eve has seen a fixed number of previous messages sent from Alice to Bob with the authentication tags attached, she will still be unable to change the current message or substitute a new one because of the computation resistance property of the MAC. This does require that Alice and Bob change the value of k when a large number of messages have been sent, so as to not give any information away to Eve about the value of k or about how to compute authentication tags.

Denes and Keedwell Scheme: [18]

Denes and Keedwell suggest using a quasigroup $(Q, \bullet)$ and its binary operation to create a MAC. Their scheme is outlined below.

Let $(Q, \bullet)$ be a quasigroup, where $|Q| = q$. Assume that $m = m_1, m_2, \ldots, m_n$ is a message over $(Q, \bullet)$ and assume that we wish to create a signature consisting of $b_0, b_1, \ldots, b_{s-1}$ for m, where $b_i \in Q$ for $i = 0, \ldots, s-1$. Choose t so that $n = st$. We will separate m into s mutually disjoint subsets S_i, for $i = 0, \ldots, s-1$ where each S_i consists of exactly t elements of the message. (If the message length n is not an exact multiple of t, so that we have $n = (s-1) * t + r$ where $0 < r < t$, then create the subsets so that the last subset S_{s-1} contains r message elements instead of t elements.) Assume that $S_i = \{m_{i_1}, m_{i_2}, \ldots, m_{i_t}\}$. Then calculate b_i by

$$b_i = [(\{(m_{i_1} \bullet m_{i_2}) \bullet m_{i_3}\} \bullet m_{i_4}) \bullet \cdots] \bullet m_{i_t}$$

The message and signature are then sent as $m_1, \ldots, m_n, b_0, b_1, \ldots, b_{s-1}$.

Example: 4.3.2

Assume that we would like to create a signature using the method above for the message 1033210322001120 using the quasigroup Q given in Table 4.3.1. Then n = 16 and we choose s = 4, implying that t = 4. Subset S_0 will consist of the 2^{nd}, 8^{th}, 11^{th}, and 13^{th} message elements, because 0 appears in positions 2, 8, 11, and 13 of Q. S_1 will consist of the 1^{st}, 6^{th}, 12^{th}, and 15^{th} message elements, because 1 appears in positions 1, 6, 12, and 15 of Q. S_2 and S_3 can be similarly constructed.

Q =

•	**0**	**1**	**2**	**3**
0	1	0	3	2
1	3	1	2	0
2	2	3	0	1
3	0	2	1	3

Table 4.3.1

Then the signature is created as follows:

$$S_0 = 0,3,0,1 \qquad b_0 = [(0 \bullet 3) \bullet 0] \bullet 1 = (2 \bullet 0) \bullet 1 = 2 \bullet 1 = 3$$
$$S_1 = 1,1,0,2 \qquad b_1 = [(1 \bullet 1) \bullet 0] \bullet 2 = (1 \bullet 0) \bullet 2 = 3 \bullet 2 = 1$$
$$S_2 = 3,0,2,1 \qquad b_2 = [(3 \bullet 0) \bullet 2] \bullet 1 = (0 \bullet 2) \bullet 1 = 3 \bullet 1 = 2$$
$$S_3 = 3,2,2,0 \qquad b_3 = [(3 \bullet 2) \bullet 2] \bullet 0 = (1 \bullet 2) \bullet 0 = 2 \bullet 0 = 2$$

Therefore, the message and signature are sent as 10332103220011203122.

Drawbacks of Denes and Keedwell Scheme:

Denes and Keedwell's scheme has some negative aspects. One of the negative feature of Denes and Keedwell's scheme is that it does not quite

satisfy the requirements for a MAC. In Definition 4.3.1, it is required that a MAC produce a fixed-length output. However, the output from this scheme is $b_0, b_1, ..., b_{s-1}$, which depends on the value of s.

Another negative aspect of this MAC comes from the fact that properties of the quasigroup $(Q, \bullet)$ are not being utilized. This authentication scheme would work if we used a group G instead of a quasigroup $(Q, \bullet)$, especially since the parentheses scheme used to create b_i remains fixed for any set S_i and any quasigroup $(Q, \bullet)$. That is, b_i is created by multiplying all the elements in S_i together from left to right. Because $(Q, \bullet)$ is a quasigroup, $(Q, \bullet)$ could be chosen so that it is non-associative. However, by multiplying from left to right, this non-associativity is not being exploited. Therefore, a new MAC called **Quasigroup Message Authentication Code (QMAC)** was created by K.A.Meyer [48] which relies on the non-associativity of $(Q, \bullet)$ for its security.

Meyer Scheme:

In QMAC, the elements of the message will again be thought of as elements of some quasigroup $(Q, \bullet)$. The secret key for QMAC will be the order in which the message elements are multiplied together to create the MAC-value. If the quasigroup $(Q, \bullet)$ is chosen to be non-associative, then the order in which message elements are multiplied will change the check digit that is created. Essentially, this allows us to have a parentheses scheme as the secret key.

Definition: 4.3.3

Let $(Q, \bullet)$ be a quasigroup.

1. By a **parenthesis scheme** on $x_1, x_2, \ldots, x_t$, we simply mean a choice of parenthesizing $x_1 \bullet x_2 \bullet \ldots \bullet x_t$ so that it forms a well-defined

expression in the non-associative operation ($\bullet$) .

2. A **QMAC key of length** t is a parenthesis scheme on $x_1, x_2, \ldots, x_t$, along with a constant $c \in Q$.

The authentication tag for a message m is then computed by multiplying the message elements together in the order specified by the key, except that every innermost multiplication $(x_i \bullet x_{i+1})$ is replaced by $(x_i \bullet c) \bullet x_{i+1}$.

Example: 4.3.4

Let Q = {0, 1, 2,…,7}and let the Cayley table for $(Q, \bullet)$ be given in Table 4.3.2. Assume that we want to create a check digit for the message m = 146277.

$\bullet$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	5	3	2	1	7	6	0	4
1	0	6	5	4	2	1	3	7
2	2	0	7	6	4	3	5	1
3	7	5	4	3	1	0	2	6
4	6	4	3	2	0	7	1	5
5	3	1	0	7	5	4	6	2
6	4	2	1	0	6	5	7	3
7	1	7	6	5	3	2	4	0

Table 4.3.2

If we choose the key to be the parenthesis scheme $(m_1 \bullet m_2) \bullet (((m_3 \bullet m_4) \bullet m_5) \bullet m_6)$ along with the constant c = 3, then our authentication tag is

$$h_k(m) = ((1 \bullet 3) \bullet 4) \bullet ((((6 \bullet 3) \bullet 2) \bullet 7) \bullet 7)$$

$$= (4 \bullet 4) \bullet (((0 \bullet 2) \bullet 7) \bullet 7)$$

$$= 0 \bullet ((2 \bullet 7) \bullet 7)$$

$$= 0 \bullet (1 \bullet 7)$$

$$= 0 \bullet 7$$

$$= 4$$

and so the message and the authentication tag would be sent as 1462774.

SECTION – 4.4

APPLICATION OF CI- QUASIGROUPS IN CRYPTOGRAPHY

The present method teaches a cryptology system using mathematical structures including crossed-inverse quasigroups or similar mathematical structures.

Cryptosystems often use a mapping, f, from the plaintext message segment, M, to a cipher text segment, C. A parameter E is the cryptographic enciphering "key". Here f is the enciphering algorithm that generates C from M and E. Thus, $C = f(M, E)$.

The message is decrypted according to a mapping g using a decryption parameter. D is the deciphering key, which may or may not be the same as the enciphering key E, and g is the deciphering algorithm that recovers the original message M from the received enciphered message C and the deciphering key D. Hence $M = g(C, D)$. In a stream cipher, these can be "small" functions. The message is divided into segments, M_k, and a stream of key, E_k, is generated. The sender computes the stream $C_k = f(M_k, E_k)$.

The recipient generates the stream D_k and decodes $M_k = g(C_k, D_k)$.

Many times, the cryptological algorithms are made public. The security against unauthorized reception is in the key stream, E_k . The mutual information between M_k and E_k should be sufficiently small that the message cannot be determined by statistical methods. Alternately, the space from which E_k is selected can be too large to be searched.

For a public key system, the objects are from very large sets, e.g., 128 bits which has $2^{128} \approx 3.4 \times 10^{38}$ elements in the set.

The public knows the function f. The encryption key, E, is distributed. A message sender computes and transmits $C = f(M, E)$ and transmits E if it is randomly generated. In this case, the owner of the system also has g and either has D or has an algorithm to generate D from E. The owner then computes $g(C, D)$ to recover M. The security of such a system is based on the difficulty of inverting the function f, without additional information that is known only to the owner.

Nearly all public key cryptosystems are based on finite algebra which is both associative and commutative. The associative property can be described as $(a \bullet b) \bullet c = a \bullet (b \bullet c)$, where $\bullet$ means any associative arithmetic operation for all a, b and c in the algebra. The commutative property also holds that $a \bullet b = b \bullet a$ for all a and b in the algebra.

This method describes an encryption system using a technique which is, in general, not associative or not commutative or neither associative nor commutative. A disclosed mode uses "Crossed-Inverse quasigroups" for encryption.

These aspects of the method will be described with reference to the attached drawings, in which:

FIGURE 4.4.1 shows a circuit used for the encryption,

FIGURE 4.4.2 shows a flowchart of a disclosed mode.

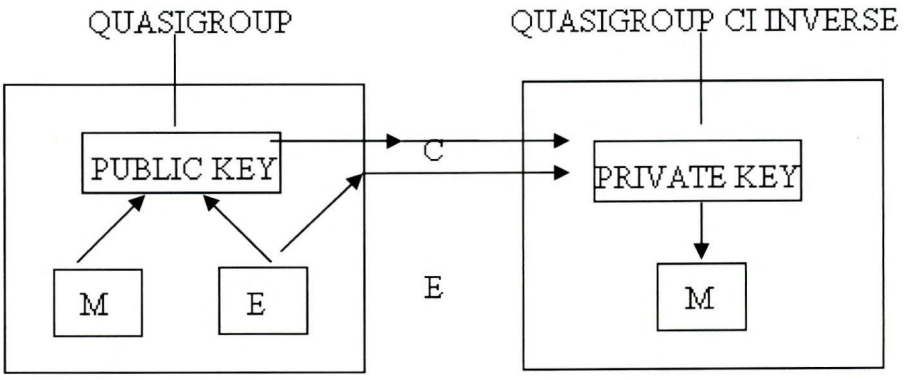


Figure 4.4.1

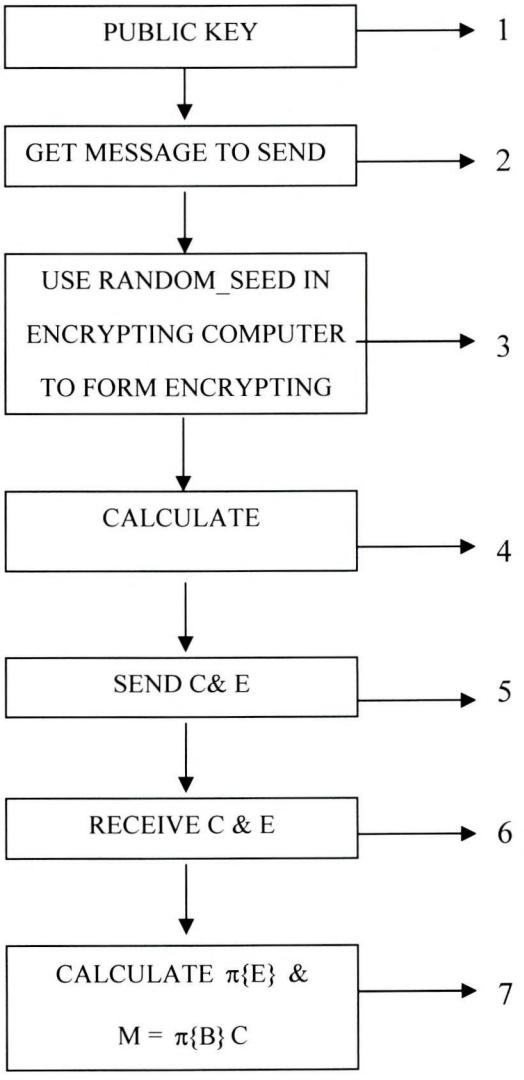


Figure 4.4.2

A Crossed-Inverse quasigroup (CI-quasigroup) has an additional requirement, or property, beyond those described for a quasigroup. For each element a, in the CI-quasigroup, there exist another element a', such that

$$a' \bullet (M \bullet a) = M, \qquad \text{for all M in quasigroup.}$$

The relation between a and a' is a permutation, $a' = \pi(a)$. This permutation is called the CI - permutation. Put a different way, operating on the left with a' undoes the result of having operated on the right with a. That is, to recover M after $(M \bullet a)$ has been computed, $(M \bullet a)$ is operated from the left with a' to get $a' \bullet (M \bullet a) = M$.

A trivial CI -quasigroup is one for which the CI -permutation, π, has the property, $\pi(\pi(x)) = x$. All others are called non-trivial.

Every group is a quasigroup, but not every quasigroup is a group. Hence, groups are a special subset of the universe of quasigroups. Unlike groups, quasigroups are not required to obey the associative rule $a * (b * c) = (a * b) * c$ for all a, b and c. Those quasigroups which are not groups are referred to herein as non-group quasigroups.

Those groups which are also Crossed-Inverse quasigroups must be commutative and have $\pi(a) = a^{-1}$. That is, the CI-permutation in this case is merely the mapping from elements to their multiplicative inverses. If we iterate the mapping, $\pi(\pi(a))$, we get a. That is, two applications of the CI-permutation gives the "identity permutation" when working in a group.

The disclosed mode uses non-trivial, non-group CI-quasigroups as the basis for encryption. The requirement of non-triviality excludes the use of groups, so only quasigroups which are not groups can be used. For use in public-key cryptography, the possessor of the public key publishes an

algorithm to compute $a \bullet b$ in the CI-quasigroup, e.g by distributing mathematical information indicating the Latin Square of the quasigroup. The possessor keeps secret the CI-permutation, B.

The operation is shown and described with reference to FIGURES 4.4.1 and 4.4. 2.

The first step is to find a CI–quasigroup and its CI-permutation, to use as a key. One way to do this is as follows.

Let G be any commutative group of n elements, where n + 1 is a composite number, i.e. n + 1 = rs, where r and s are both integers greater than 1. The group operation in G can be represented by juxtaposition, so that the "product" in G of two of its elements, x and y, is denoted by xy. Define a new operation "$\bullet$" on the elements of G by the relation $a \bullet b = a^r b^s$. Then the elements of G with the operation $\bullet$ form a CI – quasigroup, where the crossed inverse of the element a is a^u, where $u = (-r)^3$. The CI- permutation for this quasigroup will depend on which commutative group is used. The quasigroup Q is distributed as the public part of the key. The CI-permutation is kept secret. For security, the quasigroup Q should have an n at least 10^{10}, and often much larger.

For use in stream cipher, the sender breaks up the message to be sent into fixed size blocks of T symbols each, where a symbol is a short segment of message, $\{M_t : 1 \leq t \leq T\}$, generates the key, $\{E_t : 1 \leq t \leq T\}$, which is often a random session key, and uses the CI– quasigroup to produce cipher text, $\{C_t : 1 \leq t \leq T\}$, as follows: $C_t = M_t \bullet E_t$.

The number of possible symbols must be sufficiently large to make an exhaustive search infeasible. The sender transmits both the key and the

cipher text, (E_t, C_t). The intended receiver knows the CI-permutation. The receiver uses the CI–permutation and the permuted key sequence, $\{D_t = B(E_t) : 1 \le t \le T\}$ to decipher the message as follows: $M_t = D_t \bullet C_t = D_t \bullet (M_t \bullet E_t)$.

The security lies in the distinction between the encipherment key and the decipherment key. An interceptor has C and E. However, solving the equation $C = M \bullet E$ is equivalent to finding the CI-permutation, which is an exceedingly difficult task for large n. That is, for sufficiently large CI-quasigroups, the cryptanalyst will not be able to decode the cipher text without knowing the crossed-inverse of E. The size of the CI-quasigroup is made large enough that it is not feasible to search for D.

FIGURE 4.4.2 shows a flowchart of operation. A "public" key is distributed at **1.** That public key is not necessarily the CI-quasigroup, but only a rule or procedure for calculating "products". The private key is the "quasi inverse" rule, which is not publicly disclosed. The fact that the product rule is that of a CI-quasigroup could also be kept secret if desired.

The user then gets a message to send at **1**, and uses a random seed at **3** to form an encrypting key E. The encrypting key E is used to form cipher text $C = M \bullet E$ at **4,** and C and E are sent at **5.**

As added security, the sender could also be provided with another encryption key E^* which is used to encipher E so that the sender sends $C = M \bullet E$ and $\{E \bullet E^*\}$. The receiver uses $\pi\{E^*\}$ to recover E and $\pi\{E\}$ to recover M. In all of this, the basic principle is that knowing the encryption key does not aid an interceptor's decipherment problem, because the decryption key is different from the encryption key.

Example: 4.4.1

The following example is presented only to illustrate the encryption/ decryption calculations and is not intended as a working system. Let the CI-quasigroup have the following "multiplication table":

•	**0**	**1**	**2**	**3**	**4**	**5**	**6**
0	3	0	6	4	2	5	1
1	2	4	1	0	5	3	6
2	0	3	5	2	1	6	4
3	5	1	4	6	3	2	2
4	1	6	2	5	0	4	3
5	4	2	0	3	6	1	5
6	6	5	3	1	4	0	2

Table 4.4.1

The CI-permutation for this quasigroup is

a	0	1	2	3	4	5	6
B(a)	1	2	3	4	5	6	0

Suppose the sender wishes to send '1' as the message and use encipher key '5'. Then from the table entry with row index 1 and column index 5, we obtain 3, $(1 \bullet 5 = 3)$, so the cipher text is '3'. The sender sends C = 3 and E = 5.

Now the recipient applies the CI – permutation to the encipher key, 5, to get D = $\pi(5)$ = 6. The recipient then computes $D \bullet C = 6 \bullet 3$. This is the entry with row index 6 and column index 3. This entry is '1' $(6 \bullet 3 = 1)$, which is the original message.

SECTION- 4.5

APPLICATIONS OF TERNARY QUASIGROUPS STRING TRANSFORMATIONS

Definition: 4.5.1

For a given quasigroup $(Q,*)$ of order n, $n!-1$ new operations on the set Q, called **parastrophes** can be derived. In this chapter we use one of them known as **left parastrophe** (denoted as "\") defined as follows:

$$x_1 * x_2 = x_3 \Leftrightarrow x_1 \setminus x_3 = x_2 \qquad (1)$$

Example: 4.5.2

Let Q= {1, 2, 3, 4} and $(Q,*)$ is binary quasigroup with the Cayley table given in Table 4.5.1. Using (1) we obtain its left parastrophe $(Q,\setminus)$ given with the Cayley table in Table 4.5.2.

*	1	2	3	4
1	4	1	2	3
2	1	4	3	2
3	2	3	1	4
4	3	2	4	1

Table 4.5.1

Quasigroup $(Q,*)$

\	1	2	3	4
1	2	3	4	1
2	1	4	3	2
3	3	1	2	4
4	4	2	1	3

Table 4.5.2

Parastrophe Quasigroup $(Q,\setminus)$

Quasigroup String Transformations: [44]

Consider an alphabet (i.e. a finite set) Q, and denote by Q^+ the set of all nonempty words (i.e. finite strings) formed by the elements of Q. The elements of Q^+ will be denoted by $a_1a_2...a_n$ rather than $(a_1,a_2,...,a_n)$,

where $a_i \in Q$. Let $*$ be a quasigroup operation on the set Q, i.e. consider a quasigroup $(Q,*)$. For each $a \in Q$ we define two functions $e_{a_*}, d_{a_*} : Q^+ \to Q^+$ as follows.

Let $a_i \in Q, \alpha = a_1 a_2 \ldots a_n$. Then

$$e_{a_*}(\alpha) = \beta = b_1 b_2 \ldots b_n \Leftrightarrow b_1 = a * a_1, b_2 = b_1 * a_2, \cdots, b_n = b_{n-1} * a_n$$

i.e. $b_{i+1} = b_i * a_{i+1}$ for each $i = 0, 1, \ldots, n-1$, where $b_0 = a$,

and

$$d_{a_*}(\beta) = \alpha = a_1 a_2 \ldots a_n \Leftrightarrow a_1 = a * b_1, a_2 = b_1 * b_2, \cdots, a_n = b_{n-1} * b_n$$

i.e. $a_{i+1} = b_i * b_{i+1}$ for each $i = 0, 1, \ldots, n-1$, where $a_0 = a$.

The functions e_{a_*}, d_{a_*} are called e- and d-transformation of Q^+ based on the operation $*$ with leader a, and their graphical representation is shown on Figure 4.5.1 and Figure 4.5.2.

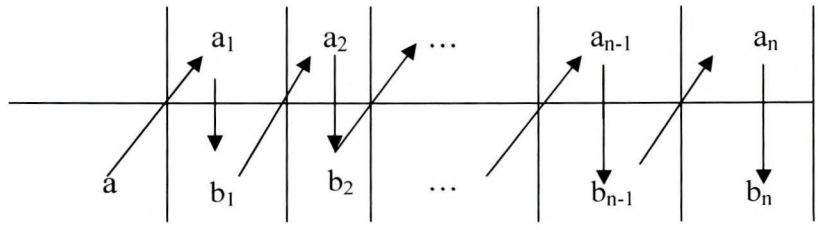


Figure 4.5.1

Graphical representation of e_{a_*} - transformation

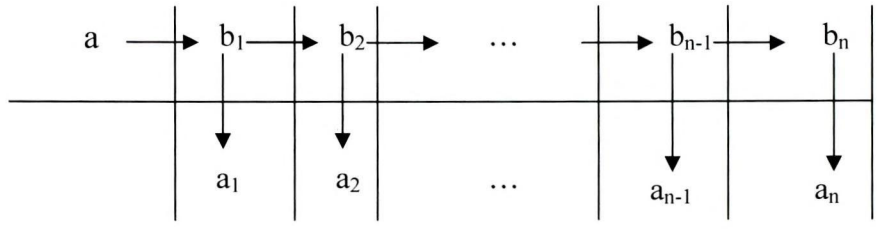


Figure 4.5.2

Graphical representation of d_{a_*} - transformation

Ternary Quasigroup String Transformations (TQST):

Let Q= {1, 2, 3, 4}. We denote by $Q^+ = \{a_1a_2...a_k \mid a_i \in Q, k \geq 2\}$ the set of all finite strings with elements of Q. Let (Q, f) be a given ternary quasigroup consisted from the binary quasigroups $(Q,f_{n_1}),(Q,f_{n_2}),...,(Q,f_{n_k})$ from the bottom to the top. For fixed elements l_1, l_2 from Q, called leaders, we define ternary quasigroup string transformations (which are denoted as TQST) $e_{t_{l_1,l_2},f}$, $d_{t_{l_1,l_2},f} : Q^+ \to Q^+$ as follows:

Let $a_i \in Q, \alpha = a_1a_2...a_n$, for each $i \in \{1,2,...,n\}$.Then

$$e_{t_{l_1,l_2},f}(\alpha) = b_1b_2...b_n \Leftrightarrow \begin{cases} f(l_1,l_2,a_1) = f_{l_1}(l_2,a_1) = b_1 \\ f(l_2,b_1,a_2) = f_{l_2}(b_1,a_2) = b_2 \\ f(b_1,b_2,a_3) = f_{b_1}(b_2,a_3) = b_3 \\ \cdots \\ f(b_{n-2},b_{n-1},a_n) = f_{b_{n-2}}(b_{n-1},a_n) = b_n \end{cases}$$

The ternary function $e_{t_{l_1,l_2}}$ is called $e_{t,f}$-transformation of Q^+ based on the ternary operation f with leader's l_1 and l_2 and its graphical representation is shown on Figure 4.5.3.

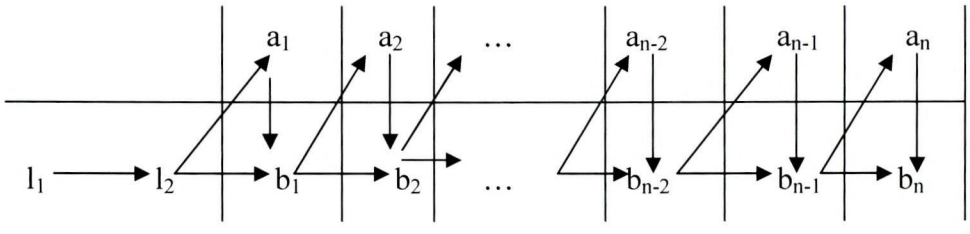


Figure 4.5.3

Graphical representation of $e_{t_{l_1,l_2},f}$ - transformation

$$d_{t_{l_1,l_2},f}(\beta) = a_1 a_2 ... a_n \Leftrightarrow \begin{cases} f(l_1,l_2,b_1) = f_{l_1}(l_2,b_1) = a_1 \\ f(l_2,b_1,b_2) = f_{l_2}(b_1,b_2) = a_2 \\ f(b_1,b_2,b_3) = f_{a_1}(b_2,b_3) = a_3 \\ \dots \\ f(b_{n-2},b_{n-1},b_n) = f_{a_{n-2}}(b_{n-1},b_n) = a_n \end{cases}$$

The ternary function $d_{t_{l_1,l_2}}$ is called $d_{t,f}$ -transformations of Q^+ based on the ternary operation f with leaders l_1 and l_2 and its graphical representation is shown on Figure 4.5.4.

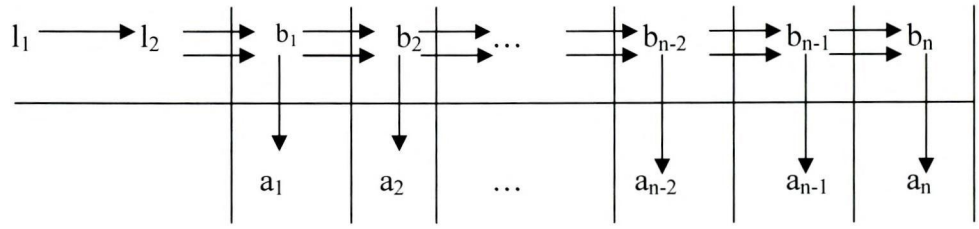


Figure 4.5.4

Graphical representation of $d_{t_{l_1,l_2},f}$ - transformation

We can apply consecutive $e_{t,f}$ or $d_{t,f}$ -transformation on a given string, as a composition of $e_{t,f}$ or $d_{t,f}$ -transformations using the same or different leaders l_1 and l_2 for each transformation. For our research we use the same starting leaders l_1 and l_2 for each transformation.

Let (Q, f) be a given ternary quasigroup consisted of four binary quasigroups (Q, f_1), (Q, f_2), (Q, f_3) and (Q, f_4). If we make parastrophe quasigroups (Q, f_i) for i=1, 2, 3, 4 where f_i' is the left parastrophic operation of f_i defined as $x_1 * x_2 = x_3 \Leftrightarrow x_1 \backslash x_3 = x_2$ for each (Q, f_i) and construct ternary quasigroup from these quasigroups then we will obtain ternary quasigroup (Q, f′) which is parastrophe on given (Q, f) quasigroup.

Example: 4.5.3

Let Q = {1, 2, 3, 4}, (Q, f) is ternary quasigroup given in Table 4.5.3, (Q, f′) is a left parastrophe ternary quasigroup given in Table 4.5.4, $l_1 = 1$ and $l_2 = 2$ and $\alpha = 2134122134221$ is the finite sequence of element of Q.

f_1	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	3	4	2	1
4	4	3	1	2

f_2	1	2	3	4
1	2	3	4	1
2	3	2	1	4
3	4	1	3	2
4	1	4	2	3

f_3	1	2	3	4
1	3	4	1	2
2	4	3	2	1
3	1	2	4	3
4	2	1	3	4

f_4	1	2	3	4
1	4	1	2	3
2	1	4	3	2
3	2	3	1	4
4	3	2	1	4

Table 4.5.3

f_1'	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	4	3	1	2
4	3	4	2	1

f_2'	1	2	3	4
1	4	1	2	3
2	3	2	1	4
3	2	4	3	1
4	1	3	4	2

f_3'	1	2	3	4
1	3	4	1	2
2	4	3	2	1
3	1	2	4	3
4	2	1	3	4

f_4'	1	2	3	4
1	2	3	4	1
2	1	4	3	2
3	3	1	2	4
4	4	2	1	3

Table 4.5.4

Firstly, if $e_{t,f}$-transformation with operation f is applied on the starting sequence α and then $d_{t,f'}$-transformation with operation f' on the obtained sequence β the starting sequence α would be returned which is shown in Table 4.5.5.

l_1	l_2	2 1 3 4 1 2 2 1 3 4 2 2 1	$=\alpha$
1	2	1 2 4 3 2 3 1 3 2 1 3 4 2	$\beta = e_{t,f}(\alpha)$
1	2	2 1 3 4 1 2 2 1 3 4 2 2 1	$\alpha = d_{t,f'}(\beta)$

Table 4.5.5

Encryption and decryption functions

For designing cryptographic primitives $e_{t,f}$-transformation based on operation f is used as encryption function and $d_{t,f'}$-transformation on the operation f' is used as decryption function.