Key Encapsulation Mechanism From Modular Multivariate Linear Equations

Muhammad Rezal Kamel Ariffin[1,*], Abderrahmane Nitaj[2], Yanbin Pan[3], Nur Azman Abu[4]

[1] Institute for Mathematical Research, Universiti Putra Malaysia, Selangor, Malaysia
[2] Université de Caen, Normandie, France
[3] Chinese Academy of Science, Beijing, China
[4] Universiti Teknikal Malaysia Melaka, Melaka, Malaysia

[1,*]rezal@upm.edu.my,[2]abderrahmane.nitaj@unicaen.fr,[3]panyanbin@amss.ac.cn, [4]nura@utem.edu.my

*Corresponding Author

Abstract. In this article we discuss the modular pentavariate and hexavariate linear equations and its usefulness for asymmetric cryptography. Construction of our key encapsulation mechanism dwells on such modular linear equations whose unknown roots can be interpreted as long vectors within a lattice which surpasses the Gaussian heuristic; hence unable to be identified by the LLL lattice reduction algorithm. By utilizing our specially constructed public key when computing the modular hexavariate linear ciphertext equation, the decapsulation mechanism can correctly output the shared secret parameter. The scheme has short key length, no decapsulation failure issues, plaintext-to-ciphertext expansion of one-to-one as well as uses "simple" mathematics in order to achieve maximum simplicity in design, such that even practitioners with limited mathematical background will be able to understand the arithmetic. Due to inexistence of efficient algorithms running upon a quantum computer to obtain the roots of our modular pentavariate and hexavariate linear equation and also to retrieve the private key from the public key, our key encapsulation mechanism can be a probable candidate for seamless post quantum drop-in replacement for current traditional asymmetric schemes.

Keywords: Post quantum cryptosystem, LLL algorithm, modular pentavariate linear equation root problem, modular hexavariate linear equation root problem

1 Introduction

Upon the discovery of Shor's algorithm in 1994 ([16]) which could solve the integer factorization problem as well as discrete logarithm based problems upon a quantum computer in polynomial time, cryptographers scrambled to find new hard mathematical problems which could resist Shor's algorithm and at the same time is able to provide asymmetric security (i.e. to be able to be used to design asymmetric cryptosystems that are quantum resistant). A compendium of potential hard problems was developed. Pioneering work can be traced to the code based cryptosystem by McEliece in 1978 ([13]). Lattice based cryptosystems which employs either the short vector problem or the closest vector problem were also popular to be utilized. Among them the NTRU cryptosystem in 1995 ([9]) and LWE cryptosystem in 2005 ([15]). Since then we have had (not limited to) schemes based on multivariate quadratic equations such as the Rainbow cryptosystem in 2005 ([5]) and the UOV cryptosystem in 2010 ([4]).

The ciphertext equation in this work which is based on the modular hexavariate linear equation, is motivated by Herrmann and May's results in 2008 ([8]) on the modular multivariate linear equation that states when the product of the upper bounds of the unknown roots of the equation is approximately equivalent (in bit size) or larger than the modulus, one cannot reduce the exponential time strategy to obtain the unknown roots. That is, under this scenario, to obtain the unknown roots one is only left with exponential running time strategies. The design of the ciphertext equation was also motivated by the fact that the LLL algorithm is unable to retrieve the vector $\mathbf{V}_0$ within a lattice when $||\mathbf{V}_0||$ is much larger than the Gaussian heuristic and the upper bound of vectors able to be output by the LLL algorithm. As for the key equation, we are motivated by two strategies. Firstly, we are motivated by brute force complexity to derive the private key from the public key by way of ensuring the root of our univariate, monic polynomial of degree 1 is larger than the required bound for Coppersmith's method to obtain it. Secondly, we are motivated by the modular pentavariate linear equation when constructing the set of five public key parameters needed for the scheme. The modular pentavariate linear equation that we utilize also has the same characteristics as described above for the hexavariate case.

1.1 Organisation of the Paper

The remainder of this paper is organized as follows. In Section 2, we discuss preliminary content surrounding the Minkowski theorem. In Section 3, we discuss the modular pentavariate linear equation root problem (MPLERP) and the modular hexavariate linear equation root problem (MHLERP). Then in Section 4, we put forward the KAZ key encapsulation mechanism. In Section 5, we observe the KAZ key problem through Coppersmith, Blackburn and lattice methodologies. One-wayness of KAZ (i.e. KAZ problem) is presented in Section 6. We conclude in Section 7.

2 Preliminary

Throughout this article, an n-bit integer a will be denoted as $a \approx 2^n$ unless mentioned otherwise. We also denote when two integers a and b are of the same bit length as $a \approx b$ unless mentioned otherwise. We also utilize the notation $[x]$ for $x \in \mathbb{R}$ as the nearest integer to x.

2.1 Minkowski's Theorem

The Minkowski Theorem which relates the length of the shortest vector in a lattice to the determinant (see [10]) provides initial information to formulate our scheme. It is as follows.

Theorem 1. *In an ω-dimensional lattice $\mathcal{L}$, there exists a non-zero vector $\boldsymbol{V}$ with*

$$\| \boldsymbol{V} \| \leq \sqrt{\omega}\ det(\mathcal{L})^{\frac{1}{\omega}}.$$

We note here that in lattices with fixed small dimension we can efficiently find the shortest vector, but for arbitrary dimensions, the problem of computing the shortest vector is known to be **NP**-hard under randomized reductions (see [1]). In order to find an approximation of the shortest vector, the LLL algorithm is able to compute in polynomial time such approximations up to a multiplicative factor of 2^{ω}, and this is sufficient for many applications. We use information from Theorem 1, to ensure that our vector $\mathbf{V}$ cannot be found by the LLL algorithm.

We will now observe the following remark.

Remark 1. The Gaussian heuristic says that a non-zero vector $\mathbf{V}_{short}$ will satisfy $||\mathbf{V}_{short}|| \approx \sigma(\mathcal{L})$ where $\sigma(\mathcal{L}) = \sqrt{\frac{\omega}{2\pi \mathrm{e}}}\det(M_{\mathcal{L}})^{\frac{1}{\omega}}$ and $M_{\mathcal{L}}$ is the corresponding matrix of the lattice $\mathcal{L}$. This is preeminently if $||\mathbf{V}_{short}|| < \sigma(\mathcal{L})$ of a particular lattice $\mathcal{L}$, then the lattice reduction algorithm LLL is likely easy to find the shortest vector when the dimension of the lattice is small.

3 Multivariate Linear Equation Root Problems

3.1 Uniqueness of Modular Multivariate Linear Equation Solutions

The following two remarks are related to the work of Herrmann and May on the modular linear equation $f(x_1, x_2, ..., x_k) = \sum_{i=1}^{k} a_i x_i \equiv 0 \pmod N$ (see [8]).

Remark 2. Let $f(x_1, x_2, ..., x_k) = a_1x_1 + a_2x_2 + ... + a_kx_k$ be a multivariate linear polynomial. One can hope to solve the modular linear equation $f(x_1, x_2, ..., x_k) \equiv 0 \pmod N$, that is to be able to find the set of solutions $(y_1, y_2, ..., y_k) \in \mathbb{Z}_N^k$, when the product of the unknowns are smaller than the modulus. More precisely, let X_i be upper bounds such that $|y_i| \leq X_i$ for $i = 1, ..., k$. Then one can roughly expect a unique solution whenever the condition $\prod_i X_i \leq N$ holds (see [8]). It is common knowledge that under the same condition $\prod_i X_i \leq N$ the unique solution $(y_1, y_2, ..., y_k)$ can heuristically be recovered by computing the shortest vector in an k-dimensional lattice by the LLL algorithm. In fact, this approach lies at the heart of many cryptanalytic results (see [3],[6] and [14]).

Remark 3. If in turn we have $\prod_i X_i \geq N^{1+\epsilon}$ then the modular linear equation given by $f(x_1, x_2, ..., x_k) = \sum_{i=1}^{k} a_i x_i \equiv 0 \pmod{N}$ usually has N^ϵ many solutions, which is exponential in the bit-size of N. As a result, there is no hope to find efficient algorithms that in general improve on this bound, since one is not able to output all roots in polynomial time.

3.2 Modular Pentavariate Linear Equation Root Problem

We now proceed to define the modular pentavariate linear equation root problem (MPLERP). Let n be an integer where we agree that 2^n is exponentially large. Let a_1, a_2, a_3, a_4 and a_5 be integers of n bit-size. Let c be an integer. Suppose we have the equation $c = a_1 v + a_2 w + a_3 x + a_4 y + a_5 z$. The MPLERP is when given $c' = a_1 v_0 + a_2 w_0 + a_3 x_0 + a_4 y_0 + a_5 z_0$ where v_0, w_0, x_0, y_0 and $z_0 \approx 2^n$, one has to identify the private parameters $(v_0, w_0, x_0, y_0, z_0)$ when $(a_1, a_2, a_3, a_4, a_5, c')$ is given. We note that, the equation can be viewed as $a_1 v_0 + a_2 w_0 + a_3 x_0 + a_4 y_0 + a_5 z_0 \equiv 0 \pmod{c'}$. Since $c \approx 2^{2n}$, we have $v_0, w_0, x_0, y_0, z_0 \approx c^{0.5}$.

The MPLERP Assumption *The advantage of any probabilistic polynomial time adversary running in time poly(n) in attempting to solve MPLERP is as stated in Remark 3.*

This MPLERP Assumption is a direct inference from Remark 3, since $v_0 w_0 x_0 y_0 z_0 \approx 2^{5n} > c' \approx 2^{2n}$.

3.3 Herrmann and May Remarks and MPLERP

Let the bounds be $v_0 < V \approx 2^n$, $w_0 < W \approx 2^n$, $x_0 < X \approx 2^n$, $y_0 < Y \approx 2^n$ and $z_0 < Z \approx 2^n$. For the MPLERP, we can see that $VWXYZ \approx 2^{5n}$. Since $c' \approx 2^{2n}$, Remark 3 can be observed within MPLERP via the equation $a_1 v_0 + a_2 w_0 + a_3 x_0 + a_4 y_0 + a_5 z_0 \equiv 0 \pmod{c'}$.

3.4 Lattice based analysis upon MPLERP

To further analyse the intractability of MPLERP, the conventional way to solve multivariate equations is to employ lattices as well as the LLL algorithm. We will focus on the equation

$$a_1 v + a_2 w + a_3 x + a_4 y + a_5 z \equiv 0 \pmod{c} \tag{1}$$

Consider the lattice $\mathcal{L}1_K$ with the matrix

$$M_{\mathcal{L}1_K} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & a_1 K \\ 0 & 1 & 0 & 0 & 0 & a_2 K \\ 0 & 0 & 1 & 0 & 0 & a_3 K \\ 0 & 0 & 0 & 1 & 0 & a_4 K \\ 0 & 0 & 0 & 0 & 1 & a_5 K \\ 0 & 0 & 0 & 0 & 0 & -cK \end{bmatrix}$$

Case 1 **(**$K = 1$**)** Let $\mathbf{V}_0$ be a vector of $\mathcal{L}1_1$. Then there exists the 6-tuple $(u_1, u_2, u_3, u_4, u_5, u_6) \in \mathbb{Z}^6$ such that

$$\begin{aligned}\mathbf{V}_0 &= (u_1, u_2, u_3, u_4, u_5, u_6)M_{\mathcal{L}1_1} \\ &= (u_1, u_2, u_3, u_4, u_5, a_1u_1 + a_2u_2 + a_3u_3 + a_4u_4 + a_5u_5 - cu_6)\end{aligned}$$

is on the lattice $\mathcal{L}1_1$. More precisely the lattice contains the vector solution $\mathbf{V}_0 = (v_0, w_0, x_0, y_0, z_0, 0)$.

Observe

$$\begin{aligned}||\mathbf{V}_0|| &= \sqrt{(v_0)^2 + (w_0)^2 + (x_0)^2 + (y_0)^2 + (z_0)^2} \\ &\approx 2^n\end{aligned}$$

We know that the LLL algorithm outputs a reduced basis where the norm of the shortest vector is less than

$$2^{\frac{\omega-1}{4}}\det(\mathcal{L}1_1)^{\frac{1}{\omega}} = 2^{\frac{5}{4}}\det(\mathcal{L}1_1)^{\frac{1}{6}} = 2^{\frac{5}{4}}(c)^{\frac{1}{6}} \approx 2^{0.3n}$$

where $\omega = \dim(\mathcal{L}1_1) = 6$ (see [10]). Obviously $2^n > 2^{0.3n}$. Then referring to Remark 1, the use of the LLL algorithm is insignificant. LLL experiments on corresponding lattice of equation (2) outputs vectors $\mathbf{V}$ where $||\mathbf{V}|| \ll ||\mathbf{V}_0||$.

Case 2 **(**$K > 1$**)** Let $K > 1$ be an integer to be determined later. We have $2^{\frac{5}{4}}\det(\mathcal{L}1_K)^{\frac{1}{6}} = 2^{\frac{5}{4}}(cK)^{\frac{1}{6}}$. We set $||\mathbf{V}_0|| < 2^{\frac{5}{4}}(cK)^{\frac{1}{6}}$ and hope that LLL will output $\mathbf{V}_0$ as the shortest vector. This will give us $K \geq \frac{2^{4n}}{2^{7.5}}$. LLL experiments on corresponding lattice of equation (2) with such values of K outputs vectors $\mathbf{V}$ where $||\mathbf{V}|| \ll ||\mathbf{V}_0||$.

3.5 Modular Hexavariate Linear Equation Root Problem

We now proceed to define the Modular Hexavariate Linear Equation Root Problem (MHLERP). Let n be an integer where we agree that $2^{\lceil \frac{3n}{20} \rceil}$ is exponentially large. Let p, a_1, a_2, a_3, a_4 and a_5 be integers of n bit-size. Let c be an integer. Suppose we have the equation $c \equiv u + a_1v + a_2w + a_3x + a_4y + a_5z \pmod{p}$. The MHLERP is when given $c' \equiv u_0 + a_1v_0 + a_2w_0 + a_3x_0 + a_4y_0 + a_5z_0 \pmod{p}$ where $u_0 \approx p^{\frac{3}{20}}$ and $v_0, w_0, x_0, y_0, z_0 \approx p^{\frac{1}{5}}$, one has to identify the private parameters $(u_0, v_0, w_0, x_0, y_0, z_0)$ when $(a_1, a_2, a_3, a_4, a_5, p, c')$ is given.

The MHLERP Assumption *The advantage of any probabilistic polynomial time adversary running in time poly(n) in attempting to solve MHLERP is at least* $O(p^{-\frac{3}{20}}) = O(2^{-\frac{3n}{20}})$.

This is the complexity to obtain u_0. From $c' - u_0 \equiv a_1v_0 + a_2w_0 + a_3x_0 + a_4y_0 + a_5z_0 \pmod{p}$ and since $v_0w_0x_0y_0z_0 < p$, the LLL algorithm might output $(v_0, w_0, x_0, y_0, z_0)$ in polynomial time [11]. See Remark 2 and Section 3.6 for discussions.

3.6 Herrmann and May Remarks and MHLERP

Let the bounds be $u_0 < U \approx p^{\frac{3}{20}}$, $v_0 < V \approx p^{\frac{1}{5}}$, $w_0 < W \approx p^{\frac{1}{5}}$, $x_0 < X \approx p^{\frac{1}{5}}$, $y_0 < Y \approx p^{\frac{1}{5}}$ and $z_0 < Z \approx p^{\frac{1}{5}}$. For the MHLERP, we can see that $UVWXYZ \approx p^{1.15}$. Hence, Remark 3 can be observed within MHLERP. From the defined MHLERP assumption, upon obtaining u_0, finding small roots via LLL from $c' - u_0 \equiv a_1v_0 + a_2w_0 + a_3x_0 + a_4y_0 + a_5z_0 \pmod p$ is feasible since $VWXYZ < p$.

3.7 Lattice based analysis upon MHLERP

To further analyse the intractability of MHLERP, the conventional way to solve multivariate equations is to employ lattices as well as the LLL algorithm. We will focus on the equation

$$c - u - a_1v - a_2w - a_3x - a_4y - a_5z \equiv 0 \pmod p \tag{2}$$

Analysis - 1 Consider the lattice $\mathcal{L}1_K$ with the matrix

$$M_{\mathcal{L}1_K} = \begin{bmatrix} [2^{0.05n}] & 0\,0\,0\,0\,0 & 0 & -K \\ 0 & 1\,0\,0\,0\,0 & 0 & -a_1K \\ 0 & 0\,1\,0\,0\,0 & 0 & -a_2K \\ 0 & 0\,0\,1\,0\,0 & 0 & -a_3K \\ 0 & 0\,0\,0\,1\,0 & 0 & -a_4K \\ 0 & 0\,0\,0\,0\,1 & 0 & -a_5K \\ 0 & 0\,0\,0\,0\,0 & [2^{0.2n}] & cK \\ 0 & 0\,0\,0\,0\,0 & 0 & pK \end{bmatrix}$$

Case 1 $(K = 1)$ Let $\mathbf{V}_0$ be a vector of $\mathcal{L}1_1$. Then there exists the 8-tuple $(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8) \in \mathbb{Z}^8$ such that

$$\begin{aligned}\mathbf{V}_0 &= (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8)M_{\mathcal{L}1_1} \\ &= ([2^{0.05n}]u_1, u_2, u_3, u_4, u_5, u_6, [2^{0.2n}]u_7, U^*)\end{aligned}$$

where $U^* = -u_1 - a_1u_2 - a_2u_3 - a_3u_4 - a_4u_5 - a_5u_6 + cu_7 + pu_8$ is on the lattice $\mathcal{L}1_1$. More precisely the lattice contains the vector solution $\mathbf{V}_0 = ([2^{0.05n}]u_0, v_0, w_0, x_0, y_0, z_0, [2^{0.2n}], 0)$. Observe

$$\begin{aligned}||\mathbf{V}_0|| &= \sqrt{(2^{0.05n}u_0)^2 + (v_0)^2 + (w_0)^2 + (x_0)^2 + (y_0)^2 + (z_0)^2 + (2^{0.2n})^2} \\ &\approx p^{\frac{1}{5}}\end{aligned}$$

We know that the LLL algorithm outputs a reduced basis where the norm of the shortest vector is less than

$$2^{\frac{\omega-1}{4}}\det(\mathcal{L}1_1)^{\frac{1}{\omega}} = 2^{\frac{7}{4}}\det(\mathcal{L}1_1)^{\frac{1}{8}} = 2^{\frac{7}{4}}(2^{0.25n}p)^{\frac{1}{8}} \approx p^{0.125}$$

where $\omega = \dim(\mathcal{L}1_1) = 8$ (see [10]). Obviously $p^{0.2} > p^{0.125}$. Then referring to Remark 1, the use of the LLL algorithm is insignificant. The LLL algorithm experiments on corresponding lattice of equation (2) outputs vectors $\mathbf{V}$ where $||\mathbf{V}|| \ll ||\mathbf{V}_0||$.

Case 2 ($K > 1$) Let $K > 1$ be an integer to be determined later. We have $2^{\frac{7}{4}} \det(\mathcal{L}1_K)^{\frac{1}{8}} = 2^{\frac{7}{4}}(2^{0.25n}pK)^{\frac{1}{8}}$. We set $||\mathbf{V}_0|| < 2^{\frac{7}{4}}(2^{0.25n}pK)^{\frac{1}{8}}$ and hope that the LLL algorithm will output $\mathbf{V}_0$ as the shortest vector. This will give us $K \geq \frac{p^{0.6}2^{-0.25n}}{2^{14}}$. The LLL algorithm experiments on corresponding lattice of equation (2) with such values of K outputs vectors $\mathbf{V}$ where $||\mathbf{V}|| \ll ||\mathbf{V}_0||$. The following is an example.

Example for 3.7 : Analysis - 1 This is an illustration of executing the LLL algorithm upon MHLERP for the case of Analysis-1 where $K > 1$. Let $n = 32$. We will use the parameters:

1. $u_0 = 21$
2. $v_0 = 95$
3. $w_0 = 103$
4. $x_0 = 87$
5. $y_0 = 74$
6. $z_0 = 86$
7. $a_1 = 3193415427$
8. $a_2 = 2205633754$
9. $a_3 = 3080513063$
10. $a_4 = 2991853793$
11. $a_5 = 3225880586$
12. $p = 3239617301$
13. $c = 1535369407$
14. $K = p^{10}$

Consider the lattice $\mathcal{L}1_K$ with the matrix $M\mathcal{L}1_K$ as discussed in Analysis-1 with the above parameters. Executing the LLL algorithm upon $M\mathcal{L}1_K$ produces:

$$M\mathcal{L}1_{K,2} = \begin{bmatrix} 24 & -6 & 10 & -3 & -26 & 7 & 0 & 0 \\ 12 & 17 & 12 & -33 & 8 & -15 & 0 & 0 \\ 18 & -7 & -28 & 1 & 1 & -21 & 0 & 0 \\ -33 & 0 & 0 & 13 & -34 & -9 & 0 & 0 \\ 6 & 32 & -7 & 34 & -10 & -30 & 0 & 0 \\ 21 & 56 & -36 & -16 & -8 & 21 & 0 & 0 \\ 3 & 4 & -2 & -21 & 15 & -2 & 84 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & -\rho \end{bmatrix}$$

where $\rho = 127331863864861663664206078030934820791581314721177559249722 998678759388452427633569090104223001$. We have

$\frac{M\mathcal{L}1_{K,2}(j,1)}{[2^{0.05n}]}+a_1M\mathcal{L}1_{K,2}(j,2)+a_2M\mathcal{L}1_{K,2}(j,3)+a_3M\mathcal{L}1_{K,2}(j,4)+a_4M\mathcal{L}1_{K,2}(j,5)+$
$a_5M\mathcal{L}1_{K,2}(j,6) - c\frac{M\mathcal{L}1_{K,2}(j,7)}{[2^{0.2n}]} \equiv 0 \pmod p$

for $j = 1, 2, 3, 4, 5, 6, 7$. Let

1. $\mathbf{V}_0 = (u_0.v_0, w_0, x_0, y_0, z_0, 0)$
2. $\mathbf{V}_j = (\frac{M\mathcal{L}1_{K,2}(j,1)}{[2^{0.05n}]}, M\mathcal{L}1_{K,2}(j,2), M\mathcal{L}1_{K,2}(j,3), M\mathcal{L}1_{K,2}(j,4), M\mathcal{L}1_{K,2}(j,5),$
   $M\mathcal{L}1_{K,2}(j,6), \frac{M\mathcal{L}1_{K,2}(j,7)}{[2^{0.2n}]})$ for $j = 1, 2, 3, 4, 5, 6, 7$.

We can see that $||\mathbf{V}_0|| \approx 201$, $||\mathbf{V}_1|| \approx 31$, $||\mathbf{V}_2|| \approx 43$, $||\mathbf{V}_3|| \approx 36$,$||\mathbf{V}_4|| \approx 39$,$||\mathbf{V}_5|| \approx 57$,$||\mathbf{V}_6|| \approx 72$ and $||\mathbf{V}_6|| \approx 26$. That is, the LLL algorithm produces much shorter vectors than $\mathbf{V}_0$. Our experiments produce such observation.

Analysis - 2

We will interpret equation (2) as $c = u + a_1v + a_2w + a_3x + a_4y + a_5z + pt$ for some $t \in \mathbb{Z}$, which implies as $u + a_1v + a_2w + a_3x + a_4y + a_5z + pt \equiv 0 \pmod c$.

Then we can have the relations $u = cu_1 - a_1u_2 - a_2u_3 - a_3u_4 - a_4u_5 - a_5u_6 + pu_8$ for some integers $(u_1, u_2, u_3, u_4, u_5, u_6, u_8)$ and $u + a_1v + a_2w + a_3x + a_4y + a_5z - cu_1 \equiv 0 \pmod p$. Continuing, we have $cu_1 - a_1u_2 - a_2u_3 - a_3u_4 - a_4u_5 - a_5u_6 + pu_8 + a_1v + a_2w + a_3x + a_4y + a_5z - cu_1 \equiv 0 \pmod p$. That is, $a_1(v - u_2) + a_2(w - u_3) + a_3(x - u_4) + a_4(y - u_5) + a_5(z - u_6) \equiv 0 \pmod p$. If $\gcd(a_1, p) = 1$, we have the equation $v = u_2 - \eta_1(w - u_3) - \eta_2(x - u_4) - \eta_3(y - u_5) - \eta_4(z - u_6) + pu_7$ for some integer u_7 and $\eta_1 = a_2a_1^{-1} \pmod p$, $\eta_2 = a_3a_1^{-1} \pmod p$, $\eta_3 = a_4a_1^{-1} \pmod p$ and $\eta_4 = a_5a_1^{-1} \pmod p$. Now consider the lattice $\mathcal{L}2$ with the matrix $M_{\mathcal{L}2}$ given by:

$$M_{\mathcal{L}2} = \begin{bmatrix} 0 & c & p \\ 1 & -a_1 & 0 \\ -\eta_1 & a_2 & 0 \\ -\eta_2 & a_3 & 0 \\ -\eta_3 & a_4 & 0 \\ -\eta_4 & a_5 & 0 \\ p & 0 & 0 \\ 0 & p & 0 \end{bmatrix}$$

Observe

$$(u_1, u_2, w - u_3, x - u_4, y - u_5, z - u_6, u_7, u_8)M_{\mathcal{L}2} = (v, u + a_2w + a_3x + a_4y + a_5z, pu_1).$$

We need $u_1 = 1$ in order to get the relation $u + a_1v + a_2w + a_3x + a_4y + a_5z + pt = c$. The dimension of $M_{\mathcal{L}2}$ is 8 and rank is 3. Its determinant is given by the relation $\det(\mathcal{L}2) = \sqrt{\det(M_{\mathcal{L}2}^t \cdot M_{\mathcal{L}2})}$. Specifically,

$$\det(\mathcal{L}2) = \left(\det \begin{bmatrix} 1+\eta_1^2+\eta_2^2+\eta_3^2+\eta_4^2+p^2 & -a_1-\eta_1 a_2-\eta_2 a_3-\eta_3 a_4-\eta_4 a_5 & 0 \\ -a_1-\eta_1 a_2-\eta_2 a_3-\eta_3 a_4-\eta_4 a_5 & a_1^2+a_2^2+a_3^2+a_4^2+a_5^2+p^2+c^2 & pc \\ 0 & pc & p^2 \end{bmatrix} \right)^{\frac{1}{2}}$$

Let $\mathbf{A}= \det(\mathcal{L}2)$. We know that the LLL algorithm outputs a reduced basis where the norm of the shortest vector is less than $2^{\frac{\omega-1}{4}}\det(\mathcal{L}2)^{\frac{1}{\omega}} = 2^{\frac{7}{4}}\det(\mathcal{L}2)^{\frac{1}{8}} = 2^{\frac{7}{4}}\mathbf{A}^{\frac{1}{8}}$ where $\omega = \dim(\mathcal{L}2) = 8$. Let $\mathbf{V}_0 = (v_0, u_0 + a_2 w_0 + a_3 x_0 + a_4 y_0 + a_5 z_0, p)$. We now analyse whether $||\mathbf{V}_0|| < 2^{\frac{7}{4}}\mathbf{A}^{\frac{1}{8}}$. Observe $2^{\frac{7}{4}}\mathbf{A}^{\frac{1}{8}} \approx 2^{\frac{7}{4}}(p^6)^{\frac{1}{8}} = 2^{\frac{7}{4}}p^{\frac{3}{4}}$ because $a_1, a_2, a_3, a_4, a_5, \eta_1, \eta_2, \eta_3, \eta_4 \approx p$. We also have the approximation as follows:

$$||\mathbf{V}_0|| = \sqrt{v_0^2 + (u_0 + a_2 w_0 + a_3 x_0 + a_4 y_0 + a_5 z_0)^2 + p^2} \approx p^{\frac{6}{5}}.$$

Obviously $p^{\frac{6}{5}} > p^{\frac{3}{4}}$. As a result $\mathbf{V}_0$ will not be among the short vectors output by the LLL algorithm. The LLL algorithm outputs vectors $\mathbf{V}$ where $||\mathbf{V}|| \ll ||\mathbf{V}_0||$. Our experiments confirm this observation. The following is an example.

Example for 3.7 : Analysis - 2 This is an illustration of executing the LLL algorithm upon MHLERP for the case of Analysis-2. Let $n = 32$. We will use the parameters:

1. $u_0 = 21$
2. $v_0 = 95$
3. $w_0 = 103$
4. $x_0 = 87$
5. $y_0 = 74$
6. $z_0 = 86$
7. $a_1 = 3193415427$
8. $a_2 = 2205633754$
9. $a_3 = 3080513063$
10. $a_4 = 2991853793$
11. $a_5 = 3225880586$
12. $p = 3239617301$
13. $c = 1535369407$
14. $\eta_1 = 1716845720$
15. $\eta_2 = 2660591034$
16. $\eta_3 = 1267372013$
17. $\eta_4 = 447448661$

Consider the lattice $\mathcal{L}2$ with the matrix $M_{\mathcal{L}2}$ as discussed in Analysis-2 with the above parameters. Executing the LLL upon $M_{\mathcal{L}2}$ produces:

$$M_{\mathcal{L}2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 11219 & 56246 & 0 \\ -60232 & -13209 & 0 \\ -4030 & 3344 & 3239617301 \end{bmatrix}$$

We have $c - M_{\mathcal{L}2}(8,2) - a_1 M_{\mathcal{L}2}(8,1) \equiv 0 \pmod{p}$.

Let $\mathbf{V}_0 = (v_0, u_0 + a_2 w_0 + a_3 x_0 + a_4 y_0 + a_5 z_0, p)$ and $\mathbf{V}_1 = (M_{\mathcal{L}2}(7,1), M_{\mathcal{L}2}(7,2), p)$. We can see that $||\mathbf{V}_0|| \approx 994013103422$ while $||\mathbf{V}_1|| \approx 3239617301$. That is, LLL produces a much shorter vector than $\mathbf{V}_0$. Our experiments produce such observation.

4 The KAZ Key Encapsulation Mechanism (KAZ KEM)

We now put forward our scheme, the KAZ Cryptosystem (KAZ). KAZ is a novel design where its ciphertext is a direct implementation of MHLERP, while the underlying difficulty of the KAZ key equation will be discussed in Sections 5.1, 5.2 and 5.4. Let $\ell(\cdot)$ be a function that outputs length of binary string of input. From the given security parameter, κ determine k (see Section 5.2 and table 1). Next generate a list of k primes, $\mathbf{P} = \{p_i\}_{i=1}^{k}$. Then compute $N_1 = \prod_{i=1}^{k} p_i$ and $n_1 = \left\lfloor \frac{\ell(N_1)}{2} \right\rfloor$. The KAZ system parameters are $(\mathbf{P}, N_1, n_1, k)$.

4.1 The KAZ Key Generation Algorithm

The following describes the key generation procedure for KAZ.

Algorithm 1 : KAZ.Key_Gen algorithm

Input: System parameters, $(\mathbf{P}, N_1, k)$.

Output: Public keys $(e_1, e_2, e_3, e_4, e_5)$, private keys (p, N_2) and public parameter n_2.

1: Generate integer $N_2 = \prod_{i=1}^{k} p_i^{a_i}$ where a_i is chosen randomly from $\{0, 1\}$. Ensure $\beta \in (0.505, 0.51)$ where $N_2 \approx N_1^{\beta}$. Else, repeat this step.
2: Compute $n_2 = \ell(N_2)$.
3: Generate a random prime $p \approx 2^{0.3n_2}$.
4: Generate random primes $q, r, s, t, u \approx 2^{0.3n_2-2}$.
5: Generate random integers $t_1, t_2, t_3, t_4, t_5 \approx 2^{2n_1}$.
6: Calculate $e_1 = pq + N_2 t_1 \pmod{N_1}$.
7: Calculate $e_2 = pr + N_2 t_2 \pmod{N_1}$.
8: Calculate $e_3 = ps + N_2 t_3 \pmod{N_1}$.
9: Calculate $e_4 = pt + N_2 t_4 \pmod{N_1}$.
10: Calculate $e_5 = pu + N_2 t_5 \pmod{N_1}$.
11: Output public keys $(e_1, e_2, e_3, e_4, e_5)$, private keys (p, N_2) and public parameter n_2.

4.2 KAZ Encapsulation and Decapsulation Algorithms

The following algorithm encapsulates the secret parameter given by the relation $sk = x_0^3(e_1x_1 + e_2x_2 + e_3x_3 + e_4x_4 + e_5x_5) \pmod{N_1}$.

Algorithm 2 : KAZ.Encaps algorithm

Input: System parameters N_1, public keys $(e_1, e_2, e_3, e_4, e_5)$.

Output: The ciphertext, c.

1: Generate a random $x_0 \approx 2^{0.3n_2-2}$.
2: Generate a random $x_1 \approx 2^{0.4n_2-3}$.
3: Generate a random $x_2 \approx 2^{0.4n_2-3}$.
4: Generate a random $x_3 \approx 2^{0.4n_2-3}$.
5: Generate a random $x_4 \approx 2^{0.4n_2-3}$.
6: Generate a random $x_5 \approx 2^{0.4n_2-3}$.
7: Compute ciphertext $c \equiv x_0 + e_1x_1 + e_2x_2 + e_3x_3 + e_4x_4 + e_5x_5 \pmod{N_1}$.
8: Output ciphertext c.

The following algorithm decapsulates from the ciphertext c the secret parameter $sk \equiv x_0^3(e_1x_1 + e_2x_2 + e_3x_3 + e_4x_4 + e_5x_5) \pmod{N_1}$.

Algorithm 3 : KAZ.Decaps algorithm

Input: Ciphertext c, private keys (p, N_2).

Output: $sk \equiv x_0^3(e_1x_1 + e_2x_2 + e_3x_3 + e_4x_4 + e_5x_5) \pmod{N_1}$.

1: Compute $Y_0 \equiv c \pmod{N_2}$. Note here that $Y_0 \equiv x_0 + x_1pq + x_2pr + x_3ps + x_4pt + x_5pu \in \mathbb{Z}$.
2: Compute $Y_1 \equiv Y_0 \pmod{p}$. Note here that $Y_1 = x_0 \in \mathbb{Z}$.
3: Compute $Y_2 \equiv x_0^3(c - Y_1) \pmod{N_1}$.
4: Output $sk = Y_2$.

Proposition 1. *KAZ.Decaps decapsulates correctly and without failure.*

Proof. From parameter selection we have $Y_0 = x_0 + x_1pq + x_2pr + x_3ps + x_4pt + x_5pu \approx 2^{n_2-5}$. Hence, $Y_0 < N_2 \approx 2^{n_2}$. Since $x_0 < p$ we can compute $Y_1 = x_0 \in \mathbb{Z}$ without modular reduction. Finally, we obtain $sk \equiv Y_1^3(c - Y_1) \equiv x_0^3(e_1x_1 + e_2x_2 + e_3x_3 + e_4x_4 + e_5x_5) \pmod{N_1}$ without failure.

5 KAZ Key Problem

This section discusses total break attempts upon the KAZ key equation as provided in Algorithm 1. That is, one has to identify the secret parameters, namely the pair (p, N_2) or any other candidate (p', N_2') that decapsulates correctly.

5.1 Coppersmith methodology and the KAZ Key Problem

We will observe the following theorem from May [12].

Theorem 2. *Let N be an integer of unknown factorization, which has a divisor $b \geq N^{\beta}$. Let $f_b(x)$ be an univariate, monic polynomial of degree δ. Furthermore, let c_N be a function that is upper-bounded by a polynomial in $\log N$. Then, we can find all solutions x_0 for the equation $f_b(x) = 0 \pmod{b}$ with $|x_0| \leq c_N N^{\frac{\beta^2}{\delta}}$ in time polynomial in $(\log N, \delta)$.*

We note that we have the integer N_1 which has an unknown factor $N_2 \approx N_1^{0.5}$. The univariate, monic polynomial to be solved is $f_{N_2}(x) = e_i - x_i = 0 \pmod{N_2}$ where $i = 1, 2, 3, 4, 5$. Thus, we have $\beta = 0.5$ and $\delta = 1$. We now have the following fact, $N_1^{0.25} \approx N_2^{0.5}$. Since for each i we have the roots $x_{0,i} \approx N_2^{0.6}$, it is clear that $x_{0,i} > N_1^{0.25}$. This renders Coppersmith's method to obtain the roots from KAZ key equations impractical.

5.2 Blackburn's Combinatorial Approach Solving the KAZ Key Problem

The following methodology is due to a strategy proposed by Blackburn [2]. The strategy focuses on reducing the complexity of finding N_2. It is currently the best strategy available to solve the KAZ key problem.

We begin with the observation of $N_2 = \prod_{i=1}^{k} p_i^{a_i}$ where $a_i \in \{0, 1\}$ is randomly chosen. We mention here again that the distinct prime factors of r with exponent value 0 (θ_1 elements) is 50% of the primes in the list $\mathbf{P}$ and the distinct prime factors of N_2 with exponent value 1 (θ_2 elements) is also 50%. The process is executed upon a pair of keys, such as upon e_1 and e_2. The process is as follows:

Step 1: We choose $I \subseteq \{1, 2, ..., k\}$ so that $N_I = \prod_{i \in I} p_i$ is a number at least $0.6n_2 + 1$ bits and hope that N_I divides N_2.

Step 2: Compute $e_1 \equiv (pq)' \pmod{N_I}$. Check if $(pq)' \approx 2^{0.6n_2}$. Otherwise return to Step 1.

Step 3: Define $I' \subseteq \{1,2,...,k\}$ by $i \in I'$ if and only if $e - (pq)' \equiv 0 \pmod{p_i}$ Note that $I' \subseteq I$.

Step 4: Return any n_2 bit integer $M = \prod_{i \in I'} p_i$ if M is big enough to have N_I as a divisor. Otherwise return to Step 1.

Step 5: Compute gcd $((pq)', e_2 \pmod{M}) = p'$, since $e_2 \equiv (pr)' \pmod{M}$. Check if $p' \approx 2^{0.3n_2}$. Otherwise return to Step 1.

Proof of Correctness

To see why this approach works, first note that our condition for membership of I' implies that $e_1 \equiv (pq)' \pmod{M}$ and $e_2 \equiv (pr)' \pmod{M}$ where $(pq)'$ and $(pr)'$ is within the prescribed interval as mentioned in Algorithm 1. So if the algorithm returns a value, it is a solution to the KAZ key problem. ■

Complexity

Let μ be the average length of primes in the list $\mathbf{P}$. We know that from the size of N_1, we have the approximation $2n_1 \approx k\mu$. The number of distinct primes that construct N_2 is given by $\theta_2 \approx \left\lceil \frac{n_2}{\mu} \right\rceil = \left\lceil \frac{kn_2}{2n_1} \right\rceil$. Since $n_1 \approx n_2$, we have $\theta_2 \approx \left\lceil \frac{k}{2} \right\rceil$. The number of distinct primes that construct N_I is given by $\theta_I \approx \left\lceil \frac{0.6n_1}{\mu} \right\rceil$. This in turn means $\theta_I \approx \left\lceil \frac{0.6k}{2} \right\rceil$. The combination probability problem statement now would be: Given a set $\mathbf{P}$ that contains θ_2 elements that constructs N_2 and $k - \theta_2$ elements that does not construct N_2, what is the probability of selecting $\left\lceil \frac{0.6k}{2} \right\rceil$ elements that constructs N_I? Let $C_1 = \binom{\theta_2}{\left\lceil \frac{0.6k}{2} \right\rceil}$ and $C_2 = \binom{k}{\left\lceil \frac{0.6k}{2} \right\rceil}$. The final complexity is $O\left(\frac{C_1}{C_2}\right)$. So our guess for I will be correct with probability approximately $\frac{C_1}{C_2}$. Thus this approach will take about $O\left(\frac{C_2}{C_1}\right)$ guesses to respond correctly to the problem. Since each guess requires about k arithmetical operations, the expected complexity of the algorithm is about $kO\left(\frac{C_2}{C_1}\right)$.■

We now can have an entropy table as below, where $\kappa = \left\lceil \log_2 \frac{C_1}{C_2} \right\rceil$.

k	κ
128	50
256	100
325	128
650	256

Table 1. KAZ.Key_Gen κ-bit entropy (i.e. 2^κ)

From Table 1, one can deduce that, we need to utilize 325 primes within the list $\mathbf{P}$ to obtain 128-bits security. If the first 325 primes greater than 2 are used, the public key N_1 will be of length 3052 bits and the private key N_2 length would be approximately 1500 bits. The following an example of Blackburn's methodology.

5.3 Example for Blackburn's Methodology

This is an example of Blackburn's combinatorial approach to solve the KAZ Key Problem.

We will use the first $k = 48$ primes larger than 2 which provides the following parameters:

1. $N_1 = 416556045624021725187814231899405191205673355204301573273089888740386463208163952286675 55$
2. $N_2 = 1004040945641498462345649482330325172068931013$
3. $p = 2884076984086464562 9039$
4. $q = 6823$
5. $PQ = pq = 196780572624219477126933097$
6. $t_1 = 298417731225723356904011708887482104512621175938810960035810610121519333088212570425286 48$
7. $e_1 = 775682003919689465568865926467451302970618897399653246422959106425223309065272765434735 1$

Assume we are able to obtain N_I =(7)(13)(17)(19)(31)(47)(61)(67)(71)(83)(97)(101)(103)(109)(113)(131) where $N_I | N_2$. We have $e_1 \equiv (PQ)' = 196780572624219477126933097 \pmod{N_I}$. Since $(PQ)'$ is in the prescribed interval, we proceed to identify the other primes, as illustrated in the table below.

Primes, $\{p_i\}$	$e - (PQ)' \equiv 0 \pmod{p_i}$
137,149,151,157,163,173,193,227	YES
3,5,11,23,29,37,41,43,53,59,73,79,89,107, 127,149,167,179,181,191,197,199,211,223	NO

The process is then continued by computing $\gcd((PQ)', e_2 \pmod{N_2}) = p$, since $e_2 \equiv pr \pmod{N_2}$.

5.4 Lattice Analysis on KAZ Keys $\{e_i\}_{i=1}^{5}$

Consider the lattice $\mathcal{L} = \{\mathbf{x} \in \mathbb{Z}^5 | e_1x_1 + e_2x2 + e_3x3 + e_4x_4 + e_5x_5 = 0 \pmod{N_1}\}$. We have the Gaussian heuristic $\sigma(\mathcal{L}) \approx N_1^{\frac{1}{6}} \approx 2^{0.3n_2}$. Executing the LLL algorithm on $\mathcal{L}$ will produce 3 basis, where the norm is less than $2^{0.3n_2}$. Next, we can observe the following:

$$x_1pq + x_2pr + x_3ps + x_4pt + x_5pu = 0$$

which can be interpreted as

$$x_1 q + x_2 r + x_3 s + x_4 t + x_5 u = 0 \tag{3}$$

We can also observe the following:

$$x_1 N_2 t_1 + x_2 N_2 t_2 + x_3 N_2 t_3 + x_4 N_2 t_4 + x_5 N_2 t_5 \equiv 0 \pmod{N_1}$$

which can be interpreted as

$$x_1 t_1 + x_2 t_2 + x_3 t_3 + x_4 t_4 + x_5 t_5 \equiv 0 \quad (\mathrm{mod}\ \frac{N_1}{N_2}) \tag{4}$$

Equation 3 will refer to the lattice $\mathcal{L}_1$, while equation 4 witll refer to the lattice $\mathcal{L}_2$. From $\mathcal{L}_1$ we have the Gaussian heuristic $\sigma(\mathcal{L}_1) \approx u^{\frac{1}{5}} \approx 2^{0.06n_2}$. While from $\mathcal{L}_2$ we have the Gaussian heuristic $\sigma(\mathcal{L}_2) \approx (\frac{N_1}{N_2})^{\frac{1}{6}} \approx 2^{0.16n_2}$.

Observe that the desirable vector from $\mathcal{L}_2$ - $\mathbf{V}_{\mathcal{L}_2}$, has its norm $||\mathbf{V}_{\mathcal{L}_2}|| = (t_1, t_2, t_3, t_4, t_5) \approx 2^{2n_2}$. This is much larger than $\sigma(\mathcal{L}_2) \approx 2^{0.16n_2}$. Hence, referring to Remark 1, the use of the LLL algorithm is insignificant.

As such, we will analyse $\mathcal{L}_1$. Furthermore, upon identifying (q, r, s, t, u), one can proceed to identify N_2 with ease.

Now consider the lattice $\mathcal{L}_{12}$ with the matrix $M_{\mathcal{L}_{12}}$ given by:

$$M_{\mathcal{L}_{12}} = \begin{bmatrix} 1 & 0 & 0 & 0 & x_1 \\ 0 & 1 & 0 & 0 & x_2 \\ 0 & 0 & 1 & 0 & x_3 \\ 0 & 0 & 0 & 1 & x_4 \\ 0 & 0 & 0 & 0 & x_5 \end{bmatrix}$$

The Gaussian heuristic is $\sigma(\mathcal{L}_{12}) \approx x_5^{\frac{1}{5}} \approx 2^{0.06n_2}$. That is we take the largest possible interpretation for x_5 since x_5 is derived via the LLL algorithm over the lattice $\mathcal{L}$, and the Gaussian heuristic is $\sigma(\mathcal{L}) \approx 2^{0.3n_2}$. Let $\mathbf{V}_0$ be a vector of $\mathcal{L}_{12}$. Then there exists the 5-tuple $(y_1, y_2, y_3, y_4, y_5) \in \mathbb{Z}^5$ such that $\mathbf{V}_0 = (y_1, y_2, y_3, y_4, y_5) M_{\mathcal{L}_{12}} = (y_1, y_2, y_3, y_4, x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 + x_5 y_5)$ is on the lattice $\mathcal{L}_{12}$. More precisely the lattice contains the vector solution $\mathbf{V}_0 = (q, r, s, t, 0)$.

Observe from KAZ Cryptosystem parameter selection we have

$$\begin{aligned} ||\mathbf{V}_0|| &= \sqrt{q^2 + r^2 + s^2 + t^2} \\ &\approx 2^{0.3n_2} \end{aligned}$$

Obviously $2^{0.3n_2} > 2^{0.06n_2}$. Then referring to Remark 1, the use of the LLL algorithm is insignificant. On the other hand, if one chooses $(q, r, s, t, u) \approx 2^{\epsilon n_2}$ where $\epsilon < 0.06$ (or maybe $\epsilon \approx 0.06$), the LLL algorithm will output the values (q, r, s, t)

and continuing one will obtain u. Upon obtaining the values (q, r, s, t, u), the private key p and N_2 will be easily extracted.

An in-depth view upon equation 3 given by the relation $f(q, r, s, t, u) = c = x_1 q + x_2 r + x_3 s + x_4 t + x_5 u = 0$ will give information that since $q, r, s, t, u \approx 2^{0.3n_2}$ and $x_1, x_2, x_3, x_4, x_5 \approx 2^{0.3n_2}$, we have $c \approx 2^{0.6n_2}$. This implies $q, r, s, t, u \approx c^{0.5}$. Hence, we have the following proposition.

Proposition 2. *The problem to solve KAZ Key Problem via equation 3 reduces to the MPLERP.*

Proof. Upon solving the MPLERP the parameters (q, r, s, t, u) are obtained. Thus, KAZ Key Problem via 3 reduces to the MPLERP.■

Remark 4. The converse of proposition 2 is still unknown.

5.5 The KAZ Key Equation and Grover's Algorithm

Grover's algorithm is a quantum algorithm that finds with high probability the unique input to a black box function that produces a particular output value, using just $O(\sqrt{N})$ evaluations of the function, where N is the size of the function's domain ([7]). Thus, in order to achieve 128-bit post quantum security against Grover's algorithm, a total of 650 primes must be used from the list **P**. If **P** is the list of the first 650 primes larger than 2, then N_1 will be approximately 6865 bits and N_2 will be approximately 3400 bits. Since both KAZ encapsulation and decapsulation procedures has low computational complexity (at most multiplication), KAZ still operates at a desirable speed.

6 The KAZ Problem : One-Wayness of KAZ

We now formally define the KAZ Problem. Upon given the KAZ public parameters $(e_1, e_2, e_3, e_4, e_5, N_1)$ and KAZ ciphertext $c \in \mathbb{Z}_{N_1}$, one needs to output u, where the unknown variable size is as specified in Algorithm 2. Upon obtaining u, one can obtain $sk \equiv u^3(c - u) \pmod{N_1}$.

6.1 KAZ Problem reduces to KAZ Key Problem

Proposition 3. *The KAZ Problem reduces to the KAZ Key Problem.*

Proof. Upon solving the KAZ Key Problem, the secret parameters (p, N_2) are obtained. Then the ciphertext can be decapsulated. Thus, KAZ Problem is reduced to the KAZ Key Problem.

6.2 KAZ Problem reduces to MHLERP

Observe the in-depth view upon the KAZ ciphertext c given by the relation $f(u,v,w,x,y,z) = u + e_1v + e_2w + e_3x + e_4y + e_5z \pmod{N_1}$. We know that from Algorithm 1, $e_1, e_2, e_3, e_4, e_5 \approx N_1$. We also know that from Algorithm 2, we have $u \approx N_2^{0.3} \approx N_1^{\frac{3}{20}}$ and $v, w, x, y, z \approx N_2^{0.4} \approx N_1^{\frac{1}{5}}$. Thus, we have the following proposition.

Proposition 4. *The KAZ Problem reduces to the MHLERP.*

Proof. Upon solving the MHLERP the parameters (u, v, w, x, y, z) are obtained. Then compute $sk = u^3(c - u) \pmod{N_1}$. Thus, KAZ Problem is reduced to the MHLERP.

Remark 5. The converse of propositions 3 and 4 is still unknown.

6.3 KAZ Ciphertext Equation and Grover's Algorithm

The best case scenario is to conduct exhaustive search from the ciphertext c the secret parameter u to solve the KAZ Problem. It takes at most $2^{0.3n_2}$ searches. With Grover's algorithm the complexity is reduced to $\approx 2^{0.15n_2}$ ([7]). For 256-bit security (i.e. KAZ Key Problem is 128-bit secure against Grover's algorithm) we have $n_2 \approx 3400$. Thus, the complexity is $\approx 2^{510}$.

7 Conclusion

In this work we have utilized the modular hexavariate linear equation root problem to design a key encapsulation mechanism. It is proven analytically that all current strategies to either extract the private key from the public key or the secret information from the ciphertext will incur exponential running time complexity. We also show that the KAZ KEM can achieve 128-bit security with each public key length of approximately 3052 bits. We also can observe that the plaintext-to-ciphertext expansion is 1-to-1, since one ciphertext $c \in \mathbb{Z}_{N_1}$ is needed to send $sk \in \mathbb{Z}_{N_1}$. That is, the ciphertext is the same size of the information being relayed. Furthermore, we have proven there is no decryption failure. With complexity running time $O(n^2)$ (where n is the length of the input) for both encryption and decryption, the KAZ KEM has desirable speed for any practical application. It can be seen that KAZ KEM utilizes "simple" mathematics in order to achieve maximum simplicity in design, such that even practitioners with limited mathematical background will be able to understand the arithmetic. Indeed, the KAZ KEM can be a seamless post quantum drop-in replacement for traditional asymmetric cryptosystems.

Acknowledgements

We acknowledge Moesfa Soeheila Mohamad from MIMOS Malaysia for her contributions, opinions and comments in making this article in its current form.

References

1. Miklòs Ajtai. The shortest vector problem in L_2 is **NP**-hard for randomized reductions. In *In ACM Symposium on Theory of Computing*, pages 10–19. ACM New York, NY, USA, 1998.
2. Simon R. Blackburn. A report on a new "hard to solve" modular equation. Private communication, 2016.
3. Daniel Bleichenbacher and Alexander May. New attacks on RSA with small secret CRT-exponents. In *In Public Key Cryptography*, pages 1–13. Springer, 2006.
4. Stanislav Bulygin, Albrecht Petzoldt, and Johannes Buchmann. Towards provable security of the unbalanced oil and vinegar signature scheme under direct attacks. In *Progress in Cryptology - INDOCRYPT 2010*, pages 17–32. Springer, 2010.
5. Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *Applied Cryptography and Network Security (ACNS) 2005*, pages 164–175. Springer, 2005.
6. Marc Girault, Philippe Toffin, and Brigitte Vallée. Computation of approximate l-th roots modulo n and application to cryptography. In *Crypto*, volume 88, pages 100–117. Springer, 1988.
7. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.
8. Mathias Herrmann and Alexander May. Solving linear equations modulo divisors: On factoring given any bits. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 406–424. Springer, 2008.
9. Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. NTRU: A ring based public key cryptosystem. In *Algorithmic Number Theory (ANTS III)*, pages 267–288. Springer, 1998.
10. Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. *An Introduction to Mathematical Cryptography*, volume 1. Springer, 2008.
11. A K Lenstra, H W Lenstra, and L Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
12. Alexander May. *New RSA vulnerabilities using lattice reduction methods.* PhD thesis, University of Paderborn, 2003.
13. Robert J McEliece. A public-key cryptosystem based on algebraic coding theory. In *DSN Progress Report*, volume 44, pages 114–116, 1978.
14. Phong Q. Nguyen. Can we trust cryptographic software? Cryptographic flaws in GNU privacy guard v1.2.3. In *EUROCRYPT*, pages 555–570. Springer, 2004.
15. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *The 37th ACM Symposium on Theory of Computing (STOC 2005)*, pages 84–93. Association of Computing Machinery, 2005.
16. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26:1484–1509, 1994.