Military Technical College
Kobry El-Kobbah,
Cairo, Egypt

7th International Conference
on Electrical Engineering
ICEENG 2010

A Public Key Cipher Algorithm Based on Multivariate Cubic Quasigroups (MCQ)

By

M. Fawaz* H. Zorkta** S. Alnazer***

***Abstract*:**

A new public key cipher algorithm is introduced in this article. This proposal is based on a specific class of quasigroups string transformations called multivariate cubic quasigroups (MCQ).
MCQ public key cipher algorithm is a public key block cipher algorithm, it is a bijective mapping; it does not perform message expansions and can be used for both encryption and signature. MCQ public key cipher algorithm consists of n multivariate cubic polynomials with $n+d$ variables where $n = d.k : k \geq 40, d = 3$.
A particular characteristic of this proposal is that it is more secure and faster than previous MQQ version in decryption, its encryption speed is comparable to the speed of previous MQQ version, it is highly parallelizable, and it is well suited for short signatures.

***Keywords*:**

Public Key Cryptosystems, Quasigroup String Transformations, Fast signature generation, Multivariate non Linear Polynomials, Multivariate Quadratic Quasigroup, Multivariate Cubic Quasigroup

* M. Fawaz : Master Candidate at Informatics College- Aleppo University- Syria
** H. Zorkta: Asst. Prof. at Network Dept.- Informatics College- Aleppo Unv. Syria
*** S. Alnazer: Head of research Dept. Informatics College- Aleppo Unv. Syria

1. Introduction:

The most popular Public Key Cryptosystem (PKC) schemes are the Diffie and Hellman (DH) key exchange scheme based on the hardness of discrete logarithm problem, the Rivest, Shamir and Adleman (RSA) scheme based on the difficulty of integer factorization, and the Koblitz and Miller (ECC –Elliptic Curve Cryptography) scheme based on the discrete logarithm problem in an additive group of points defined by elliptic curves over finite fields. There are two common characteristics of these well known PKCs (DH, RSA and ECC) [1]:

- their speed – which frequently is a thousand times lower than the symmetric cryptographic schemes,
- their security – which relies on one of two hard mathematical problems: efficient computation of discrete logarithms and factorization of integers.

Recently a new public key scheme called MQQ which is based on multivariate quadratic polynomials and quasigroup string transformations was proposed by Gligoroski et al., This cryptosystem is considered to be of higher potential and expected to be as fast as block cipher [1, 2].

In this paper, a new public key cipher algorithm, based on a specific class of quasigroups string transformations called multivariate cubic quasigroups (MCQ) is introduced. This MCQ public key cipher algorithm is a bijective mapping and can be used for both encryption and signature. It is faster than previous MQQ versions, and it is well suited for short signatures.

2. Preliminaries:

In this section quasigroup string transformations is introduced in 2.1, representation of the quasigroups as vector valued Boolean functions in 2.2, definition and generation of the multivariate cubic quasigroups in 2.3, and a detailed example in 2.4.

2.1. Quasigroup string transformations: [2]

Definition 1. A quasigroup *(Q, *)* is a groupoid satisfying the law

$$(\forall u, v \in Q)(\exists! x, y \in Q) : u * x = v \,\&\, y * u = v. \qquad \textbf{(1)}$$

It follows from (1) that for each $a, b \in Q$ there is a unique $x \in Q$ such that $a * x = b$. Then we denote $x = a \setminus_* b$ where $\setminus_*$ is a binary operation in Q (called a left parastrophe of *) and the groupoid $(Q, \setminus_*)$ is a quasigroup too. The algebra $(Q, *, \setminus_*)$ satisfies the identities

$$x \setminus_* (x * y) = y, \; x * (x \setminus_* y) = y. \qquad \textbf{(2)}$$

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. In this paper, we will use two notifications for the elements of Q^+: $a_1a_2 \ldots a_n$ and $(a_1, a_2, \ldots, a_n)$, where $a_i \in Q$.

Let * be a quasigroup operation on the set Q. For each $l \in Q$ we define two functions $e_{l,*}, d_{l,*} : Q^+ \to Q^+$ as follows:

Definition 2. Let $a_i \in Q$, $M = a_1a_2 \ldots a_n$. Then

$$e_{l,*}(M) = b_1b_2 \ldots b_n \Leftrightarrow$$
$$b_1 = l * a_1,\ b_2 = b_1 * a_2,\ \ldots,\ b_n = b_{n-1} * a_n,$$
$$d_{l,*}(M) = c_1c_2 \ldots c_n \Leftrightarrow$$
$$c_1 = l * a_1,\ c_2 = a_1 * a_2,\ \ldots,\ c_n = a_{n-1} * a_n,$$

The functions $e_{l,*}$ and $d_{l,*}$ are called the *e*–transformation and the *d*–transformation of Q^+ based on the operation * with leader l respectively.

Theorem 1. If $(Q, *)$ is a finite quasigroup, then $e_{l,*}$ and $d_{l,\backslash *}$ are mutually inverse permutations of Q^+, i.e.,

$$d_{l,\backslash *}(e_{l,*}(M)) = M = e_{l,*}(d_{l,\backslash *}(M))$$

for each leader $l \in Q$ and for every string $M \in Q^+$.

2.2 Quasigroups as vector valued Boolean functions **[1]**

Vector valued boolean functions (v.v.b.f.) is used to present finite quasigroups $(Q, *)$ of order 2^d in this paper**,** and we choose a bijection $\beta : Q \to \{0, 1, \ldots, 2^d - 1\}$ and represent $a \in Q$ by the d-bit representation $\beta(a)$. Hence, for each $a \in Q$ there are uniquely determined bits $x_1, x_2, \ldots, x_d \in \{0, 1\}$ (which depend on the choice of the bijection β) such that a is represented by the string $x_1x_2 \ldots x_d$. Then we identify a and its d-bit representation and write $a = x_1x_2 \ldots x_d$ or, sometimes, $a = (x_1, x_2, \ldots, x_d)$. Now, the binary operation * on Q can be seen as a vector valued operation $*_{vv} : \{0, 1\}^{2d} \to \{0, 1\}^d$ defined as:

$$a * b = c \Leftrightarrow *_{vv}(x_1, x_2, \ldots, x_d, y_1, y_2, \ldots, y_d) = (z_1, z_2, \ldots, z_d),$$

where $x_1 \ldots x_d$, $y_1 \ldots y_d$, $z_1 \ldots z_d$ are binary representations of a, b, c respectively.

Each z_i depends of the bits $x_1, x_2, \ldots, x_d, y_1, y_2, \ldots, y_d$ and is uniquely determined by them. So, each z_i can be seen as a 2d-ary Boolean function $z_i = f_i(x_1, x_2, \ldots, x_d, y_1, y_2, \ldots, y_d)$, where $f_i : \{0, 1\}^{2d} \to \{0, 1\}$ strictly depends on, and is uniquely determined by, *. Thus, we have the following:

Lemma 1. For every quasigroup $(Q, *)$ of order 2^d and for each bijection $Q \to \{0, 1 \ldots, 2^d - 1\}$ there are a uniquely determined v.v.b.f. $*_{vv}$ and d uniquely determined 2d-ary Boolean functions $f_1, f_2, \ldots, f_d$ such that for each $a, b, c \in Q$

$$a * b = c \Leftrightarrow *_{vv}(x_1, \ldots, x_d, y_1, \ldots, y_d) =$$
$$(f_1(x_1, \ldots, x_d, y_1, \ldots, y_d), \ldots, f_d(x_1, \ldots, x_d, y_1, \ldots, y_d)).$$

Each k-ary Boolean function $f(x_1, \ldots, x_k)$ can be represented in a unique way by its algebraic normal form (ANF), i.e., as a sum of products

$$ANF(f) = a_0 + \sum_{i=1}^{k} a_i x_i + \sum_{1 \le i < j \le k} a_{i,j} x_i x_j + \sum_{1 \le i < j < s \le k} a_{i,j,s} x_i x_j x_s + \ldots, \qquad \textbf{(3)}$$

where the coefficients a_0, a_i, $a_{i,j}$, . . . are in the set {0,1} and the addition and multiplication are in the field *GF(2).*

2.3 Multivariate Cubic Quasigroups

In this subsection a special class of quasigroups is presented, called multivariate cubic quasigroups (MCQs) that can be of different types.

Definition 3. A quasigroup (*Q*, *) of order 2^d is called Multivariate Cubic Quasigroup (MCQ) of type $Cub_cQuad_qLin_l$ if $c=d-(q+l)$ of the polynomials f_i are of degree 3 (i.e., are cubic) and *q* of them are of degree 2 (i.e., are quadratic) and *l* of them are of degree 1 (i.e., are linear), where $0 \le l \le q < c \le d$.

Definition 4. A quasigroup (*Q*, *) of order 2^d is called *Cubic_QG* if it is MCQ of type $Cub_dQuad_0Lin_0$.

Theorem 2. Let $A_1 = [f_{ij}]_{d\times d}$ and $A_2 = [g_{ij}]_{d\times d}$ be two $d \times d$ matrices of linear Boolean expressions, and let $b_1 = [u_i]_{d\times 1}$ and $b_2 = [v_i]_{d\times 1}$ be two $d \times 1$ vectors of linear or quadratic Boolean expressions. Let the functions f_{ij} and u_i depend only on variables $x_1, \ldots, x_d$, and let the functions g_{ij} and v_i depend only on variables $x_{d+1}, \ldots, x_{2d}$. If $Det(A_1) = Det(A_2) = 1$ in *GF(2)* and if $A_1\cdot(x_{d+1}, \ldots, x_{2d})^T + b_1 \equiv A_2\cdot(x_1, \ldots, x_d)^T + b_2$, then the vector valued operation $*_{vv}(x_1, \ldots, x_{2d}) = A_1 \cdot (x_{d+1}, \ldots, x_{2d})^T + b_1$ defines a quasigroup (Q, *) of order 2^d that is MQQ [1].

By using previous theorem it concluded that the formula $A \cdot (x_{d+1}, \ldots, x_{2d})^T + b$, where *A*, *b* depend only on variables $x_1, \ldots, x_d$ and $Det(A_1) = 1$ in *GF(2)*, may define a quasigroup; Replacing the two underlined phrases by *" quadratic Boolean expressions"* conclude that formula $A \cdot (x_{d+1}, \ldots, x_{2d})^T + b$ may define a quasigroup that is MCQ. In this manner, to keep the possibility of generating a valid quasigroup in a high range, the new quadratic Boolean expressions must behave like the linear ones, i.e. they cover the set $\{0\ldots 2^d\text{-}1\}$ when $(x_1, \ldots, x_d)$ moves over the same set; for $d = 3$, there are 28 quadratic Boolean expressions whose behave like the linear ones. Table (1) illustrates these expressions.

Table (1)*: Quadratic Boolean expressions of 3 variables behave like the linear ones*

$x_1+x_2x_3$	$x_2+x_3+x_1x_2$	$x_3+x_1x_2+x_2x_3$	$x_1+x_2+x_1x_2+x_1x_3$
$x_2+x_1x_3$	$x_2+x_3+x_1x_3$	$x_1+x_2+x_3+x_2x_3$	$x_1+x_2+x_1x_2+x_2x_3$
$x_3+x_1x_2$	$x_1+x_1x_2+x_2x_3$	$x_1+x_2+x_3+x_1x_3$	$x_1+x_3+x_1x_2+x_1x_3$
$x_1+x_2+x_1x_3$	$x_1+x_1x_3+x_2x_3$	$x_1+x_2+x_3+x_1x_2$	$x_1+x_3+x_1x_3+x_2x_3$
$x_1+x_2+x_2x_3$	$x_2+x_1x_2+x_1x_3$	$x_1x_2+x_1x_3+x_2x_3$	$x_1+x_2+x_1x_2+x_1x_3+x_2x_3$
$x_1+x_3+x_1x_2$	$x_2+x_1x_3+x_2x_3$	$x_2+x_3+x_1x_3+x_2x_3$	$x_1+x_3+x_1x_2+x_1x_3+x_2x_3$
$x_1+x_3+x_2x_3$	$x_3+x_1x_2+x_1x_3$	$x_2+x_3+x_1x_2+x_2x_3$	$x_2+x_3+x_1x_2+x_1x_3+x_2x_3$

For $d = 3$, a matrix A has a $Det(A) = 1$ in $GF(2)$ if it is of forms:

$$\begin{pmatrix} \alpha & \bar{\alpha} & \alpha \\ \bar{\alpha} & \alpha & \alpha \\ \beta & \bar{\beta} & \bar{\alpha} \end{pmatrix}, \begin{pmatrix} \alpha & \bar{\alpha} & \alpha \\ \bar{\alpha} & \beta & \alpha \\ \beta & \bar{\beta} & \bar{\alpha} \end{pmatrix}, \begin{pmatrix} \alpha & \bar{\alpha} & \alpha \\ \bar{\alpha} & \alpha & \bar{\beta} \\ \beta & \bar{\beta} & \bar{\alpha} \end{pmatrix}, \begin{pmatrix} \alpha & \bar{\alpha} & \alpha \\ \bar{\alpha} & \beta & \bar{\beta} \\ \beta & \bar{\beta} & \bar{\alpha} \end{pmatrix};$$ where $\bar{\alpha} = 1+\alpha$, and $\bar{\beta} = 1+\beta$;

In Table (2) a procedure MCQ(3) for generating a *Cubic_QGs* of order 2^3 is defined.

***Table (2)**: Algorithm for generating MCQs of order 2^3*

MCQ(3)
Input: All expressions in Table (1) **Output:** a set of *Cubic_QGs* of order $2^3 = 8$.
1. Choose randomly two different elements from Table (1) as α and β 2. Form a 3×3 matrix A such that ***Det(A) = 1*** in $GF(2)$ 3. Find all 3×1 vectors B_i of quadratic Boolean expressions of variables x_1, x_2, x_3 such that, for $X_2 = (x_4, x_5, x_6)^T$ the formula $A \cdot X_2 + B_i$ gives a valid quasigroup. 4. Set CB as the set of all B_i founded in step 3; ***If*** CB is an empty set ***then GoTo 1***; ***Else***, for each vector B_i in CB compute the vector $*i_{vv} = A \cdot X_2 + B_i$ ***if*** $*i_{vv}$ is a *Cubic_ QG* ***then*** **return** $*i_{vv}$; 5. ***if*** all vectors in CB did not return any *Cubic_ QG* ***then GoTo 1***;

2.4. Example:

This example executes the previous algorithm step by step;

1- Choose randomly two different elements from Table (1); Let $\alpha = x_1x_2+x_1x_3+x_2x_3$, and $\beta = x_2+x_3+x_1x_2+x_2x_3$.

2- Form a 3 × 3 matrix A, such that $Det(A) = 1$ in $GF(2)$, it is obtained

$$A= \begin{bmatrix} x_1x_2+x_1x_3+x_2x_3 & 1+x_1x_2+x_1x_3+x_2x_3 & x_1x_2+x_1x_3+x_2x_3 \\ 1+x_1x_2+x_1x_3+x_2x_3 & x_1x_2+x_1x_3+x_2x_3 & x_1x_2+x_1x_3+x_2x_3 \\ x_2+x_3+x_1x_2+x_2x_3 & 1+x_2+x_3+x_1x_2+x_2x_3 & 1+x_1x_2+x_1x_3+x_2x_3 \end{bmatrix}$$

3- Find all 3×1 vectors B_i of quadratic Boolean expressions of variables x_1, x_2, x_3 such that, for $X_2 = (x_4, x_5, x_6)^T$ the formula $A \cdot X_2 + B_i$ gives a valid quasigroup, for this case, there is a 256 vectors.

4- Set CB as the set of all B_i founded in previous step;

5- Take a vector B from CB $\begin{bmatrix} 1+x_1+x_2+x_1x_3 \\ 1+x_1+x_3 \\ 1+x_1+x_3+x_1x_2+x_1x_3+x_2x_3 \end{bmatrix}$

Now compute $*i_{vv} = A \cdot X_2 + B = (f_1, f_2, f_3)^T$

$f_1 = 1+x_1+x_2+x_1x_3+x_1x_2x_4+x_1x_3x_4+x_2x_3x_4+x_5+x_1x_2x_5+x_1x_3x_5+x_2x_3x_5+x_1x_2x_6+x_1x_3x_6+x_2x_3x_6,$
$f_2 = 1+x_1+x_3+x_4+x_1x_2x_4+x_1x_3x_4+x_2x_3x_4+x_1x_2x_5+x_1x_3x_5+x_2x_3x_5+x_1x_2x_6+x_1x_3x_6+x_2x_3x_6,$
$f_3 = 1+x_1+x_3+x_1x_2+x_1x_3+x_2x_3+x_2x_4+x_3x_4+x_1x_2x_4+x_2x_3x_4+x_5+x_2x_5+x_3x_5+x_1x_2x_5+x_2x_3x_5+x_6+x_1x_2x_6+ x_1x_3x_6+x_2x_3x_6;$

***Table(3)**: A quasigroup (Q,*) and its left parastrophe (Q,\) of order 2^3*

*	0	1	2	3	4	5	6	7
0	7	6	2	3	5	4	0	1
1	4	5	0	1	7	6	3	2
2	3	2	7	6	0	1	4	5
3	1	7	3	5	4	2	6	0
4	0	1	5	4	2	3	7	6
5	6	0	4	2	3	5	1	7
6	5	3	6	0	1	7	2	4
7	2	4	1	7	6	0	5	3

\	0	1	2	3	4	5	6	7
0	6	7	2	3	5	4	1	0
1	2	3	7	6	0	1	5	4
2	4	5	1	0	6	7	3	2
3	7	0	5	2	4	3	6	1
4	0	1	4	5	3	2	7	6
5	1	6	3	4	2	5	0	7
6	3	4	6	1	7	0	2	5
7	5	2	0	7	1	6	4	3

The result is a valid quasigroup, moreover it is a *Cubic_ QG*.
The corresponding left parastrophe $(Q, \backslash)$ of resulting quasigroup is also a *Cubic_ QG*: $\backslash_{vv}(x_1, \ldots, x_{2d}) = (g_1,g_2,g_3)^T$ where

$g_1= x_1x_2x_4+x_5+1+x_1+x_3+x_1x_3x_5+x_1x_3+x_2x_3x_5+x_1x_2x_6+x_1x_3x_6+x_2x_3x_6+x_2x_3,$
$g_2 = x_4+1+x_1+x_2+x_1x_2x_4+x_1x_3x_5+x_2x_3x_5+x_1x_2x_6+x_1x_3x_6+x_2x_3x_6+x_2x_3,$
$g_3 = x_4+x_1x_3x_4+x_2x_4+x_1x_2+x_3x_4+x_2x_3+x_1x_3x_5+x_2x_5+x_3x_5+x_6+x_1x_3;$

3. Description of the algorithm:

A generic description for MCQ scheme can be expressed as a typical multivariate cubic system: $T \circ P' \circ S: \{0, 1\}^n \rightarrow \{0, 1\}^n$ where T and S are two nonsingular linear transformations, and P′ is a bijective multivariate cubic mapping on $\{0, 1\}^n$.
First an algorithm will be presented in Table (4) to show how the mapping $P': \{0,1\}^n \rightarrow \{0,1\}^n$ is defined. Public and private keys generation process is described in Table (5). While Table (6) illustrates the signing process algorithm.

***Table (4):** Definition of the nonlinear mapping $P' : \{0, 1\}^n \rightarrow \{0, 1\}^n$*

P'(n)
Input: Integer n, where $n = d.k$: $k \geq 40$, $d=3$, a vector $x = (f_1, \ldots, f_n)$ of n linear Boolean functions of n variables, and a $d \times 1$ vector $L = (l_1, \ldots, l_d)$ of d variables.
Output: d quasigroups $*_1, \ldots, *_d$ and n multivariate cubic polynomials $P'_1 (x_1, \ldots, x_n, l_1, \ldots, l_d)$, $P'_i (x_1, \ldots, x_n)$, $i = 2, \ldots, n$.
1. Represent a vector $x = (f_1, \ldots, f_n)$ of n linear Boolean functions of n variables $x_1, \ldots, x_n$, as a string $x = X_1 \ldots X_k$ where X_i are vectors of dimension d;
2. By calling the procedure MCQ(d) generate large set of Cubic_QG of order 2^d; Then pick randomly d different Cubic_QGs $*_1, \ldots, *_d$;
3. Compute $y = Y_1 \ldots Y_k$ where: $Y_1 = L *_1 X_1$, $Y_j = X_{j-1} *_{[(j-1) \bmod d]+1} X_j$, for $j = 2, \ldots, k$;
4. Output: d Quasigroups $*_1, \ldots, *_d$ and y as n multivariate cubic polynomials $P'_1 (x_1, \ldots, x_n, l_1, \ldots, l_d)$, $P'_i (x_1, \ldots, x_n)$, $i = 2, \ldots, n$.

Table (5): *Algorithm for generating the public and private key for the MCQ scheme*

Input: Integer n, where $n = d.k$: $k \geq 40$, $d=3$.
Output:
Public key P: n multivariate cubic polynomials $P_i(x_1, \ldots, x_n, l_1, \ldots, l_d), i = 1, \ldots, n$;
Private key: Two nonsingular Boolean matrices T and S of order $n \times n$ and d Cubic_QGs $*_1, \ldots, *_d$ of order 2^d and one vector $L = (l_1, \ldots, l_d)$.
1. Generate two nonsingular $n \times n$ Boolean matrices T and S (uniformly at random).
2. Generate a $d \times 1$ vector $L = (l_1, \ldots, l_d)$ of d variables.
3. Call the procedure for definition of $P'(n) : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and from there also obtain the quasigroups $*_1, \ldots, *_d$.
4. Compute $y = T(P'(S(x)))$ where $x = (x_1, \ldots, x_n)$.
5. Output: The **public key is y** as n multivariate cubic polynomials $P_i(x_1, \ldots, x_n, l_1, \ldots, l_d), i = 1, \ldots, n,$ and the **private key** is the tuple $(T, S, *_1, \ldots, *_d, L)$.

Table (6): *Algorithm for decryption/signing with the private key* $(T, S, *_1, \ldots, *_d, L)$

Input: a vector $y = (y_1, \ldots, y_n)$.
Output: a vector $x = (x_1, \ldots, x_n)$ such that $\boldsymbol{P}(x) = y$.
1. Set $y' = T^{-1}.y$.
2. Represent y' as $y' = Y_1 \ldots Y_k$ where Y_i are vectors of dimension d.
3. By using the left parastrophes $\backslash_i$ of the quasigroups $*_1, \ldots, *_d$, obtain $x' = X_1 \ldots X_k$, such that: $X_1 = L \backslash_1 Y_1$, and $X_i = X_{i-1} \backslash_{[(i-1) \bmod d]+1} Y_i$, for $i = 2, \ldots, k$;
4. Compute $x = S^{-1}.x'$.

The algorithm for encryption with the public key is a straightforward application of the set of n multivariate polynomials $P=\{P_i(x_1, \ldots, x_n, l_1, \ldots, l_d), i = 1, \ldots, n\}$ over a vector $x = (x_1, \ldots, x_n)$, i.e. $y = P(x)$.

4. MCQ vs. MQQ (Operating characteristics):

Comparative study between MCQ and MQQ schemes is presented in this section.

4.1. Speed of keys generation comparison

The algorithm for generating *MCQs* is highly effective and straight procedure, especially if it is compared with the randomly algorithm which is proposed in [1] for generating *MQQs*. Figure (1) shows slightly that difference.

In [1] the bijection of Dobbertin [4] for m = 6 is used to eliminate the linear coordinates from MQQ scheme, namely in *P'(n)* procedure. This is adding an extra time T_{Dob} to the consumed time for public key generating. Then $T_{MCQ\_Pub} = T_{MQQ\_Pub} - T_{Dob}$.

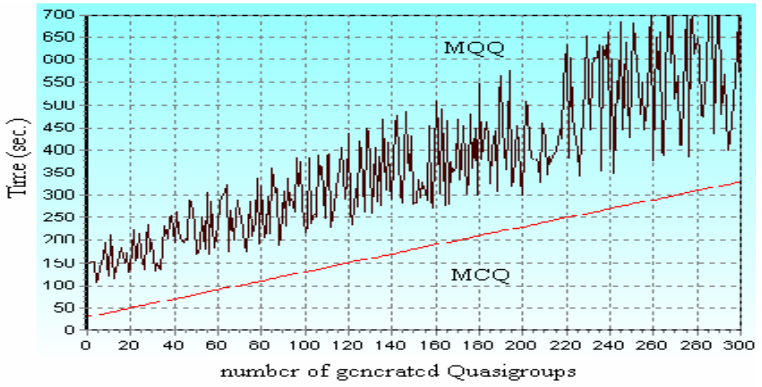

Figure (1): *Speed of keys generation comparison*

4.2. Speed of encryption and decryption comparison

MCQ scheme is time-consuming in encryption with the public key more than MQQ, because its public key consists of cubic Boolean expressions; while it is faster in decryption/signing in private key than MQQ, because MQQ uses the inverse bijection of Dobbertin, then $T_{MCQ\_Dec\_Pri} = T_{MQQ\_Dec\_Pri} - T_{Dob^{-1}}$; for this reason, MCQ is suitable for signing and especially for short signature.

4.3. Size of public and private keys

Public key consists of n multivariate polynomials $P_i(x_1, ..., x_n, l_1,..., l_d)$, $i = 1...n$. rename the variables $x_1, ..., x_n, l_1, ..., l_d$ by $t_1, ..., t_n, t_{n+1}, ..., t_{n+d}$ respectively, and set $m=n+d$; every P_i can be represented as

$$P_i = a_{i,0,0,0} + \sum_{j=1}^{m} a_{i,j,0,0} t_j + \sum_{j=1}^{m-1} \sum_{k=j+1}^{m} a_{i,j,k,0} t_{k-j} t_k + \sum_{j=1}^{m-2} \sum_{k=j+1}^{m-1} \sum_{h=k+1}^{m} a_{i,j,k,h} t_{k-j} t_k t_h$$

So, for every P_i there are:

- 1 term of degree 0,
- m terms of degree 1,
- $\frac{m(m-1)}{2}$ terms of degree 2,
- $\frac{m(m-1)(m-2)}{6}$ terms of degree 3;

Then it is needed $1+m+\frac{m(m-1)}{2}+\frac{m(m-1)(m-2)}{6}$ bits to store every P_i; so, the size of the public key is $\left[n(1+m+\frac{m(m-1)}{2}+\frac{m(m-1)(m-2)}{6})\right]$ bits.

The private key consists of:

- two $n \times n$ Boolean matrices T and S need $2n^2$ bits,
- one leader vector $(l_1, ..., l_d)^T$ needs d bits.
- d quasigroups, each of them consists of d multivariate polynomials $f_i(x_1, ..., x_{2d})$. Memory size needed for storage these quasigroups is $d^2(1+2d+\frac{2d(2d-1)}{2}+\frac{2d(2d-1)(2d-2)}{6})$ bits,

Then, the size of the private key is $\left[2n^2 + d^2(1+2d+d(2d-1)+\frac{2d(d-1)(2d-1)}{3})+d\right]$ bits.

Table (7) illustrates the size of public and private key of MCQ and MQQ for $n \geq 120$, where $d = 3$

***Table (7)** Memory size in KBytes for public and private key*

n	MCQ		MQQ[1].	
	Size of the Public key	**Size of the Private Key**	**Size of the Public key**	**Size of the Private Key**
120	4545	4	106	9
135	7220	5	151	10
150	10932	6	207	11
165	15920	7	276	12
180	22447	8	358	13
195	30800	9	455	14

4.4. Security analysis and security level comparison:

MQQ security analysis is presented in [1] with a conjectured security level $2^{(n/2)}$ when $n \geq 140$. MCQ scheme has the same strength as MQQ against the different types of attack; Moreover MCQ scheme has an important advantage against XL attack [3] and Grobner basis attacks, appears in that the number of variables is more than the equations number, where the equations number is *n*, and the variables number is $n + d$; This is rising the conjectured security level of MCQ to 2^{n+d} when $n \geq 120$.

4.5. Pool size of MCQs of order 2^3 and MQQs of order 2^5 comparison:

Size of the pool set for 8 MQQs of order 2^5 is at least of order 2^{149} [1]; For MCQ scheme with $d = 3$, there are more than 2^{11} choices for the matrix *A*, and for each of them there are more than 2^7 *B* vectors, then the size of the pool set for 3 MCQs of order 2^3 is at least of order 2^{57}.

5. The parallelizable nature of MCQ scheme:

MCQ scheme has a highly parallelizable nature; that is slightly appears in Figure (2) where 150-bit MCQ was implemented in C, using OpenMP 2.0 under Microsoft Visual Studio 2008. It is clear that increasing of the number of CPU cores can speed up MCQ algorithm almost linearly with the number of cores. Thus, MCQ is very suitable for signing and especially for short signatures.

6. Conclusions:

The results about our PKC can be briefly summarized as:

- it is a public key block cipher algorithm with no message expansion;
- it has one parameter $n \geq 120$ the bit length of the encrypted block;
- it is a deterministic one-to-one mapping;
- its encryption speed is comparable to the speed of MQQ scheme;

- its decryption/signature speed is more than MQQ speed, which has decryption/signature speed as a typical symmetric block cipher (i.e., in the range of 500–1000 times faster than the most popular public key schemes)[1];
- its performance can be increased many times if it is implemented in parallel hardware or software environment because his nature is highly parallelizable;

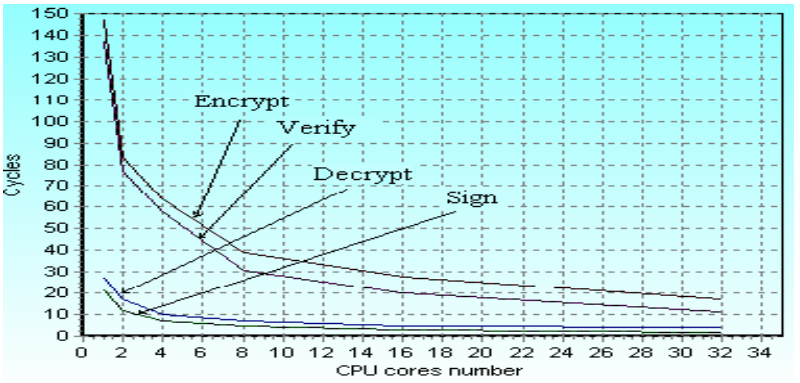

Figure (2): *Software speeds (in number of cycles)by using OpenMP*

- it is well suited and suitable for short signatures;
- its conjectured security level when $n = d.k : k \geq 40, d = 3$ is $2^{(n+d)}$.

References

[1] "Multivariate Quadratic Trapdoor Functions Based on Multivariate Quadratic Quasigroups", Proceedings of the AMERICAN CONFERENCE ON APPLIED MATHEMATICS (MATH '08), Cambridge, Massachusetts, USA, March 24-26, 2008.

[2] S. Markovski, "Quasigroup string processing and applications in cryptography", in Proc. 1-st Inter. Conf.Mathematics and Informatics for industry – MII, Thessaloniki 2003, pp. 278–290, 2003.

[3] M. Sugita, M. Kawazoe and H. Imai, "Relation between the XL Algorithm and Grobner Basis Algorithms", IEICE-Tran Fund Elec, Comm & Comp Sci Vol. E89-A, Number 1 pp. 11–18, 2006.

[4] H. Dobbertin, "One-to-one highly nonlinear power functions on GF(2^n)", Appl. Algebra Eng. Commun. Comput., Vol. 9(2), pp. 139-152, 1998.

APPENDIX

An example of private and public key generation with n = 9 bits and d = 3; Let x = (x_1, x_2, . . . , x_9) be a vector of 9 Boolean variables. The private and the public key is created by the following procedure:

1) Set

S =

1	0	0	1	0	1	1	1	1
1	0	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1
0	1	1	1	0	1	1	1	0
0	1	1	0	1	0	0	0	1
1	1	0	0	1	0	1	1	1
1	1	0	1	0	0	1	1	1
1	0	0	1	0	0	1	1	1
1	1	0	1	1	1	1	0	0

T =

1	1	1	0	1	1	0	0	1
1	0	0	1	0	1	1	1	0
1	1	1	0	1	1	1	1	0
1	0	1	0	1	1	1	0	1
0	0	1	0	0	1	1	0	0
0	0	1	1	0	1	1	1	1
0	0	0	1	1	0	0	0	1
1	1	1	0	0	1	1	0	0
1	1	0	0	0	1	1	1	1

where T and S are a nonsingular 9×9 Boolean matrices generated uniformly at random;

2) Set

$*_1 (x_1, \ldots, x_6) =$

$(1+x_1+x_3+x_1x_2+x_1x_4+x_2x_4+x_1x_2x_4+x_1x_3x_4+x_2x_3x_4+x_5+x_1x_5+x_2x_5+x_1x_2x_5+x_1x_3x_5+x_2x_3x_5+x_1x_6+x_2x_6+x_1x_2x_6+x_1x_3x_6+x_2x_3x_6,$
$x_2+x_3+x_1x_3+x_2x_3+x_4+x_1x_4+x_2x_4+x_1x_2x_4+x_1x_3x_4+x_2x_3x_4+x_1x_5+x_2x_5+x_1x_2x_5+x_1x_3x_5+x_2x_3x_5+x_1x_6+x_2x_6+x_1x_2x_6+x_1x_3x_6+x_2x_3x_6,$
$x_3+x_1x_2+x_2x_3+x_1x_4+x_3x_4+x_1x_2x_4+x_5+x_1x_5+x_3x_5+x_1x_2x_5+x_6+x_1x_6+x_2x_6+x_1x_2x_6+x_1x_3x_6+x_2x_3x_6);$

$*_2 (x_1, \ldots, x_6) =$

$(1+x_1+x_2+x_1x_3+x_1x_2x_4+x_1x_3x_4+x_2x_3x_4+x_5+x_1x_2x_5+x_1x_3x_5+x_2x_3x_5+x_1x_2x_6+x_1x_3x_6+x_2x_3x_6,$
$1+x_1+x_3+x_4+x_1x_2x_4+x_1x_3x_4+x_2x_3x_4+x_1x_2x_5+x_1x_3x_5+x_2x_3x_5+x_1x_2x_6+x_1x_3x_6+x_2x_3x_6,$
$1+x_1+x_3+x_1x_2+x_1x_3+x_2x_3+x_2x_4+x_3x_4+x_1x_2x_4+x_2x_3x_4+x_5+x_2x_5+x_3x_5+x_1x_2x_5+x_2x_3x_5+x_6+x_1x_2x_6+x_1x_3x_6+x_2x_3x_6);$

$*_3(x_1, \ldots, x_6) =$

$(1+x_1+x_2+x_1x_2+x_1x_3+x_1x_4+x_1x_2x_4+x_2x_3x_4+x_5+x_1x_5+x_1x_2x_5+x_2x_3x_5+x_1x_6+x_1x_2x_6+x_2x_3x_6,$
$x_2+x_1x_3+x_2x_3+x_4+x_1x_4+x_1x_2x_4+x_2x_3x_4+x_1x_5+x_1x_2x_5+x_2x_3x_5+x_1x_6+x_1x_2x_6+x_2x_3x_6,$
$1+x_1+x_3+x_2x_3+x_2x_4+x_1x_3x_4+x_2x_3x_4+x_5+x_2x_5+x_1x_3x_5+x_2x_3x_5+x_6+x_1x_6+x_1x_2x_6+x_2x_3x_6);$

3) Set a Leader L=(e_1,e_2,e_3); we replace the notation L=(l_1, l_2,l_3) by (e_1, e_2, e_3) to make it easy to distinguish between number 1 and letter *l*.

4) The tuple (S, T, $*_1$, $*_2$, $*_3$, L) is the private key;

5) Set $x' = S \,.\, x^T =$

$$\begin{pmatrix} x_1+x_4+x_6+x_7+x_8+x_9 \\ x_1+x_3+x_4+x_5+x_6+x_7+x_8 \\ x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9 \\ x_2+x_3+x_4+x_6+x_7+x_8 \\ x_2+x_3+x_5+x_9 \\ x_1+x_2+x_5+x_7+x_8+x_9 \\ x_1+x_2+x_4+x_7+x_8+x_9 \\ x_1+x_4+x_7+x_8+x_9 \\ x_1+x_2+x_4+x_5+x_6+x_7 \end{pmatrix}$$

6) Represent the vector x' by chunks of 3 bits, i.e. $x' = X_1X_2X_3$;

7) Compute $y'=Y_1Y_2Y_3$ such that $Y_1 = L *_1 X_1$, $Y_2 = X_1 *_2 X_2$, $Y_3 = X_2 *_3 X_3$;

8) Compute $y = T \,.\, y'^T$; The following relations will be obtained:

$y_1 = e_1+e_2+e_1e_3+e_3+e_1x_1+e_1x_4+e_1x_6+e_1x_7+e_1x_8+e_3x_9+e_1e_2x_1+e_1e_2x_4+e_1e_2x_6+e_1e_2x_7+e_1e_2x_8+e_3x_3+e_3x_5+e_1x_2+e_2x_1+e_2x_2+e_2x_3+e_2x_4+e_2x_5+$
$e_2x_6+e_2x_7+e_2x_8+e_2x_9+e_1e_2x_2+e_1e_3x_1+e_1e_3x_2+e_1e_3x_3+e_1e_3x_4+e_1e_3x_5+e_1e_3x_6+e_1e_3x_7+e_1e_3x_8+e_1e_3x_9+e_2e_3x_1+e_2e_3x_2+e_2e_3x_3+e_2e_3x_4+e_2e_3x_5+$
$e_2e_3x_6+e_2e_3x_7+e_2e_3x_8+e_2e_3x_9+x_1x_2x_5+x_2x_4x_5+x_4x_5+x_2x_5x_6+x_5x_6+x_1x_3x_9+x_4x_9+x_6x_9+x_4+x_5x_8+x_7x_9+x_1x_3x_4+x_1x_3x_6+x_1x_3x_8+x_1x_4x_5+$
$x_1x_5x_6+x_1x_5x_8+x_3x_5x_7+x_2x_5x_8+x_3x_5x_8+x_3x_5x_9+x_1x_2x_7+x_3+x_7+x_8+x_1+x_2x_7+x_3x_5+x_3x_9+x_2x_4+x_2x_5+x_2x_4x_9+x_2x_3x_5+x_3x_4x_7+x_4x_5x_9+x_4x_7x_9+$
$x_4x_8+x_2x_6+x_5x_6x_7+x_5x_6x_9+x_6x_8+x_2x_4x_8+x_4x_5x_7+x_2x_7x_9+x_2x_3x_7+x_2x_6x_8+x_2x_6x_9+x_7x_8+x_1x_7x_9+x_3x_6x_7+x_6x_7x_9,$

$y_2 = e_1+e_3+e_1e_2+e_1x_1+e_1x_4+e_1x_6+e_1x_7+e_1x_8+e_2x_1+e_2x_4+e_2x_6+e_2x_7+e_2x_8+e_1e_2x_1+e_1e_2x_4+e_1e_2x_6+e_1e_2x_7+e_1e_2x_8+e_1e_3x_1+e_1e_3x_4+e_1e_3x_6+$
$e_1e_3x_7+e_1e_3x_8+e_2e_3x_1+e_2e_3x_4+e_2e_3x_6+e_2e_3x_7+e_2e_3x_8+e_1x_2+e_2x_2+e_1e_2x_2+e_1e_3x_2+e_2e_3x_2+x_3x_9+x_1x_7+x_1x_8+x_1x_2x_5+x_2x_4x_5+x_2x_5x_6+x_1x_3x_9+$
$x_8x_9+x_2x_7+x_1x_4+x_1x_6+x_1x_3x_4+x_1x_3x_6+x_1x_3x_8+x_1x_4x_5+x_1x_5x_6+x_1x_5x_8+x_3x_5x_7+x_2x_5x_8+x_3x_5x_8+x_3x_5x_9+x_1x_2x_4+x_1x_2x_6+x_1x_2x_8+x_4+x_9+$
$x_2x_5+x_2x_9+x_4x_5+x_5x_6+x_7x_9+x_2x_8+x_3x_4+x_2x_5x_9+x_1x_2x_9+x_1x_3x_5+x_3x_4x_5+x_1x_2x_7+x_1x_3x_7+x_1x_5x_7+x_1x_5x_9+x_2x_5x_7+x_3x_6+x_6+x_3x_5x_6,$

$y_3 = e_1+e_2+e_1e_3+e_3+e_1x_1+e_1x_4+e_1x_6+e_1x_7+e_1x_8+e_3x_9+e_1e_2x_1+e_1e_2x_4+e_1e_2x_6+e_1e_2x_7+e_1e_2x_8+e_3x_3+e_3x_5+e_1x_2+e_2x_1+e_2x_2+e_2x_3+e_2x_4+e_2x_5+$
$e_2x_6+e_2x_7+e_2x_8+e_2x_9+e_1e_2x_2+e_1e_3x_1+e_1e_3x_2+e_1e_3x_3+e_1e_3x_4+e_1e_3x_5+e_1e_3x_6+e_1e_3x_7+e_1e_3x_8+e_1e_3x_9+e_2e_3x_1+e_2e_3x_2+e_2e_3x_3+e_2e_3x_4+e_2e_3x_5+$
$e_2e_3x_6+e_2e_3x_7+e_2e_3x_8+e_2e_3x_9+x_1x_7+x_1x_8+x_1x_2x_5+x_2x_4x_5+x_2x_5x_6+x_1x_3x_9+x_3x_7+x_5x_7+x_3x_8+x_8x_9+x_1x_4+x_1x_6+x_1x_3x_4+x_1x_3x_6+x_1x_3x_8+$
$x_1x_4x_5+x_1x_5x_6+x_1x_5x_8+x_3x_5x_7+x_2x_5x_8+x_3x_5x_8+x_3x_5x_9+x_1x_2x_4+x_1x_2x_6+x_1x_2x_8+x_8+x_1x_2+x_1x_5+x_5+x_2x_5+x_1x_3+x_2x_4+x_2x_6+x_7+x_7x_9+x_5x_8+$
$x_2x_5x_9+x_1x_2x_9+x_1x_3x_5+x_3x_4x_5+x_1x_2x_7+x_1x_3x_7+x_1x_5x_7+x_1x_5x_9+x_5x_9+x_2x_5x_7+x_3x_5x_6,$

$y_4 = e_1+e_2e_3+e_3x_9+e_3x_3+e_3x_5+e_2x_3+e_2x_5+e_2x_9+e_1e_3x_3+e_1e_3x_5+e_1e_3x_9+e_2e_3x_3+e_2e_3x_5+e_2e_3x_9+x_1x_2x_5+x_2x_4x_5+x_4x_5+x_2x_5x_6+x_5x_6+x_1x_3x_9+$
$x_4x_9+x_6x_9+x_1x_3x_4+x_1x_3x_6+x_1x_3x_8+x_1x_4x_5+x_1x_5x_6+x_1x_5x_8+x_3x_5x_7+x_2x_5x_8+x_3x_5x_8+x_3x_5x_9+x_4+x_2x_8+x_1x_2+x_4x_7+x_6x_7+x_2x_3x_4+x_2x_6x_7+$
$x_2x_3x_6+x_2x_4x_7+x_1x_2x_3+x_1x_5+1+x_6+x_8+x_1+x_9+x_2x_7+x_2x_3+x_3x_5+x_3x_9+x_2x_4+x_2x_5+x_5x_9+x_2x_5x_9+x_1x_3x_5+x_3x_4x_5+x_4x_8+x_2x_6+x_6x_8+x_1x_5x_7+$
$x_2x_4x_8+x_1x_5x_9+x_2x_5x_7+x_2x_6x_8+x_1x_2x_9+x_1x_3x_7+x_3x_5x_6,$

$y_5 = e_3+e_1e_2+e_2e_3+e_1x_1+e_1x_4+e_1x_6+e_1x_7+e_1x_8+e_3x_9+e_1e_2x_1+e_1e_2x_4+e_1e_2x_6+e_1e_2x_7+e_1e_2x_8+e_3x_3+e_3x_5+e_1x_2+e_2x_1+e_2x_2+e_2x_3+e_2x_4+e_2x_5+$
$e_2x_6+e_2x_7+e_2x_8+e_2x_9+e_1e_2x_2+e_1e_3x_1+e_1e_3x_2+e_1e_3x_3+e_1e_3x_4+e_1e_3x_5+e_1e_3x_6+e_1e_3x_7+e_1e_3x_8+e_1e_3x_9+e_2e_3x_1+e_2e_3x_2+e_2e_3x_3+e_2e_3x_4+e_2e_3x_5+$
$e_2e_3x_6+e_2e_3x_7+e_2e_3x_8+e_2e_3x_9+x_6+x_7+x_1x_3+x_3x_7+x_5x_7+x_3x_8+x_8x_9+x_2x_4+x_2x_6+x_2x_7+x_2x_9+x_1x_4+x_1x_6+x_1x_3x_4+x_1x_3x_6+x_1x_3x_8+x_1x_4x_5+$
$x_1x_5x_6+x_1x_5x_8+x_3x_5x_7+x_2x_5x_8+x_3x_5x_8+x_2x_3x_9+x_3x_5x_9+x_1x_2x_4+x_1x_2x_6+x_1x_2x_8+x_3x_4x_9+x_3x_6x_9+x_4x_8+x_6x_8+x_2x_5x_9+x_1x_2x_9+x_2x_7x_9+x_2x_3x_5+$
$x_2x_6x_7+x_3x_6x_7+x_5x_6x_7+x_6x_7x_9+x_2x_3x_7+x_1x_7x_9+x_2x_4x_7+x_3x_4x_7+x_1x_5x_9+x_4x_5x_7+x_4x_7x_9+x_7x_8,$

$y_6 = e_3+e_1e_2+e_2e_3+e_1x_1+e_1x_4+e_1x_6+e_1x_7+e_1x_8+e_3x_9+e_1e_2x_1+e_1e_2x_4+e_1e_2x_6+e_1e_2x_7+e_1e_2x_8+e_3x_3+e_3x_5+e_1x_2+e_2x_1+e_2x_2+e_2x_3+e_2x_4+e_2x_5+$
$e_2x_6+e_2x_7+e_2x_8+e_2x_9+e_1e_2x_2+e_1e_3x_1+e_1e_3x_2+e_1e_3x_3+e_1e_3x_4+e_1e_3x_5+e_1e_3x_6+e_1e_3x_7+e_1e_3x_8+e_1e_3x_9+e_2e_3x_1+e_2e_3x_2+e_2e_3x_3+e_2e_3x_4+e_2e_3x_5+$
$e_2e_3x_6+e_2e_3x_7+e_2e_3x_8+e_2e_3x_9+x_1x_2x_5+x_2x_4x_5+x_2x_5x_6+x_1x_3x_9+x_2+x_1x_3x_4+x_1x_3x_6+x_1x_3x_8+x_1x_4x_5+x_1x_5x_6+x_1x_5x_8+x_3x_5x_7+x_2x_5x_8+x_3x_5x_8+$
$x_3x_5x_9+x_5+x_4x_5+x_5x_6+x_7x_9+x_2x_8+x_1x_2x_7+x_6+x_9+x_2x_3+x_3x_5+x_3x_7+x_3x_8+x_5x_7+x_1x_9+x_2x_4+x_5x_9+x_2x_4x_9+x_2x_3x_5+x_3x_4x_7+x_4x_5x_9+x_4x_7x_9+$
$x_4x_8+x_2x_6+x_5x_6x_7+x_5x_6x_9+x_6x_8+x_2x_4x_8+x_4x_5x_7+x_2x_7x_9+x_2x_3x_7+x_2x_6x_8+x_2x_6x_9+x_7x_8+x_1x_7x_9+x_3x_6x_7+x_6x_7x_9,$

$y_7 = x_5x_8+x_2+x_5+x_1x_2+x_4x_5+x_5x_6+x_2x_8+x_1x_3+x_1x_5+1+x_3+x_6+x_7+x_8+x_2x_3+x_3x_5+x_1x_9+x_8x_9+x_2x_5+x_1x_2x_4+x_2x_5x_9+x_2x_4x_9+x_1x_3x_5+$
$x_2x_3x_5+x_3x_4x_5+x_3x_4x_7+x_4x_5x_9+x_4x_7x_9+x_4x_8+x_1x_6+x_5x_6x_7+x_5x_6x_9+x_6x_8+x_1x_7+x_1x_5x_7+x_1x_8+x_2x_4x_8+x_1x_5x_9+x_2x_5x_7+x_4x_5x_7+x_1x_2x_8+$
$x_2x_7x_9+x_2x_3x_7+x_1x_2x_6+x_2x_6x_8+x_2x_6x_9+x_1x_2x_9+x_1x_4+x_7x_8+x_1x_3x_7+x_1x_7x_9+x_3x_5x_6+x_3x_6x_7+x_6x_7x_9,$

$y_8 = e_1+x_5+e_2+e_1e_3+e_3+e_1x_1+e_1x_4+e_1x_6+e_1x_7+e_1x_8+e_3x_9+e_1e_2x_1+e_1e_2x_4+e_1e_2x_6+e_1e_2x_7+e_1e_2x_8+e_3x_3+e_3x_5+e_1x_2+e_2x_1+e_2x_2+e_2x_3+e_2x_4+$
$e_2x_5+e_2x_6+e_2x_7+e_2x_8+e_2x_9+e_1e_2x_2+e_1e_3x_1+e_1e_3x_2+e_1e_3x_3+e_1e_3x_4+e_1e_3x_5+e_1e_3x_6+e_1e_3x_7+e_1e_3x_8+e_1e_3x_9+e_2e_3x_1+e_2e_3x_2+e_2e_3x_3+e_2e_3x_4+$
$e_2e_3x_5+e_2e_3x_6+e_2e_3x_7+e_2e_3x_8+e_2e_3x_9+x_6+x_7+x_9+x_1x_3+x_3x_7+x_5x_7+x_3x_8+x_8x_9+x_2x_4+x_2x_6+x_2x_7+x_2x_9+x_1x_4+x_1x_6+x_1x_3x_4+x_1x_3x_6+x_1x_3x_8+$
$x_1x_4x_5+x_1x_5x_6+x_1x_5x_8+x_3x_5x_7+x_2x_5x_8+x_3x_5x_8+x_2x_3x_9+x_3x_5x_9+x_1x_2x_4+x_1x_2x_6+x_1x_2x_8+x_3x_4x_9+x_3x_6x_9+x_4x_8+x_6x_8+1+x_3+x_2x_5x_9+x_1x_2x_9+$
$x_2x_7x_9+x_2x_3x_5+x_2x_6x_7+x_3x_6x_7+x_5x_6x_7+x_6x_7x_9+x_2x_3x_7+x_1x_7x_9+x_2x_4x_7+x_3x_4x_7+x_1x_5x_9+x_4x_5x_7+x_4x_7x_9+x_7x_8,$

$y_9 = e_1+e_1e_2+e_2+e_1e_3+e_2e_3+x_2+x_1x_3x_4+x_1x_3x_6+x_1x_3x_8+x_1x_4x_5+x_1x_5x_6+x_1x_5x_8+x_3x_5x_7+x_2x_5x_8+x_3x_5x_8+x_2x_3x_9+x_3x_5x_9+x_3x_4x_9+x_3x_6x_9+$
$x_4x_5+x_4x_7+x_4x_9+x_5x_6+x_6x_7+x_6x_9+x_7x_9+x_5x_8+x_2x_8+x_2x_3x_4+x_2x_3x_6+x_1x_2x_7+x_1x_2x_3+x_3+x_6+x_7+x_8+x_2x_3+x_3x_5+x_3x_9+x_2x_4+x_2x_9+x_2x_5x_9+$
$x_2x_3x_5+x_3x_4x_7+x_4x_7x_9+x_2x_6+x_5x_6x_7+x_1x_7+x_1x_8+x_2x_4x_8+x_1x_5x_9+x_4x_5x_7+x_2x_7x_9+x_2x_3x_7+x_2x_6x_8+x_1x_2x_9+x_7x_8+x_1x_7x_9+x_3x_6x_7+x_6x_7x_9;$

9) The public key is y, where

$y_i = P_i(x_1, x_2, \ldots, x_9, e_1, e_2, e_3)$, $i = 1, ..., 9$; P_i are multivariate cubic polynomials of 12 Boolean variables.