Evaluation of Alternative NTRU Modular Inverse Solutions

Juliet N. Gaithuru, *Member, IAENG,* Mazleena Salleh, *Member, IAENG,* and Nor Muhainiah

***Abstract*—The NTRU public key cryptosystem key generation algorithm requires the computation of two modular polynomial inverses whereby it is difficult to predict whether an inverse exists for any randomly chosen private polynomial. In this study, we use the ring homomorphism from the NTRU lattice to a circulant matrix to find the solution of a system of simultaneous linear equations in order to solve the modular inverse problem. We propose and evaluate alternative modular inverse solutions, namely naïve matrix inversion, Strassen-type matrix inverse, Gauss Jordan elimination with partial pivoting, LU decomposition, Gram Schmidt orthogonalization and least squares method. The least squares method is identified as the best solution following a performance comparison of these alternative solutions in terms of the probability of finding an inverse, speed of inversion and computational time complexity. The least squares method applies the Pseudo-Inverse in finding the least squares solution to the NTRU modular inverse. It has the highest probability of finding an inverse, resulting in a 50% increase in the number of invertible polynomials (compared to the classical inverse algorithm) thereby doubling the size of the key space with a trade-off in computational cost.**

***Index Terms*—public key cryptography, NTRU, modular polynomial inverses, Pseudo-Inverse, least squares, matrix inversion, Gauss Jordan elimination, Gram Schmidt orthogonalization.**

I. Introduction

The internet has become an indispensable part of today's society. The internet facilitates essential communication, commerce, social connections along with a myriad of means for information exchange. All these interactions need to be secured in order to facilitate secure commerce, communications and protection of personal data [1]. It is imperative that these large volumes of data are scalably stored and secured [2]. Cryptography is a crucial tool for protecting this information in computer systems and over the internet.

Once quantum computers become commercially available, classical cryptography primitives such as RSA and Elliptic Curve Cryptography will be obsolete following Shors [3] revelation that the discrete logarithm and integer factorization problems can be effectively solved using polynomial time quantum computing algorithms [4].

Post-quantum cryptography offers secure alternatives for public key cryptography that are quantum secure. Post-quantum cryptography is based on the principles of lattices, codes, hashes, multivariate quadratic polynomials and isogenies [5]. In this study we pay particular attention to NTRU [6], a lattice-based public key cryptosystem.

This paper looks into the key generation process of the NTRU public key cryptosystem, with particular focus on the modular inverse algorithm. Key generation in NTRU involves the computation of two modular polynomial inverses whereby there is an inherent difficulty in predicting the existence of an inverse [7] for any randomly chosen private polynomial. In case an inverse is not found, the classical NTRU inverse algorithm goes into an infinite loop of trying to find an inverse, thus no inverse is found, no private key is generated and as a consequence the message encryption process is halted. Previous research studies propose alternative modular inverse algorithms, such as [8], [7], [9], [10] in an effort to counteract this problem. This study embarks on finding a solution to this modular inverse problem by considering a range of possible alternative inverse algorithms using varied algebraic structures and finally zeroing in on the solution with the most favourable results and improved chances of finding a modular polynomial inverse.

Given that NTRU is a lattice-based public key cryptosystem, we use an adaptation of the circulant matrix for the inverse polynomial algorithm. Based on the assertion by [11] that any circulant matrix can effectively be expressed as a polynomial function, along with the concept of a ring homomorphism (a mapping that preserves both additions and multiplications [12]), we use matrices to solve a system of simultaneous linear equations thereby solving the modular inverse problem by considering alternative inverse algorithms. These alternative NTRU inverse algorithms are discussed and presented in subsequent sections, along with a comparative evaluation of their performance against that of the classical NTRU inverse algorithm [13], [6], [14]. The main objective of this study is to propose an NTRU inverse algorithm which provides for the highest probability of finding a modular polynomial inverse for any randomly chosen private polynomial. This will effectively enlarge the key space thereby enhancing the algorithm's security along with eliminating the need to have a parameter selection algorithm since any private polynomial can be inverted.

This paper presents a background on the NTRU public key cryptosystem, a discussion of the NTRU inverse algorithm and the mathematical background to the solutions evaluated in this study in Section II. This is followed by an evaluation of the alternative modular inverse solutions in Section III. A performance comparison of the alternative modular inverse solutions is provided in Section IV eventually followed by a conclusion in Section V.

II. Background

NTRU is a public key cryptosystem which was developed and presented in the rump session of CRYPTO '96 and later patented by Hoffstein, Pipher and Silverman (1998) [6]. NTRU is taken to be an abbreviation for *Number Theorists aRe Us* or *Number Theory Research Unit* by other researchers [15].

Manuscript received January 15, 2021; revised July 3, 2021.

Juliet N. Gaithuru is a PhD candidate of the School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Skudai 81310, Johor, Malaysia; phone: +971508341140; e-mail: julietgaithuru@gmail.com.

Mazleena Salleh is an Associate Professor at the School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Skudai 81310, Johor, Malaysia; e-mail: mazleena@utm.my.

Nor Muhainiah is an Associate Professor at the Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, Skudai 81310, Johor, Malaysia; e-mail: normuhainiah@utm.my.

NTRU is a ring-based cryptosystem which is made up of polynomials with integer coefficients. It is defined over the polynomial ring $R = \frac{Z[x]}{(x^N-1)}$ based on the three key integer parameters N, p and q where $N > 1$, $q > p$ and the integers p and q are relatively prime. R is the set of convolution polynomials which have a degree $(N-1)$ [6]. The degree of polynomials in R is reduced modulo $(x^N - 1)$ while the polynomials' integer coefficients are reduced modulo p or q.

NTRU cryptosystem is comprised of three services: NTRUSign, NTRUKE and NTRUEncrypt which are a family of signature schemes with a structure similar to the NTRU lattice, a key exchange protocol and encryption algorithm, respectively.

NTRUEncrypt was made available by Security Innovation on an open-source General Public License (GPL) free for non-commercial use making its application possible on open source products such as web browsers and TLS/SSL (Transport Layer Security/ Secure Socket Layer) servers [16].

The basic framework of NTRUEncrypt allows for great variation in the types of parameter sets generated. Polynomials utilized in the cryptosystem may be binary, trinary/ternary, or product-form. Ternary polynomials provide a better trade-off between security and efficiency, as opposed to the binary polynomials; while binary polynomials provide for a smaller key size [17]. This paper focuses on NTRUEncrypt which will simply be referred to by the term NTRU in this paper.

Operation in classical NTRU begins with the establishment of the integers N, p, q and the polynomials f, g, r and m. The private key is then generated by first obtaining the multiplicative inverse of $f \bmod p$ and $f \bmod q$ such that: $f * F_p \bmod p = 1$ and $f * F_q \bmod q = 1$. The private key is then set as the polynomial pair (f, F_p). The public key h is obtained by computing $h = p * F_q * g \bmod q$. The message m is encrypted using the public key h by computing $e = r * h + m \bmod q$. The ciphertext is decrypted by the recipient using the private key by computing $a = f * e \bmod q$. After which adjustment is done to a by ensuring that the coefficients of a are in the range of $\frac{q}{2}$ and $\frac{-q}{2}$. This is then followed by retrieving the decrypted message by computing $C = F_p * a \bmod p$.

Decryption is made possible because the polynomials p, r, g and m are chosen to have small values in the polynomial convolution ring R thus ensuring the polynomial $p*r*g+f*m$ has a high probability of having width b (which represents $p * r * g + f * m$) less than q. The coefficients of these terms are selected in a manner that ensures that their absolute value does not exceed the range $\frac{q}{2}$ and $\frac{-q}{2}$ [17], [18].

A. NTRU Inverse Algorithm

The NTRU key generation process requires the computation of the multiplicative inverse of the private polynomial f modulo a small integer p and a large integer q, given by F_p and F_q respectively. These polynomials' coefficients are in Z/pZ and Z/qZ respectively. These inverses F_p and F_q are in the rings R_p, R_q and reduction is done modulo $(x^N - 1)$, in order to ensure the parameter size N is not exceeded [13], [6], [14].

Definition 1. [19] Z/pZ and Z/qZ are described as the set of residue classes modulo p and q respectively. A residue class $t \bmod p$ is defined as the equivalence class of results from adding integer multiples of p, that is $t + pZ$. The class has p elements based on the number of possible remainders $0, \ldots, p-1$. For example, the residue classes of modulo 3 are $0 + 3Z, 1 + 3Z, 2 + 3Z$.

A description of the algorithm used to compute the inverses in NTRU is provided in the NTRU Technical Report [20]. The report describes the algorithm used to find the inverse of a polynomial $a(x)$ in the ring Z/pZ and Z/qZ where $p = 2$, $p = 3$ and modulo power p^r (where r is an integer). The algorithm used is an adaptation of the "Almost Inverse Algorithm" of [21] provided in [20]. The report describes separate algorithms for:

i. Inverse of a polynomial (x) in the ring $\frac{Z/2Z[x]}{x^N-1}$.
ii. Inverse of a polynomial (x) in the ring $\frac{Z/3Z[x]}{x^N-1}$.
iii. Inverse of a polynomial (x) in the ring $\frac{Z/2^rZ[x]}{x^N-1}$ where 2^r stands for an integral power of 2.
iv. Inverse of a polynomial (x) in the ring $\frac{Z/pZ[x]}{x^N-1}$ where p is an arbitrary prime.

The IEEE P1363.1 standard [22], EESS 1 Version 3.0 [23], and [24] also provide a description of the polynomial inverse algorithm described in [20]. The algorithm is also cited in [25], [26]. [27] also provide inverse polynomial algorithms in F_p and F_q based on the algorithms in [20]. The algorithm contains a combination of the division algorithm, a variation of the Extended Euclidean algorithm applicable to an integer ring and the process of finding possible inverses in the convolution ring.

The classical NTRU inverse polynomial algorithm in F_q is illustrated in Algorithm 1 [27], [23], [20], [28], [24]. The same algorithm is used to find the inverse polynomial in F_p by replacing input parameters $(a(x), N, q)$ with $(a(x), N, p)$.

B. Solution Background

In this study, the ring homomorphism of the ring R and an $N \times N$ circulant matrix with integer coefficients is considered. This is based on the concept of the structure of the NTRU lattice [29], [30] and an adaptation of the circulant matrix for the inverse polynomial algorithm. This is supported by the assertion by [11] that any circulant matrix can be expressed as a linear combination of the set of basis matrices $I, K, \cdots, K^{N-1}$ thereby facilitating the expression of F as a polynomial function given by Equation (1).

$$\begin{aligned} F &= f_0 I + f_1 K + \cdots + f_{N-1} K^{N-1} \\ &= f(K). \end{aligned} \tag{1}$$

A ring homomorphism from R to $\acute{R}$ is defined as a mapping θ from a ring R to $\acute{R}$ such that for all r and s in R

$$\theta(r + s) = \theta(r) + \theta(s) \text{ and } f(xy) = f(x)f(y) \tag{2}$$

which is otherwise denoted as $\theta : R \to \acute{R}$. Ring homomorphism is a mapping that preserves both additions and multiplications [12].

Under this homomorphism, the polynomial $f(x) = \sum_i f_i x^i$ maps onto the matrix F of dimension $(N \times N)$, where f_i are the coefficients of $f(x)$ and $0 \leq i \leq (N-1)$. An N-dimensional complex vector space is created by generating a circulant matrix which is defined by its first row [31],

Algorithm 1 Classical inverse polynomial algorithm in F_q

procedure INVERSE POLYNOMIAL $F_q(a(x), b(x))$
  Input: $a(x), N, q$
  Output: F_q
  Initialize $k = 0, b = 1, c = 0, f(x) = a(x), g = 0$ ▷ Steps 5-6 set $g(x) = x^N - 1$.
  $g[0] = -1,\ g[N] = 1$
  loop
    while $f[0] = 0$ **do**
      for $i = 1$ to N **do**
        $f[i-1] = f[i]f(x) = f(x)/x$
        $c[N+1-i] = c[N-i]c(x) = c(x) \cdot x$
      end for
      $f[N] = 0,\ c[0] = 0$
      $k = k + 1$
    end while
    if $deg(f) = 0$ **then**
      goto Step 32
    end if
    if $deg(f) < deg(g)$ **then**
      $temp = f$ ▷ Exchange f and g.
      $f = g,\ g = temp$
      $temp = b$ ▷ Exchange b and c.
      $b = c,\ c = temp$
    end if
    $f = f \oplus g,\ b = b \oplus c$
  end loop
  $j = 0$
  $k = k \bmod N$
  for $i = N - 1$ downto 0 **do**
    $j = i - k$
    if $j < 0$ **then**
      $j = j + N$
    end if
    $F_q[j] = b[i]$
  end for
  $v = 2$
  while $v < q$ **do**
    $v = v * 2$
    StarMultiply $(a, F_q, temp, N, v)$
    $temp = 2 - temp \bmod v$
    StarMultiply$(Fq, temp, Fq, N, v)$
  end while
  for $i = N - 1$ downto 0 **do**
    if $F_q[i] < 0$ **then**
      $F_q[i] = F_q[i] + q$
    end if
  end for
  return F_q
end procedure

[32], [33] and each subsequent row is generated by cyclic permutations of the vector with an offset equal to the column index. In this study, we generate F using Algorithm 2.

The aim of the study is to solve the problem of the form $Ax = b$, restated as

$$F\,x_i = I_i \tag{3}$$

$$\begin{bmatrix} f_0 & f_1 & f_2 & \cdots & f_{N-1} \\ f_{N-1} & f_0 & f_1 & \cdots & f_{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_1 & f_2 & f_3 & \cdots & f_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

whereby the circulant matrix F is derived from polynomial f, x_i are the columns of the inverse matrix F^{-1} mod q where q is a large prime and I_i are the columns of the identity matrix I with $0 \leq i \leq (N-1)$.

Algorithm 2 Generating circulant matrix F

procedure CIRCULANT MATRIX$(f(x))$
  Input: $f(x), N$
  Output: F
  Initialize $f = (f_0, f_1, f_2, \ldots, f_{N-1})$
  function CIRCULANTMATRIX(S)
    $v = S$, N= No. of polynomial terms in S
    for $i = (N+1)$ downto 2^t **do** ▷ $2^t > N$
      Append $S \longleftarrow 0$
    end for
    $N = 2^t$
    $S \leftarrow (s_0, s_1, s_2, \ldots, s_{N-1})$
    $v = S,\ t = [\,]$
    for $i = 1$ downto N **do**
      $T = Shift(S, 1)$
      $T \cdot t,\ S = T$
    end for
    $T_1 = t \cdot t,\ Circl(S) = v \cdot T_1$
    return $Circl(S)$
  end function
  $F = circulantMatrix(f)$
  return F
end procedure

III. EVALUATING ALTERNATIVE MODULAR INVERSE SOLUTIONS

In the quest for a befitting modular inverse solution, we evaluated several alternative inverse solutions, namely:

i. Naïve matrix inversion
ii. Strassen-type matrix inverse
iii. Gauss Jordan elimination with partial pivoting
iv. LU decomposition
v. Fast fourier transform
vi. Gram Schmidt orthogonalization
vii. Least squares method.

These alternative inverse solutions are discussed in the subsequent sections, along with a description of their suitability in addressing the inverse problem in the NTRU public key cryptosystem. These solutions were considered on the premise that the NTRU private polynomial could be translated to a representative matrix structure. The circulant matrix was considered based on the fact that the matrix is identified by its first row and every subsequent row is obtained by repetitive shift operations [34]. A circulant matrix $F = Circ(f)$ associated with vector f (the coefficients of polynomial $f(x)$) was used in the subsequent evaluations. The alternative solutions were evaluated in terms of the speed of inversion, correctness of the inverse and the computational complexity of the resulting algorithm.

The ideal objective of this investigation was to find a solution which has the following beneficial features:

i. It allows for the selection of any random polynomial f; thereby eliminating the need for a parameter selection criteria for polynomial f.
ii. Ensures that a modulus inverse F_p and F_q can be obtained regardless of the random polynomial f selected i.e., f will always be invertible.
iii. In the event that f is not invertible modulo p and modulo q, the user should be notified accordingly and an alternative f selected.

A. Naïve Matrix Inversion

The first solution we considered was the naïve matrix inverse. This approach of finding the inverse solution was first proposed in [10] and involves first generating a circulant matrix F from the coefficients of the private polynomial f. This is followed by a computation of the determinant d of the matrix F, after which the matrix of cofactors is computed. The inverse matrix is then obtained by computing the product of the transpose of the matrix of cofactors (otherwise referred to as the adjoint matrix) and the multiplicative inverse of d modulo an integer k (which represents either p or q) all reduced modulo k as given by Equation (4).

$$(Cofactors(F))^T \times (d^{-1} \bmod k) \bmod k = F^{-1} \bmod k \quad (4)$$

The first row of elements in F^{-1} is retrieved as the coefficients of the modular inverse F_k.

This solution yielded correct modular inverses F_k but at the expense of long computation times for matrices of large dimension. This is attributed to the computational complexity of the adjoint operation which takes N^3 iterations.

The algorithm's efficiency was improved by computing the inverse of the first row only and deriving the rest of the matrix by a series of successive shift operations. This method involves finding the adjoint of 1 row, cofactors of one column, determinant of the matrix (arithmetic using cofactors) and eventually finding the inverse matrix at which point only one row is printed. Algorithm 3 shows an implementation of this naïve matrix inversion solution within the NTRU structure whose corresponding computational complexity is O(N^3).

Algorithm 3 Finding NTRU Inverse Polynomial Algorithm Using Naïve Matrix Inversion Method

```
procedure INVERSE POLYNOMIAL ALGORITHM(F, k)
    Input: F, N, k
    Output: f_k
    t = [ ]                                   ▷ Finding Adjoint(F)
    for i = 1 downto N do
        element_i1 = Cofactor(F, i, 1)
        Append element_i1 tot
    end for
    row_i1 = t                    ▷ transposing column 1 to row 1
    for j = 1 downto row_i1 do
        row_i1[j] * F[j, 1]       ▷ products of element and cofactors
    end for
    d = det^-1 mod k
    X = Matrix(Ncolumns1row, row_i1)
    X_0 = d * X                   ▷ To compute d * X mod k
    X_1 = X_0/q
    X_2 = Matrix(X_0 − (X_1 * q))
    InvXmodInt = X_2
    F_row1 = InvXmodInt
    F_inv = ElementToSequence(F_row1)
    F_k = F_inv
    return F_k
end procedure
```

a) Analysis:: The efficient version of the naïve matrix inversion algorithm took a comparatively long time to generate an inverse compared to the classical NTRU inverse algorithm.

B. Strassen-type Matrix Inverse

The second alternative solution considered is the Strassen-type matrix inverse. This approach was implemented and evaluated as follows:

i. Generate the matrix F from the polynomial coefficients of f.
ii. Given that the parameter size N is prime, the matrix is padded with zeros up to the nearest power of 2.
iii. Subsequently, a circulant matrix $F = Circ(f)$ associated with vector f is generated resulting in a dimension which is an integral power of 2 and thus divisible into $\frac{2}{N}$ dimension matrices.
iv. The matrix F is subdivided into four square sub-matrices of dimension $\frac{2}{N}$ up to the (1×1) level and the inverse is computed as follows:
Given that
$$F = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}$$
the inverse F^{-1} is given by
$$F^{-1} = \begin{bmatrix} {F_{11}}^{-1} + F_{12}S^{-1}F_{21}{F_{11}}^{-1} & -{F_{11}}^{-1}F_{12}S^{-1} \\ -S^{-1}F_{21}{F_{11}}^{-1} & S^{-1} \end{bmatrix}$$
where $S = F_{22} - F_{21}{F_{11}}^{-1}F_{12}$ which is referred to as the Schur complement of F_{11}.
v. The inverse algorithm is used recursively and the multiplication is carried out using the Strassens matrix multiplication algorithm [35], [36] resulting in the inverse matrix F^{-1}.
vi. The inverse matrix modulo k is obtained by finding the product of the inverse matrix F^{-1} the determinant and the multiplicative inverse of the determinant modulo k given by
$$F_k = F^{-1} \bmod k = F^{-1} * |F| * d \bmod k \quad (5)$$

This results in the modular inverse matrix F_k. The first row of the resultant inverse matrix F_k gives the polynomial coefficients of the multiplicative inverse of $f \bmod k$ given by $F_k(x)$. An illustration of the integration of this Strassen-type matrix inverse solution with the NTRU structure is given by Algorithm 4 which has a computational complexity of O($\frac{6}{5}N^{2.807}$).

a) Analysis:: This solution yielded correct modular inverses F_k. However, the algorithm had a long computation time for matrices of large dimension. The time taken to find an inverse for a polynomial of parameter size $N = 251$ which translates to a circulant matrix of dimension 251 took over 1 hour and 40 minutes to generate and print an inverse using the Strassen-type matrix inverse algorithm. The classical NTRU inverse algorithm took a fraction of a second to find an inverse of the same polynomial.

This observation is supported by the assertion made in [36] that the practical implementation of the Strassen inverse algorithm encounters an overhead as a result of the subroutine calls, which subsequently results in slower performance. In practice, full recursion is not used, but rather a switch is made to another algorithm at an appropriate block size. It is recommended that a switch is made to the LU factorization algorithm [36].

C. Gauss Jordan Elimination with Partial Pivoting

The third solution considered was Gauss Jordan elimination with partial pivoting. The matrix inverse of F^{-1} was obtained by implementing Gauss-Jordan elimination of the

Algorithm 4 Finding NTRU Inverse Polynomial Algorithm Using the Strassen-Type Method

procedure INVERSE POLYNOMIAL ALGORITHM(F, k)
  Input: F, N, k
  Output: F_k
  function PARTITION(F)
    $F_{11} = F[1,1]$ to $F[(\frac{N}{2}),(\frac{N}{2})]$
    $F_{12} = F[1,(\frac{N}{2})]$ to $F[(\frac{N}{2}),(\frac{N}{2})]$
    $F_{21} = F[(\frac{N}{2}+1),1]$ to $F[(\frac{N}{2}),(\frac{N}{2})]$
    $F_{11} = F[(\frac{N}{2}+1),(\frac{N}{2}+1)]$ to $F[(\frac{N}{2}),(\frac{N}{2})]$
    return $F_{11}, F_{12}, F_{21}, F_{22}$
  end function
  function STRASSENINVERSE(F)
    ${F_{11}}^{-1} = \frac{1}{|F_{11}|} * AdjF_{11}$
    $R_1 = {F_{11}}^{-1}$
    $R_2 = F_{21} * R_1$
    $R_3 = R_1 * F_{12}$
    $R_4 = F_{21} * R_3$
    $R_5 = R_4 - F_{22}$
    ${R_5}^{-1} = \frac{1}{|R_5|} * AdjR_5$
    $R_6 = {R_5}^{-1}$
    $C_{12} = R_3 * R_6$
    $C_{21} = R_6 * R_2$
    $R_7 = R_3 * C_{21}$
    $C_{11} = R_1 - R_7$
    $C_{22} = -R_6$
    $P_1 = C_{11}, C_{12}$
    $P_2 = C_{21}, C_{22}$
    $C = P_1, P_2$
    return C
  end function
  $F^{-1} = strassenInverse(F)$
  if $GCD(|F|, k) = 1$ **then**
    Multiplicative inverse exists.
  else
    exit. Select another f.
  end if
  $d = |F|^{-1} \bmod k$
  $F^{-1} \bmod k = F^{-1} * |F| * d \bmod k$
  $F_k = F^{-1}[1]$, $F_k(x) \Leftarrow F_k$
  return $F_k(x)$
end procedure

augmented matrix $[f|I]$. The steps followed in finding the polynomial inverse F_k, are as follows:

i. Partially pivot the augmented matrix $[F|I]$ containing $2N$ columns. Since division by zero results in a fatal error, rows are interchanged in order to ensure that there is a non-zero element (pivot element) along the diagonal. If $M_{[i,i]} = 0$, then row i cannot be used to eliminate the elements in the column i below the diagonal. Thus, it is necessary to find a row j, where $M_{[j,i]} \neq 0$ and $j > i$, then interchange row i and row j so that a non-zero pivot element is obtained.
ii. Eliminate the elements below and above the diagonal to result in the reduced row echelon form. This was done using elementary row operations.
iii. Reduce the resulting reduced row echelon form to the identity matrix with only 1's along the diagonal. This was done by dividing each row by the reciprocal of the diagonal element. The resulting matrix on the left should be the identity I while the matrix on the right is the inverse matrix F^{-1}.
iv. Compute the determinant $|F|$. Check if $GCD(|F|, k) = 1$ in order to establish whether the two integers are relatively prime. Once it is ascertained that $|F|$ and k are coprime, then a multiplicative inverse exists. If not, then another polynomial f should be selected.
v. Compute the multiplicative inverse F_k using Equation (5).

An illustration of this matrix inverse solution using Gauss Jordan elimination with partial pivoting integrated in the NTRU structure is given by Algorithm 5 which has a computational complexity of $O(N^2)$. Details of the findings of this evaluation of an NTRU inverse algorithm using Gauss Jordan elimination with partial pivoting were published in [37].

Algorithm 5 Inverse polynomial algorithm based on circulant matrices using Gauss-Jordan elimination with partial pivoting

procedure INVERSE POLYNOMIAL ALGORITHM(F, k)
  Input: F, N, k
  Output: F_k
  I = identity(F), $B = F \cdot I$
  n = rows(F), j = columns(F), n_1 = columns(B)
  for $i = 2$ downto n **do** ▷ Partial pivoting.
    if $B[i-1,1] < B[i,1]$ **then**
      for $j = 1$ downto n_1 **do**
        $d = B[i,j]$, $B[i,j] = B[i-1,j]$
        $B[i-1,j] = d$
      end for
    end if
  end for
  for $i = 1$ downto n **do**
    for $j = 1$ downto n **do**
      if $j \neq i$ **then**
        if $B[i,i] = 0$ **then**
          $B[i,i] \leftarrow B[j,i]$
        end if
        $d = B[j,i]/B[i,i]$
        $(B[i] * -d) + B[j]$
      end if
    end for
  end for
  for $i = 1$ downto n **do**
    $d = B[i,i]$, $(B[i] * 1/d)$ ▷ Reduction to I.
  end for
  $F^{-1} = B[n+1]$ to $B[n_1]$
  if $GCD(|F|, k) = 1$ **then**
    Multiplicative inverse exists.
  else
    exit. Select another f.
  end if
  $d = |F|^{-1} \bmod k$
  $F^{-1} \bmod k = F^{-1} * |F| * d \bmod k$
  $F_k = F^{-1}[1]$, $F_k(x) \Leftarrow F_k$
  return $F_k(x)$
end procedure

a) Analysis:: Implementation of Gauss Jordan elimination with partial pivoting in the NTRU key generation structure was found to be more simplified and can be extensible to polynomials with coefficients in other fields besides binary and ternary fields, allowing for the use of non-prime modulus. However, the algorithm was slower than classical NTRU inverse algorithm, but faster than the naïve matrix inversion and Strassen-type matrix inverse algorithms implemented in NTRU. It was also characterized by division by zero errors, despite the use of partial pivoting, which limits the parameter selection to polynomials without a series of consecutive zero coefficients and a polynomial whose first coefficient is not a zero.

D. LU Decomposition

The fourth solution considered was the LU (lower-and-upper) decomposition method. This approach was implemented as follows:

i. F was decomposed into the product LU. Then the decomposed product was substituted into the original equation resulting in the statement of the problem as $L \cdot U \cdot X = I$.
ii. Letting $U \cdot X = Y$ thus $L \cdot Y = I$. Then, the solution for Y in $L \cdot Y = I$ was obtained by solving the N system of linear equations by forward substitution and the process repeated for all the N columns of the identity matrix I, which were then combined to form matrix Y.
iii. The solution for X in $U \cdot X = Y$ was obtained by solving the N system of linear equations by substituting back the values of y and the process repeated for all the N columns of the matrix Y, which were then combined to form matrix X which represents the inverse matrix F^{-1}.
iv. The multiplicative inverse F_k was then computed using Equation (5).

An illustration of the integration of this LU decomposition matrix inverse solution with the NTRU structure is given by Algorithm 6.

a) Analysis:: This algorithm was found to be faster and less complex than the naïve matrix inverse [10] algorithm, but is slower and more complex than the classical NTRU inverse algorithm and the inverse algorithm using Gauss Jordan elimination but with the benefit of allowing for the use of a non-prime modulus. However, the algorithm was characterized by division by zero errors, which limits the parameter selection to polynomials without a series of consecutive zero coefficients and a polynomial whose first coefficient is not a zero.

E. Gram Schmidt orthogonalization

The fifth solution considered was the use of Gram Schmidt orthogonalization. This solution using Gram Schmidt orthogonalization was considered in this study on the premise that, given an orthogonal matrix, the columns of the matrix can be transformed so as to possess orthonormal properties thereby making it an orthonormal matrix. Consequently, the transpose of an orthogonal matrix is its inverse. Thereby, it would eliminate the need for computing an inverse provided that any randomly selected polynomial f is transformed into a circulant matrix and subsequently transformed into an orthonormal matrix whose inverse is its transpose. Gram Schmidt orthogonalization is formally defined by [38] as:

Definition 2. The process of forming an orthogonal sequence q_n from a linearly independent sequence x_n of members from a finite or infinite inner-product space by defining q_n inductively as

$$q_1 = x_1,\ q_n = x_n - \sum_{k=1}^{n-1} \frac{< q_k, x_n >}{|q_k|^2},\ n \geq 2 \qquad (6)$$

An orthonormal sequence can be obtained by replacing each q_n by $\frac{q_n}{|q_n|}$.

Algorithm 6 Finding NTRU inverse polynomial algorithm using LU decomposition method of matrix inversion

procedure INVERSE POLYNOMIAL ALGORITHM(F, k)
  Input: F, N, k
  Output: F_k
  function LUDECOMPOSITION(B)
    L = identity(F),
    for $i = 1$ downto N **do** ▷ Partial pivoting.
      for $j = 1$ downto N **do**
        if $j > i$ **then**
          $d = B[j,i]/B[i,j]$
          $L[j,i] = d,\ (B[i] * -d) + B[j]$
        end if
      end for
    end for
    return B, L
  end function
  $U, L = LUDecomposition(F)$
  function GAUSSELIMINATIONSOLVE(B)
    for $i = 1$ downto N **do**
      for $j = 1$ downto N **do**
        if $j \neq i$ **then**
          if $B[i,i] = 0$ **then**
            $B[i,i] \leftarrow B[j,i]$
          end if
          $d = B[j,i]/B[i,i]$
          $(B[i] * -d) + B[j]$
        end if
      end for
    end for
    for $i = 1$ downto N **do**
      $d = B[i,i],\ (B[i] * 1/d)$ ▷ Reduction to I.
    end for
    return B
  end function
  I = identity(F)
  $Q = [\,]$
  for $i = 1$ downto N **do**
    $B \leftarrow I[i]$
    $C = L \cdot B$
    $B_I = gaussEliminationSolve(C)$ ▷ Solving $LY = B$ for Y.
    $Y \leftarrow B_I[N+1]$
    $D = U \cdot Y$
    $D_I = gaussEliminationSolve(D)$ ▷ Solving $UX = Y$ for X.
    $Q \cdot X$
  end for
  $F^{-1} = Q^T$
  $d = |F|^{-1} \bmod k,\ F^{-1} \bmod k = F^{-1} * |F| * d \bmod k$
  $F_k = F^{-1}[1],\ F_k(x) \Leftarrow F_k$
  return $F_k(x)$
end procedure

This approach was implemented as follows:

i. The orthogonal transformation of the vectors of matrix F was obtained by using Gram Schmidt orthogonalization. This involves splitting the matrix F into vectors defined row-wise from F. The vectors are then transformed by applying Equation (6). The orthogonal basis is described by a row-wise combination of the transformed vectors.
ii. The orthonormal transformation $\acute{F}$ of F was obtained by dividing each of the vectors in F by the norm and then combining the results column-wise.
iii. The inverse of $\acute{F}$ (denoted by $\acute{F}^{-1}$) was obtained by finding the transpose of $\acute{F}$. The resulting $\acute{F}^{-1}$ was taken to represent the inverse F^{-1}.
iv. Finally, the multiplicative inverse F_k was obtained by applying Equation (5).

An integration of this Gram Schmidt orthogonal transformation solution in the NTRU structure is given by Algorithm 7 which has a computational complexity of $O(2MN^2)$ for a matrix of dimension $M \times N$ [39]; given that NTRU matrix F has a dimension of $N \times N$, the corresponding computational complexity is $O(2N^4)$.

Algorithm 7 Inverse polynomial algorithm based on Gram Schmidt orthogonal transformation

procedure INVERSE POLYNOMIAL ALGORITHM(F)
  Input: F, q
  Output: F_k
  $n =$ number of columns(F)
  $m =$ number of rows(F)
  Initialize V and B as null space matrices of dimension $n \times n$
  for $j = 1$ downto m **do**
    $V[j] = F[j]$
    for $i = 1$ downto $j - 1$ **do**
      $r = (F[j], V[i])\,/\,(V[i], V[i])$
      $V[j] = V[j] - r * V[i]$
    end for
  end for
  Norm $= 0$
  for $k = 1$ downto m **do**
    Norm $= 0$
    for $i = 1$ downto n **do**
      Norm $= (V[k, 1])^2 +$ Norm
    end for
    Norm $= \sqrt{\text{Norm}}$
    $B[k] = V[k]/\text{Norm}$
  end for
  $B^{-1} = B^T$
  $d = |B|$
  $F_k = B^{-1} * d \mod q$ ▷ $F_k * B \mod q = I$.
  return F_k
end procedure

a) Analysis:: The results of this evaluation revealed that, despite the ease in finding the inverse provided by transforming F (derived from any random polynomial f) into its orthonormal form by simply finding its transpose, the resulting inverse matrix F^{-1} and subsequently F_k had non-integral coefficients.

For example: a polynomial f with coefficients $[1, 1, 1, 1, 1, 0, 0]$ resulting in a matrix F whose determinant is 5, had a resulting orthonormal form $\acute{F}$ whose first row of elements was $[0.4472, 0.4472, 0.4472, 0.4472, 0.4472, 0, 0]$ and whose resulting inverse $\acute{F}_k^{-1}$ had the first row of the coefficients $[0.4472, -0.5963, 0.0833, 0.4692, 0.0726, 0.3451, -0.3050]$ with $k = 37$. Therefore, this solution was not considered to be suitable for this study, due to the resulting inverse with non-integral coefficients.

F. Least Squares Method

The sixth and final solution considered was the use of the Pseudo-Inverse in finding the least squares solution to an inverse matrix problem. This alternative inverse solution is founded on the premise that a unique "matrix-like" inverse referred to as a pseudo-inverse can be applied in finding the solution to a problem of the form $Ax = b$. For instance, when b is within the range of A, one or more solutions to the system exist. Conversely, if b lies outside the range of A, no solutions to the system exist but it is possible to find a value $\hat{x}$ closest to a solution. This value will be an approximate solution to the system $Ax = b$ whereby there is minimum residual error $r = Ax - b$ [40], [41].

Definition 3. [42] The least squares solution to a system is a vector such that

$$\|\hat{r}\| = \|A\hat{x} - b\| \leq \|Ax - b\| \tag{7}$$

where $A \in \mathbb{R}^{m \times n}$ is a matrix with linearly independent columns. The vector $b \in \mathbb{R}^m$ is a vector whose entries are subject to random errors of equal variance.

The unique least squares solution is the one with a minimum norm in the residual vector. The norm of a vector x is given by $\|x\|$ where $\|x\| = \sqrt{x^2}$ [40].
The best approximate solution to the problem $Ax = b$ is given by

$$\hat{x} = A^+ b = (A^T A)^{-1} A^T b \tag{8}$$

An implementation of the NTRU inverse polynomial algorithm using the solution of least squares is given by Algorithm 8.

Algorithm 8 Efficient NTRU inverse polynomial algorithm using the solution of least squares

procedure INVERSE POLYNOMIAL ALGORITHM(F, k)
  Input: F, N, k
  Output: F_k
  I = identity(F)
  $B = I[1]$
  $A = F^T * F$
  $X[1] = A^{-1} * F^T * B$
  $\hat{x} = X[1]$
  $F_t = circulantMatrix(X)$
  $F^{-1} = F_t^T$
  $d = |F^{-1}| \, mod \, k$
  $F^{-1} \, mod \, k = F^{-1} * |F| * d \, mod \, k$
  $F_k = F^{-1}[1],\ F_k(x) \Leftarrow F_k$
  return $F_k(x)$
end procedure

We describe an implementation of the least squares solution to the NTRU inverse problem using a small-scale example.

Example 1. *If we take the following NTRU parameters: $N = 7, p = 2$ and $f = x + x^3 + x^5$ whereby $d_f = 3$, the modular polynomial inverse F_p is computed as follows:*

i. Given that an N-dimensional complex vector space is created by generating a circulant matrix which is defined by its first row $(f_0, f_1, f_2, \cdots, f_{N-1})$ [31], [33], [32], we formulate matrix $F = Circ(f)$ (derived from the polynomial f) and state the problem $F\,x_1 = I_1$, where x_1 is the first column of the inverse matrix and I_1 is the first column of the identity matrix.

$$F\,x_1 = I_1$$

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \cdot x_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{9}$$

ii. Solving for x_1 by multiplying both sides of the equation with F^T and making x_1 the subject of the equation, results in:

$$x_1 = (F^T \cdot F)^{-1} \cdot F^T \cdot I_1$$

$$x_1 = \hat{x_1} = \begin{bmatrix} -0.67 \\ 0.33 \\ 0.33 \\ 0.33 \\ 0.33 \\ 0.33 \\ -0.67 \end{bmatrix} \quad (10)$$

which gives the least squares solution $\hat{x_1}$ for the first column of the inverse matrix F^{-1}.

iii. Based on the premise of Conjecture 1, once the first column of $\hat{x}_i$ is obtained where $i = 1$, instead of repeating the process of finding $\hat{x_i}$ for columns $2 \cdots N$, the least squares solution $\hat{x_1}$ is transformed into a row-wise sequence, that is, $\hat{x}_1^T$.

Conjecture 1. *A square circulant matrix exhibits circulant properties both row-wise and column-wise. Therefore, if you solve for one column of a square circulant matrix, the remaining subsequent columns can be obtained by cyclic shifts of the columns by an offset equal to the row index.*

iv. A square circulant matrix X of dimension $N \times N$ is generated using $\hat{x}_1^T$ by transforming it into a column-wise sequence by obtaining its transpose, which is taken as the inverse matrix F^{-1}.

v. The inverse matrix modulo p is obtained by computing

$$F_p = F^{-1} \times |F| \times (|F|^{-1} mod\, p)$$

$$= \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Extracting the first column of the matrix $[0, 0, 1, 1, 1, 1, 1]$ and expressing it as a polynomial in the ring $(X^N - 1)$, that is $(X^7 - 1)$, results in the modular polynomial inverse

$$F_p = x^6 + x^5 + x^4 + x^3 + x^2. \quad (11)$$

It can be confirmed that the product $f * F_p \, mod\, p \equiv 1$ in the polynomial ring $(x^7 - 1)$ thereby proving that the inverse is correct in the ring $(X^7 - 1)$.

IV. Analysis of Alternative NTRU Inverse Polynomial Algorithms

An overall evaluation of the alternative modular inverse solutions provided valuable insight on the strengths and weaknesses of the alternative NTRU inverse solutions considered for this study. The alternative inverse solutions are evaluated in terms of the following attributes:

i. It allows for the selection of any random polynomial f; thereby eliminating the need for a parameter selection criteria for polynomial f.
ii. It allows for the use of a non-prime modulus, that is an inverse can be found even when p and q are not relatively prime.
iii. The result is free from division by zero errors.
iv. The resulting inverse is correct ($F * F_k mod k = 1$) and the inverse has integral coefficients.
v. Computational complexity and thereby time taken to find an inverse.

An evaluation of the six alternative inverse solutions in terms of these attributes yielded the results in Table I.

TABLE I
COMPARISON OF ALTERNATIVE INVERSE SOLUTIONS

Inverse Solution	**Random polynomial f selection**	**Allows use of non-prime modulus**	**Free of division by zero errors**	**Correct inverse result with integral coeffi-cients**	**Computational complexity**
Naïve matrix inversion [10]	✓	✓	✓	✓	O(N^3)
Strassen-type matrix inverse	✗	✓	✓	✓	O($\frac{6}{5}N^{2.807}$)
Gauss Jordan elimination with partial pivoting [37]	✗	✓	✗	✓	O(N^2)
LU decomposition [43]	✗	✓	✗	✓	O(N^2)
Gram Schmidt orthogonalization	✓	✓	✓	✗	$O(2N^4)$
Least squares method	✓	✓	✓	✓	O(N^3)

The evaluation revealed that the naïve matrix inversion and least squares method have the most favourable results. A much closer look at the two solutions reveals that the least squares method has a faster speed of inversion compared to the naïve matrix inversion.

Further evaluation of the alternative inverse algorithms in terms of: the probability of finding an inverse, speed of inversion and computational time complexity is discussed in the subsequent sections.

A. Probability of finding an inverse

A comparison (using NTRU binary polynomials) of the proposed alternative inverse solutions against that of the classical NTRU Inverse algorithm yielded the results shown in Table II.

In accordance with the results depicted in Table II, compared to the other inverse algorithms, the NTRU inverse algorithm using the least squares method of finding a generalized (or "almost inverse") provides the greatest probability of finding an inverse. The classical inverse algorithm has no result when the algorithm encounters a randomly selected private polynomial f which is not invertible, thereby going into an infinite loop of trying to find an inverse. This thereby justifies the recommendation of a parameter selection criteria.

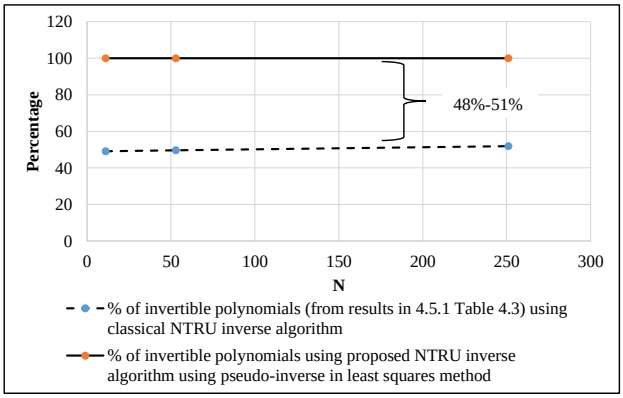
The experimental evaluation conducted in this study revealed a probability of finding an inverse that ranges from

TABLE II
COMPARISON OF THE PROBABILITY OF FINDING AN INVERSE

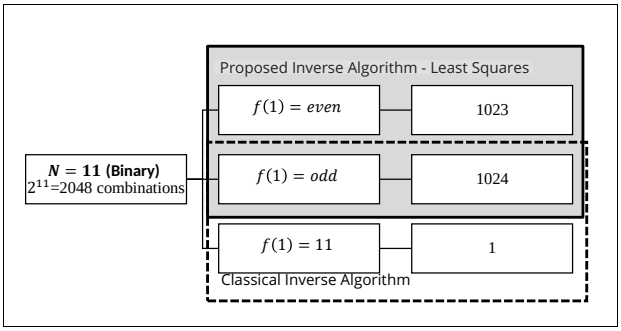
Algorithm	Probability of Finding an Inverse
Classical NTRU inverse algorithm [44], [14], [6]	0.49 to 0.51.
Zhao and Su inverse algorithm [10]	Polynomials with an odd number of 1 coefficients; matrices with a non-zero determinant
Inverse algorithm using Gauss Jordan Elimination [37]	Polynomials yielding matrices with a non-zero determinant; matrices without a division by zero error while conducting row operations
Inverse algorithm-solution using LU Decomposition [43]	Matrices with a non-zero determinant, matrices without a division by zero error while conducting row operations
Proposed inverse algorithm-solution using least squares method	100% of the polynomials; all binary polynomials except for polynomials with all 1 coefficients

0.49 to 0.51. The algorithm by [10] finds inverses for matrices with a non-zero determinant, in other words non-singular matrices. The algorithms using Gauss Jordan elimination and LU decomposition methods encounter division by zero errors.

A comparison of the probability of finding an inverse using the proposed NTRU inverse algorithm using the least squares method compared to the classical NTRU inverse algorithm is shown in Figure 1a, with an example using test parameters for $N = 11$ in Figure 1b.



(a) Comparative percentages.



(b) Size of the key space.

Fig. 1. Comparison of invertible polynomials for the least squares inverse algorithm and the classical NTRU inverse algorithm

As depicted by Figure 1b, the application of the Pseudo-Inverse in finding the least squares solution in NTRU resulted in a 50% increase in the number of invertible polynomials thereby doubling the size of the key space. The classical inverse algorithm covers only 49.95% of the polynomial space while the improved inverse algorithm using the Pseudo-Inverse in finding the least squares solution covers 99.95% of the polynomial space. An inverse and thus key generation can be successfully done for all binary NTRU polynomials using the Pseudo-Inverse in finding the least squares solution with the exception of the polynomial with all 1 coefficients (that is, with $f(1) = N$). Therefore, the least squares solution of inversion provides for wholly random selection of polynomial f for a specified parameter size.

B. Speed of Inversion

The comparative speed of inversion is shown in Figure 2. The tests were conducted on the Magma Computational Algebra System [45] running on a Windows 10 Enterprise operating system (with an Intel(R) Core(TM) i7-4790 processor running at 3.60GHz, 4GB RAM and 1TB Hard-disk capacity). The parameters tested were obtained from test vectors published in [46], [47], [48].

As depicted in Figure 2, the proposed algorithm using the least squares method shows better performance in terms of the speed of inversion compared to the algorithm by [10] as well as the inverse algorithm using LU decomposition. However, the classical inverse algorithm [44], [14], [6] and the inverse algorithm using Gauss Jordan elimination show faster speeds of inversion, with the classical inverse algorithm being the fastest.

Overall, the classical NTRU inverse algorithm takes the shortest time to find an inverse which is attributed to the speed of convolution multiplication operation. This is followed by the inverse algorithm using Gauss Jordan elimination in second place. However, the application of the inverse algorithm using Gauss Jordan elimination is characterized by instances of division by zero errors, arising as a result of the arrangement of 0 and 1 coefficients.

The proposed inverse algorithm using the least squares method is in third place, followed by the Zhao and Su Inverse algorithm, while the inverse algorithm using LU decomposition has the longest computation time. The power consumption increases as the size of the matrices gets larger, which has a consequence on the computation speed. This is particularly observed in the NTRU inverse algorithms considered in this research study, with the inverse algorithm using LU decomposition having the greatest resource consumption at $N = 449$. This implies that the LU decomposition is less efficient for inversion of matrices of larger dimension.

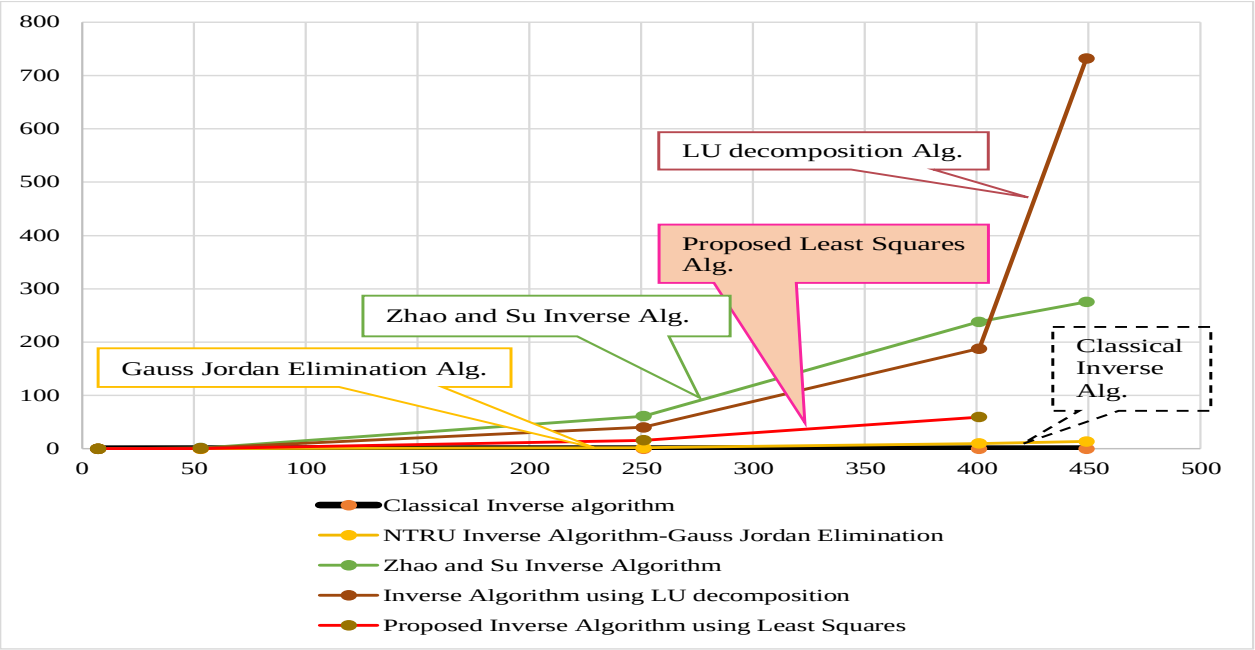


Fig. 2. Comparative speed of inversion

Despite the fast speed of inversion using the classical NTRU inverse algorithm, a randomly selected private polynomial f is not always invertible thereby restricting the parameter selection range. The speed of inversion and thereby utility of computational resources by the proposed inverse algorithm using the least squares method is at a moderate level among the five algorithms, which coupled with the probability of finding an inverse provides flexibility in parameter selection thereby making it a favourable solution among the alternatives.

C. Computational Time Complexity

The third metric of comparison we considered was the computational time complexity. The five inverse algorithms were compared in terms of their computational time complexity yielding the results illustrated in the Table III.

TABLE III
COMPARATIVE TIME COMPLEXITY

Algorithm	Time complexity
Classical NTRU inverse algorithm	$O(N^2 log_2\, q + 1)$
Zhao and Su inverse algorithm	$O(N^2)$
Inverse algorithm using Gauss Jordan Elimination	$O(10N^2 + 4N)$
Inverse algorithm-solution using LU Decomposition	$O(13N^2 + 6N)$
Proposed inverse algorithm-solution using least squares method	$O(N^3 + N^2 + N + log^2 N)$

The computational complexity of the proposed inverse algorithm using the least squares method is $O(N^3 + N^2 + N + log^2 N \simeq N^3 + N^2 + N)$ which simplifies to $O(N^3)$. On the other hand, the complexity of the inverse algorithm using Gauss Jordan Elimination is $O(10N^2 + log^2 N + 4N + 22 \simeq 10N^2 + 4N)$ which simplifies to $O(N^2)$ while the algorithm using LU decomposition has a complexity of $O(13N^2 + 6N)$. A closer evaluation of the complexities reveals that the classical NTRU inverse algorithm requires fewer operations of $O(N^2\, log_2\, q + 1 \simeq N^2\, log_2\, q)$ which justifies the faster numerical speed of inversion illustrated in Figure 2. This is attributed to the convolution multiplication operation and computation of the products of small coefficients.

V. Conclusion

The NTRU public key cryptosystem's key generation entails the computation of two modular polynomial inverses thereby pointing to the crucial value of modular inversion in NTRU. However, previous studies point out the difficulty in determining whether a polynomial is invertible [7]. Therefore, this study seeks to address the problem of invertibility in the NTRU key generation process by studying alternative inverse algorithms, presenting their integration in the NTRU structure and identifying the best inverse algorithm.

This study presents a discussion of the classical NTRU inverse algorithm, an evaluation of alternative modular inverse algorithms examined in the course of this study along with a proposal of the best inverse algorithm. We compare naïve matrix inversion, Strassen-type matrix inverse, Gauss Jordan elimination with partial pivoting, LU decomposition, Gram Schmidt orthogonalization and least squares method. Given that a generalized inverse is not unique, we apply the principle of the pseudo-inverse in finding the least squares solution (with a minimum norm in the residual error for an over-determined system of linear equations) of a matrix problem because it results in a unique solution

A comparative evaluation of the algorithms in terms of the probability of finding an inverse, the speed of inversion and the computational time complexity of the algorithms revealed that the least squares algorithm provides 100% probability of finding a modular inverse for NTRU binary polynomials. This guarantees that an inverse can be found provided the polynomial does not have all N number of 1 coefficients.

Therefore, the integration of this algorithm in the NTRU structure allows for random selection of any private key polynomial f in the ring $R = \frac{\mathbb{Z}[x]}{(X^N-1)}$ (i.e., any polynomial combination with maximum degree $(N-1)$) which thereby enlarges the key space and consequently improves security.

REFERENCES

[1] G. C. Kessler, "An overview of cryptography," 2019.

[2] X. A. Wang, X. Yang, C. Li, Y. Liu, and Y. Ding, "Improved functional proxy re-encryption schemes for secure cloud data sharing," *Computer Science and Information Systems*, vol. 15, no. 3, pp. 585–614, 2018.

[3] P. W. Shor, "Algorithms for quantum computation," in *35th Annual Symposium on Foundations of Computer Science (FOCS)*.

[4] J. A. Buchmann, D. Butin, F. Göpfert, and A. Petzoldt, "Post-quantum cryptography: state of the art," in *The New Codebreakers*. Springer, 2016, pp. 88–108.

[5] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[6] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic number theory*, J. P. Buhler, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 267–288.

[7] X.-R. Luo and C.-H. J. Lin, "Discussion on matrix NTRU," *International Journal of Computer Science and Network Security*, vol. 11, no. 1, pp. 32–35, 2011.

[8] R. Nayak, C. Sastry, and J. Pradhan, "Algorithmic comparison between polynomial base and matrix base NTRU cryptosystem," *International Journal of Computer and Network Security (IJCNS) Vol*, vol. 2, 2010.

[9] W. D. Banks and I. E. Shparlinski, "A variant of NTRU with non-invertible polynomials," in *International Conference on Cryptology in India*. Springer, 2002, pp. 62–70.

[10] N. Zhao and S. Su, "An improvement and a new design of algorithms for seeking the inverse of an NTRU polynomial," in *7th International Conference on Computational Intelligence and Security (CIS)*, 2011, Conference Proceedings, pp. 891–895.

[11] P. I. Etingof, O. Golberg, S. Hensel, T. Liu, A. Schwendner, D. Vaintrob, and E. Yudovina, *Introduction to representation theory*. American Mathematical Society Providence, RI, 2011, vol. 59.

[12] A. P. Hillman and G. L. Alexanderson, *Abstract algebra: A first undergraduate course*. PWS Publishing Company, 1994.

[13] N. Howgrave-Graham, J. H. Silverman, and W. Whyte, *Choosing parameter sets for NTRUEncrypt with NAEP and SVES-3*. Springer, 2005, pp. 118–135.

[14] J. Hoffstein, J. Pipher, J. H. Silverman, and J. H. Silverman, *An introduction to mathematical cryptography*. Springer, 2008, vol. 1.

[15] R. T. Monteiro, "Post-quantum cryptography: lattice-based cryptography and analysis of NTRU public-key cryptosystem," Faculdade de Ciencias, Departamento de Matematica, 2016.

[16] S. R. Fluhrer, "Quantum cryptanalysis of NTRU." *IACR Cryptology ePrint Archive*, vol. 2015, p. 676, 2015.

[17] P. S. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte, "Choosing NTRUEncrypt parameters in light of combined lattice reduction and mitm approaches," in *Applied Cryptography and Network Security*, ser. Lecture Notes in Computer Science, M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, Eds. Springer Berlin Heidelberg, 2009, vol. 5536, pp. 437–455. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01957-9-27

[18] N. Howgrave-Graham, P. Nguyen, D. Pointcheval, J. Proos, J. Silverman, A. Singer, and W. Whyte, "The impact of decryption failures on the security of NTRU encryption," in *Advances in Cryptology - CRYPTO 2003*, ser. Lecture Notes in Computer Science, D. Boneh, Ed. Springer Berlin Heidelberg, 2003, vol. 2729, pp. 226–246. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45146-4-14

[19] J. A. Buchmann, *Congruences and Residue Class Rings*. New York, NY: Springer New York, 2004, pp. 29–70. [Online]. Available: https://doi.org/10.1007/978-1-4419-9003-7_2

[20] J. H. Silverman, "Almost inverses and fast NTRU key creation," *NTRU Cryptosystems,(Technical Note# 014): http://www. ntru. com/cryptolab/pdf/NTRU Tech014. pdf*, 1999.

[21] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck, "Fast key exchange with elliptic curve systems," in *Annual International Cryptology Conference*. Springer, 1995, Conference Proceedings, pp. 43–56.

[22] IEEE, "IEEE standard specification for public key cryptographic techniques based on hard problems over lattices," *IEEE Std 1363.1-2008*, pp. C1–69, March 2009.

[23] A. K. Jones, "Efficient embedded security standard (EESS) 1," March 2015. [Online]. Available: https://github.com/NTRUOpenSourceProject/ntru-crypto

[24] IEEE, "Efficient embedded security standards (EESS)," p. 78, June 2003.

[25] P. G. Tata, H. Narumanchi, and N. Emmadi, "Analytical study of implementation issues of NTRU," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2014, Conference Proceedings, pp. 700–707.

[26] N. Challa and J. Pradhan, "Performance analysis of public key cryptographic systems rsa and ntru," *International Journal of Computer Science and Network Security*, vol. 7, no. 8, pp. 87–96, 2007.

[27] C. O. Rourke and B. Sunar, "Achieving NTRU with montgomery multiplication," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 440–448, 2003.

[28] "IEEE draft standard specification for public- key cryptographic techniques based on hard problems over lattices," *IEEE Unapproved Draft Std P1363.1/D12, Oct 2008*, p. 1, 2008.

[29] W. Stallings, *Cryptography and Network Security: Principles and Practice, International Edition: Principles and Practice*. Pearson Higher Ed, 2014, vol. 5.

[30] W. Whyte, *Ntru*. Boston, MA: Springer US, 2005, pp. 427–430. [Online]. Available: http://dx.doi.org/10.1007/0-387-23483-7-279

[31] P. J. Davis, *Circulant matrices*. American Mathematical Soc., 2012.

[32] I. Kra and S. R. Simanca, "On circulant matrices," *Notices of the AMS*, vol. 59, no. 3, pp. 368–377, 2012.

[33] D. S. G. Pollock, "Circulant matrices and time-series analysis," *International Journal of Mathematical Education in Science and Technology*, vol. 33, no. 2, pp. 213–230, 2002.

[34] I. Kra and S. R. Simanca, "On circulant matrices," *Notices of the AMS*, vol. 59, no. 3, pp. 368–377, 2012.

[35] S. M. Balle, P. C. Hansen, and N. Higham, "A strassen-type matrix inversion algorithm," *Advances in Parallel Algorithms*, pp. 22–30, 1994.

[36] M. D. Petkovic and P. S. Stanimirovic, "Generalized matrix inversion is not harder than matrix multiplication," *Journal of Computational and Applied Mathematics*, vol. 230, no. 1, pp. 270 – 282, 2009. [Online]. Available: //www.sciencedirect.com/science/article/pii/S0377042708006237

[37] G. J. Nyokabi, M. Salleh, and I. Mohamad, "Ntru inverse polynomial algorithm based on circulant matrices using gauss-jordan elimination," in *6th ICT International Student Project Conference (ICT-ISPC)*. IEEE, May 2017, pp. 1–5.

[38] G. James and R. C. James, "Mathematics dictionary," *Mathematics dictionary, by James, Glenn; James, Robert C. Princeton, NJ, Van Nostrand [1959]*, 1959.

[39] V. Lyubashevsky and T. Prest, "Quadratic time, linear space algorithms for gram-schmidt orthogonalization and gaussian sampling in structured lattices," in *Advances in Cryptology – EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 789–815.

[40] R. MacAusland, "The moore-penrose inverse and least squares," *Math 420: Advanced Topics in Linear Algebra*, 2014.

[41] S. Boyd. (2008) Least squares. [Online]. Available: https://see.stanford.edu/materials/lsoeldsee263/05-ls.pdf

[42] S. J. Leon, Å. Björck, and W. Gander, "Gram-schmidt orthogonalization: 100 years and more," *Numerical Linear Algebra with Applications*, vol. 20, no. 3, pp. 492–532, 2013.

[43] J. N. Gaithuru, M. Salleh, and I. Mohamad, "Ntru inverse polynomial algorithm based on the lu decomposition method of matrix inversion," in *IEEE Conference on Application, Information and Network Security (AINS)*, Nov 2017, pp. 1–6.

[44] J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, W. Whyte, and Z. Zhang, "Choosing parameters for NTRUEncrypt," 2015. [Online]. Available: http://eprint.iacr.org/2015/708.pdf

[45] M. Group *et al.*, "Magma computational algebra system," Version 2.22-7, 2016.

[46] Y. Angel, "NTRUEncrypt," 2014. [Online]. Available: https://github.com/Yawning/ntru/tree/master/testvectors

[47] A. Yawning, "Package NTRU," 2016. [Online]. Available: https://godoc.org/github.com/Yawning/ntru

[48] J. N. Gaithuru, M. Salleh, and M. Bakhtiari, "Identification of influential parameters for NTRU decryption failure and recommendation of extended parameter selection criteria for elimination of decryption failure." *IAENG International Journal of Computer Science*, vol. 44, no. 3, pp. 358–367, 2017.

Juliet N. Gaithuru was born in Thika, Kenya on the 15th of November 1985. This author attained a BSc. in Computer Information Systems from Kenya Methodist University in 2011. The author also holds a Master of Computer Science in Information Security from Universiti Teknologi Malaysia, 2013, where she carried out research on the S-Box in the Advanced Encryption Standard (AES) Algorithm. The author is currently pursuing a PhD degree in computer science specializing in the field of information security at Universiti Teknologi Malaysia. Her research interests are in the field of symmetric and asymmetric cryptography with particular interest in post-quantum cryptography.

Mazleena Salleh is an Associate Professor at the Faculty of Computing, Universiti Teknologi Malaysia. The author holds a PhD in Computer Science from Universiti Teknologi Malaysia. The author holds an MSc in Electrical Engineering from Virginia Polytechnic and State University in USA. The author attained a BSc in Electrical Engineering from University of Southern California, USA as well as a Diploma in Electrical Engineering from Universiti Teknologi Malaysia.

Nor Muhainiah Mohd Ali is an Associate Professor in the Mathematical Sciences department at the Faculty of Science, Universiti Teknologi Malaysia. Her research interests are in pure mathematics, discrete mathematics, graph theory, algebra and analysis.