Two RSA-based Cryptosystems

A. Telveenus[1*]

[1*]International Study Centre, Kingston University, Kingston Hill Campus, London, KT2 7LB, UK.

Corresponding author(s). E-mail(s): t.fernandezantony@kingston.ac.uk;

Abstract

The cryptosystem RSA is a very popular cryptosystem in the study of Cryptography. In this article, we explore how the idea of a primitive m^{th} root of unity in a ring can be integrated into the Discrete Fourier Transform, leading to the development of new cryptosystems known as RSA-DFT and RSA-HGR.

Keywords: Primitive m^{th} roots of unity, halidon rings, halidon group rings, RSA cryptosystem, Discrete Fourier Transform.

MSC Classification: 16S34 , 20C05 , 11T71

1 Introduction

The development of asymmetric cryptography, also known as public-key cryptography, is the greatest and perhaps the only true revolution in the history of cryptography.A significant portion of the theory behind public-key cryptosystems relies on number theory [6]. RSA is a widely used public-key cryptosystem for secure data transmission. It was first described publicly in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. An equivalent system had been secretly developed in 1973 by Clifford Cocks at the British signals intelligence agency, GCHQ.
In RSA, the encryption key is public, while the decryption key is private. To create a public key, a user chooses two large prime numbers and an auxiliary value, and then publishes this information. The prime numbers are kept secret. Anyone can use the public key to encrypt messages, but only someone who knows the private key can decrypt them. The security of RSA relies on the factoring problem, which is the practical difficulty of factoring the product of two large prime numbers. Breaking RSA encryption is known as the RSA problem. RSA is a relatively slow algorithm, so it is

not commonly used to directly encrypt user data. Instead, it is used to transmit shared keys for symmetric-key cryptography. The author establishes a connection between RSA and Halidon rings using Discrete Fourier Transform and Halidon group rings in this article.

In 1940, the famous celebrated mathematician Graham Higman published a theorem [2],[5] in group algebra which is valid only for a field or an integral domain with some specific conditions. In 1999, the author noticed that this theorem can be extended to a rich class of rings called halidon rings[7].

A primitive m^{th} root of unity in a ring with unit element is completely different from that of in a field, because of the presence of nonzero zero divisors. So we need a separate definition for a primitive m^{th} root of unity. An element ω in a ring R is called a *primitive* m^{th} root if m is the least positive integer such that $\omega^m = 1$ and

$$\begin{aligned}\sum_{r=0}^{m-1} \omega^{r(i-j)} &= m, \quad i = j(\ mod\ m)\\ &= 0, \quad i \neq j(\ mod\ m).\end{aligned}$$

More explicitly,

$$\begin{aligned}1 + \omega^r + (\omega^r)^2 + (\omega^r)^3 + (\omega^r)^4 + ... + (\omega^r)^{m-1} &= m, \quad r = 0\\ &= 0, \quad 0 < r \leq m-1.\end{aligned}$$

A ring R with unity is called a *halidon* ring with index m if there is a primitive m^{th} root of unity and m is invertible in R. The ring of integers is a halidon ring with index $m = 1$ and $\omega = 1$. The halidon ring with index 1 is usually called a *trivial* halidon ring. The field of real numbers is a halidon ring with index $m = 2$ and $\omega = -1$. The field $\mathbb{Q}(i) = \{a + ib | a, b \in \mathbb{Q}\}$ is a halidon ring with $\omega = i$ and $m = 4$. $\mathbb{Z}_p$ is a halidon ring with index $p - 1$ for every prime p. Interestingly, $\mathbb{Z}_{p^k}$ is also a halidon ring with same index for any integer $k > 0$ and it is not a field if $k > 1$. Note that if ω is a primitive m^{th} root of unity, then ω^{-1} is also a primitive m^{th} root of unity.

2 Preliminary results

In this section, we state some new results and the results essential for constructing the RSA-DFT and RSA-HGR Cryptosystems only. The readers who are interested in the properties of halidon ring, can refer to [7], [8] and [9].

Let $U(R)$ denote the unit group of R.The following theorem will give the necessary and sufficient conditions for a ring to be a halidon ring. The author has used this theorem to develop the computer programme-1 given below.

Theorem 1. *(A. Telveenus [8]) A finite commutative ring R with unity is a halidon ring with index m if and only if there is a primitive m^{th} root of unity ω such that m, $\omega^d - 1 \in U(R)$; the unit group of R for all divisors d of m and $d < m$.*

Proposition 2. *The homomorphic image of a commutative halidon ring with index m and primitive m^{th} of unity ω is also a halidon ring with index m.*

Proposition 3. *Let R be a commutative halidon ring with index m and let $k > 1$ be a divisor of m. Then R is also a halidon ring with index k.*

In the rest of the section, let $R = \mathbb{Z}_n$ be a halidon ring with index m and primitive m^{th} root of unity ω.

Lemma 4. *Let p be an odd prime number and k a positive integer. Then*

1. *$U(\mathbb{Z}_p) =< \omega >$ for some $\omega \in U(\mathbb{Z}_p)$ with order $p-1$,*
2. *$U(\mathbb{Z}_{p^k}) =< \omega >$ for the same ω treating as an element in $U(\mathbb{Z}_{p^k})$ with order $\phi(p^k)$.*

Proof. 1. Since $\mathbb{Z}_p$ is a field, the result follows.

2. $\omega^{p-1} = 1 \ mod \ p \Rightarrow \omega^{p-1} = 1 + lp$, where l is an integer and using the binomial expansion, we get $\omega^{p^{k-1}(p-1)} = 1 \ mod \ p^k$. Let s be the order of ω in $U(\mathbb{Z}_{p^k})$. Therefore $s \leq p^{k-1}(p-1)$. If $s < p^{k-1}(p-1)$, then $s|p^{k-1}(p-1)$ and this implies $s|p-1$ which is not possible as the order of ω in $U(\mathbb{Z}_p)$ is $p-1$. Thus we have the order of ω in $U(\mathbb{Z}_{p^k})$ as $p^{k-1}(p-1) = \phi(p^k)$.

□

Definition 1. *Let R be a ring and $\alpha \in R$. The element α is called a primitive element or primitive root if α multiplicatively generates the unit group $U(R)$ of the ring R.*

Example 1. *$U(\mathbb{Z}_5) =< 2 >$ with order 4 and $U(\mathbb{Z}_{5^3}) =< 2 >$ with order $\phi(125) = 100$. Clearly, $2 \in \mathbb{Z}_{5^3}$ is a primitive root but not a primitive root of unity in $\mathbb{Z}_{5^3}$.*

Proposition 5. *Let p be an odd prime number. Then $\mathbb{Z}_{p^k}$ is a halidon ring with index $m = p-1$ and $\omega_1 = \omega^{p^{k-1}}$ is a primitive m^{th} root of unity for positive integers $k \geq 1$.*

Proof. Since $\mathbb{Z}_p$ is a halidon ring with ω as a primitive m^{th} root of unity, $\omega^m = \omega^{p-1} = 1$ and m, $\omega^r - 1 \in U(Z_p)$ for $r = 1, 2, 3, .., m-1$. Clearly $\omega_1^m = 1$ and $\omega_1^r - 1 = \omega^{p^{k-1}r} - 1 \in U(Z_p)$ for $r = 1, 2, 3, ..m-1$. By lemma 4, $\mathbb{Z}_{p^k}$ is a halidon ring with index $m = p-1$. □

The complete characterisation of the halidon property in $\mathbb{Z}_n$, where n is odd, is given by the following theorem.

Theorem 6. *The ring $\mathbb{Z}_n$, where $n = p_1^{e_1} p_2^{e_2} p_3^{e_3} .....p_k^{e_k}$ with $2 < p_1 < p_2 < ..... < p_k$ is a halidon ring with index m and the primitive m^{th} root of unity ω if and only if each $\mathbb{Z}_{p_i^{e_i}}$ is a halidon ring with index m and primitive m^{th} root of unity $\omega_i = \omega \ mod \ p_i^{e_i}$ for each $i = 1, 2, 3, ..., k$.*

Proof. We define a map $f : \mathbb{Z}_n \to \Pi_{i=1}^k \mathbb{Z}_{p_i^{e_i}}$ by $f(a) = \Pi_{i=1}^k a \ mod \ p_i^{e_i}$. Clearly f is an isomorphism and using the proposition 2, $\omega \in U(\mathbb{Z}_n)$ is a primitive m^{th} root of unity if and only if each $\omega_i = \omega \ mod \ p_i^{e_i} \in U(\mathbb{Z}_{p_i^{e_i}})$ is a primitive m^{th} root of unity in $\mathbb{Z}_{p_i^{e_i}}$. If ω_i's are known, then we can calculate $\omega \in U(\mathbb{Z}_n)$ using the Chinese remainder theorem. □

Definition 2. *Let $p_1, p_2, p_3, ..., p_k$ be odd primes and let $\phi(x)$ be the Euler's totient function. We define the halidon function*

$$\psi(n) = \begin{cases} gcd\{\phi(p_1^{e_1}), \phi(p_2^{e_2}), \phi(p_3^{e_3}), ..., \phi(p_k^{e_k})\}, & n = p_1^{e_1}p_2^{e_2}p_3^{e_3}.....p_k^{e_k} \\ 1, & n \text{ is even} \end{cases}$$

Proposition 7. *Let n be as in definition 2. Then the halidon function*

$$\psi(n) = gcd\{p_1 - 1, p_2 - 1, p_3 - 1, ..., p_k - 1\},$$

which is independent of the exponents $e_1, e_2, e_3, ..., e_k$.

Now we can prove the following theorem, which was previously a conjecture(see [9]):

Theorem 8. *(A. Telveenus [9]) If $R = \mathbb{Z}_n$ and $n = p_1^{e_1}p_2^{e_2}p_3^{e_3}.....p_k^{e_k}$ with primes $p_1 < p_2 < p_3 < .... < p_k$ including 2, then R is a halidon ring with maximal index $m_{max} = \psi(n)$.*

Proof. Suppose n is odd. Since the map $f : \mathbb{Z}_n \to \Pi_{i=1}^k \mathbb{Z}_{p_i^{e_i}}$ by $f(a) = \Pi_{i=1}^k a \bmod p_i^{e_i}$ is an isomorphism, the order of ω, $o(\omega) = m \Leftrightarrow o(\omega_i) = m$, for $i = 1, 2, 3, .., k$. Also, $\omega_i \in \mathbb{Z}_{p_i^{e_i}} \Rightarrow \omega_i^{\phi(p_i^{e_i})} = 1 \Rightarrow m \mid \phi(p_i^{e_i}) = p_i^{e_i-1}(p_i - 1)$. Since m is even, $m \nmid p_i^{e_i-1} \Rightarrow m \mid p_i - 1$ for $i = 1, 2, 3, .., k \Rightarrow m_{max} = gcd(p_1 - 1, p_2 - 1, p_3 - 1, ..., p_k - 1) = \psi(n)$ by definition 2. If n is even, then $m = 1 = \psi(n)$ as m is invertible in R. □

From proposition 5, we have $\mathbb{Z}_{p^k}$ is a halidon ring with index $m = p - 1$ and the primitive m^{th} root of unity $\omega_1 = \omega^{p^{k-1}}$ for positive integer $k \geq 1$ where ω is a primitive m^{th} root of unity in $\mathbb{Z}_p$.

Lemma 9. *Let p be a prime number. Then the number of primitive k^{th} roots of unity in $\mathbb{Z}_p$ is $\phi(k)$.*

Proof. Since $\mathbb{Z}_p$ is a field, it is a halidon ring with maximum index $p - 1$. It is clear that every non zero element in $\mathbb{Z}_p$ is a primitive k^{th} root of unity for some positive integer $k|p - 1$. It is well known that

$$\sum_{d|p-1} \phi(d) = p - 1.$$

$\therefore$ the number of k^{th} root of unity in $\mathbb{Z}_p$ is $\phi(k)$. □

Theorem 10. *Let $n = p_1^{e_1}p_2^{e_2}p_3^{e_3}.....p_k^{e_k}$ such that $e_i > 0$ are integers and $p_i = mt_i + 1$; where m is the maximum index of the halidon ring $\mathbb{Z}_n$ and t_i's are relatively prime for all $i = 1, 2, 3..., k$. Then the number of primitive m^{th} root of unity in $\mathbb{Z}_n$ is $[\phi(m)]^k$.*

Proof. The result follows from theorem 6, theorem 8 and lemma 9. □

The following computer code based on theorem 1 computes primitive m^{th} root of unity in the halidon ring $\mathbb{Z}_n$ and verifies theorem 10.

Programme-1: To check whether Z(n) is a trivial or nontrivial halidon ring.

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
cout << "To check whether Z(n) is a trivial or nontrivial halidon ring." << endl;
unsigned long long int t = 0, n = 1, w = 1, hcf, hcf1,
d = 1, k = 1, q = 1, p = 1, b=0, c=0, temp = 1;
cout << "Enter an integer n >0: ";
cin >> n;
if (n % 2 == 0) {
cout << "Z(" << n << ") is a trivial halidon ring." << endl;
}
for (w = 1; w < n; ++w) {
for (int i = 1; i <= n; ++i) {
if (w % i == 0 && n % i == 0) {
hcf = i;

}
} if (hcf == 1) {
++t; // cout << "  " << w << "  ";
}
}
for (w = 1; w < n; ++w) {
for (int i = 1; i <= n; ++i) {
if (w % i == 0 && n % i == 0) {
hcf = i;

}
}
if (hcf == 1) {
for (int k = 1; k <= t; ++k) {
q = q * w; q = q % n;
if (q == 1) { if (temp <= k) { temp = k; } break; }
}
}
}

for (w = 2; w < n; ++w) {
for (int i = 1; i <= n; ++i) {
if (w % i == 0 && n % i == 0) {
hcf = i;
}
}
if (hcf == 1) {
```

```
for (int k = 1; k <= temp; ++k) {
q = q * w; q = q % n; if (q == 1) {
for (int i = 1; i <= n; ++i) {
if (k % i == 0 && n % i == 0) {
hcf1 = i;
}
}
if (hcf1 == 1) {
for (int j = 1; j < k; ++j)
{
if (k%j == 0) {
d = j;
for (int l = 1; l <= d; ++l)
{
p = (p*w); p = p % n;
}
for (int i = 1; i <= n; ++i) {
if ((p - 1) % i == 0 && n % i == 0) {
hcf = i;
}
}
             if (hcf == 1) {

p = 1; b = b + 1;
}
else p = 1;
c = c + 1;
      }
}
if (c == b) { cout << "   Z(" << n << ")" <<

" is a halidon ring with index m= " << k <<

" and w= " << w << "."<< endl; } {p = 1; c = 0; b = 0; }

break;
}
}
}
}
} return 0;
}
```

3 Discrete Fourier Transforms

In this section, we deal with the ring of polynomials over a halidon ring which has an application in Discrete Fourier Transforms [4]. Throughout this section, let R be a finite commutative halidon ring with index m and $R[x]$ denotes the ring of polynomials degree less than m over R. Also, refer to [9].

Definition 3. *[4] Let $\omega \in R$ be a primitive m^{th} root of unity in R and let $f(x) = \sum_{j=0}^{m-1} f_j x^j \in R[x]$ with its coefficients vector $(f_0, f_1, f_2, ..., f_{m-1}) \in R^m$. The **Discrete Fourier Transform** (DFT) is a map*

$$DFT_\omega : R[x] \to R^m$$

defined by

$$DFT_\omega(f(x)) = (f_0(1), f_1(\omega), f_2(\omega^2), ..., f_{m-1}(\omega^{m-1})).$$

Remark 1. *Clearly DFT_ω is a R-linear map as $DFT_\omega(af(x) + bg(x)) = aDFT_\omega(f(x)) + bDFT_\omega(g(x))$ for all $a, b \in R$. Also, if $R = \mathbb{C}$, the field of complex numbers, then $\omega = cos(\frac{2\pi}{m}) + isin(\frac{2\pi}{m}) = e^{i\frac{2\pi}{m}}$ and the Fourier series will become the ordinary series of sin and cos functions.*

Definition 4. *[4] The **convolution** of $f(x) = \sum_{j=0}^{m-1} f_j x^j$ and $g(x) = \sum_{k=0}^{m-1} g_k x^k$ in $R[x]$ is defined by $h(x) = f(x) * g(x) = \sum_{l=0}^{m-1} h_l x^l \in R[x]$ where $h_l = \sum_{j+k=l \ mod \ m} f_j g_k = \sum_{j=0}^{m-1} f_j g_{l-j}$ for $0 \leq l < m$.*

The notion of convolution is equivalent to polynomial multiples in the ring $R[x]/ < x^m - 1 >$. The l^{th} coefficient of the product $f(x)g(x)$ is $\sum_{j+k=l \ mod \ m} f_j g_k$ and hence

$$f(x) * g(x) = f(x)g(x) \ mod(x^m - 1).$$

Proposition 11. *[4] For polynomials $f(x), g(x) \in R[x]$, $DFT_\omega(f(x) * g(x)) = DFT_\omega(f(x)).DFT_\omega(g(x))$, where . denotes the pointwise multiplication of vectors.*

Proof. $f(x) * g(x) = f(x)g(x) + q(x)(x^m - 1)$ for some $q(x) \in R[x]$.
Replace x by ω^j, we get

$$f(\omega^j) * g(\omega^j) = f(\omega^j)g(\omega^j) + 0.$$
$$\therefore \quad DFT_\omega(f(x) * g(x)) = DFT_\omega(f(x)).DFT_\omega(g(x)).$$

□

Theorem 12. *For a polynomial $f(x) \in R[x]$, $DFT_\omega^{-1}(f(x)) = \frac{1}{m} DFT_{\omega^{-1}}(f(x))$.*

Proof. The matrix of the transformation $DFT_\omega(f(x))$ is

$$[DFT_\omega(f(x))] = \phi = \begin{pmatrix} 1 & 1 & 1 & ..... & 1 \\ 1 & \omega & \omega^2 & ..... & \omega^{m-1} \\ 1 & \omega^2 & (\omega^2)^2 & ..... & (\omega^2)^{m-1} \\ . & . & . & ..... & . \\ . & . & . & ..... & . \\ . & . & . & ..... & . \\ 1 & \omega^{m-1} & (\omega^{m-1})^2 & ..... & (\omega^{m-1})^{m-1} \end{pmatrix}$$

The matrix ϕ is the well known Vandermonde matrix and its inverse is $\frac{1}{m}\phi^*$, where ϕ^* is the matrix transpose conjugated [1]. Since ϕ is a square matrix and the conjugate of ω is ω^{-1}, we have $DFT_\omega^{-1}(f(x)) = \frac{1}{m}DFT_{\omega^{-1}}(f(x))$. □

Example 2. *We know that $R = Z_{49}$ is a halidon ring with index $m = 6$ and $\omega = 19$. Also, $\omega^{-1} = \omega^5 = 31$. Let $f(x) = 2 + x + 2x^2 + 3x^3 + 5x^4 + 10x^5 \in R[x]$. Then $DFT_\omega(f(x))$ can be expressed as*

$$\begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 \\ 1 & \omega^2 & \omega^4 & 1 & \omega^2 & \omega^4 \\ 1 & \omega^3 & 1 & \omega^3 & 1 & \omega^3 \\ 1 & \omega^4 & \omega^2 & 1 & \omega^4 & \omega^2 \\ 1 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} \Rightarrow \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \end{pmatrix} = \begin{pmatrix} 23 \\ 24 \\ 32 \\ 44 \\ 9 \\ 27 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} =$$

$$6^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \\ 1 & \omega^4 & \omega^2 & 1 & \omega^4 & \omega^2 \\ 1 & \omega^3 & 1 & \omega^3 & 1 & \omega^3 \\ 1 & \omega^2 & \omega^4 & 1 & \omega^2 & \omega^4 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 \end{pmatrix} \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = 41 \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 31 & 30 & 48 & 18 & 19 \\ 1 & 30 & 18 & 1 & 30 & 18 \\ 1 & 48 & 1 & 48 & 1 & 48 \\ 1 & 18 & 30 & 1 & 18 & 30 \\ 1 & 19 & 18 & 48 & 30 & 31 \end{pmatrix} \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \end{pmatrix}$$

If $\begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \end{pmatrix} = \begin{pmatrix} 23 \\ 24 \\ 32 \\ 44 \\ 9 \\ 27 \end{pmatrix}$, *then a direct calculation gives* $\begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 2 \\ 3 \\ 5 \\ 10 \end{pmatrix}$

as expected.

Programmes 2 and 3 will enable us to calculate Discrete Fourier Transform and its inverse. We can cross-check the programmes against example 2.

`Programme-2: Discrete Fourier Transform`

```
#include <iostream>
#include<cmath>
using namespace std;
int main()
{
cout << "Discrete Fourier Transform" << endl;
unsigned long long int a[1000][1000], b[1000][1000],
mult[1000][1000],q=1,m=1, n=1, w2=1,w=1, r1, c1, r2,
c2, i, j, k, t=1;
cout << "Enter n,m,w: ";
cin >> n >> m >> w;
r1 = m; c1=m;
r2 = m; c2=1;
for (i = 0; i < r1; ++i)
for (j = 0; j < c1; ++j)
{
 t = (i*j)%m;
 if (t == 0) a[i][j] = 1;
 else
for (q = 1; q < t + 1; ++q) { w2 = (w2 * w) % n; }
 a[i][j] = w2; w2 = 1;
}
for (i = 0; i < r1; ++i)
for (j = 0; j < c1; ++j)
   {
cout<<"  a"<<i+1<<" "<<j+1<<"="<<a[i][j] ;
if (j == c1 - 1)
cout << endl;
}
cout << endl << "Enter coefficient vector of
    the polynomial:" << endl;
for (i = 0; i < r2; ++i)
for (j = 0; j < c2; ++j)
{
    cout << "Enter element f" << i << " = ";
        cin >> b[i][j];
}
for (i = 0; i < r1; ++i)
for (j = 0; j < c2; ++j)
{
     mult[i][j] = 0;
}
for (i = 0; i < r1; ++i)
for (j = 0; j < c2; ++j)
```

```
for (k = 0; k < c1; ++k)
{
     mult[i][j] += (a[i][k]) * (b[k][j]);
}
cout << endl << "DFT Output: " << endl;
for (i = 0; i < r1; ++i)
for (j = 0; j < c2; ++j)
{
cout << "F"<< i << "="<< mult[i][j]%n;
if (j == c2 - 1)
cout << endl;
}
return 0;
        }
```

Programme-3: Inverse Discrete Fourier Transform

```
#include <iostream>
#include<cmath>
using namespace std;
int main()
{
cout << "Inverse Discrete Fourier Transform" << endl;
unsigned long long int a[1000][1000], b[1000][1000],
    mult[1000][1000], p=1, q=1, l=1, m = 1, m1 = 1, w1 = 1,
    w2=1, n = 1, w = 1, r1, c1, r2, c2, i, j, k,
    c=1,t = 1;
cout << "Enter n,m, w: ";
cin >> n >> m >> w;
for (l = 1; l < n; ++l)
        {
c = (l * m) % n;
if (c == 1)
        {
m1 = l;
}
    }
for (p = 1; p < m; ++p)
     {
w1 = (w1 * w) % n;
    }
r1 = m; c1 = m;
r2 = m; c2 = 1;
for (i = 0; i < r1; ++i)
```

```
for (j = 0; j < c1; ++j)
{
t = (i * j) % m;
if (t == 0) a[i][j] = 1;
else
for (q = 1; q < t + 1; ++q) { w2 = (w2 * w1) % n; }
a[i][j] = w2; w2 = 1;
}
for (i = 0; i < r1; ++i)
for (j = 0; j < c1; ++j)
{
cout << "  a" << i + 1 << j + 1 << "=" << a[i][j];
if (j == c1 - 1)
cout << endl;
}
cout << endl << "Enter DFT vector :" << endl;
for (i = 0; i < r2; ++i)
for (j = 0; j < c2; ++j)
{
cout << "Enter element F" << i << " = ";
cin >> b[i][j];
}
for (i = 0; i < r1; ++i)
for (j = 0; j < c2; ++j)
{
mult[i][j] = 0;
}
for (i = 0; i < r1; ++i)
for (j = 0; j < c2; ++j)
for (k = 0; k < c1; ++k)
{
mult[i][j] += (a[i][k]) * (m1 * b[k][j]);
    }
cout << endl << "Polynomial vector: " << endl;
for (i = 0; i < r1; ++i)
for (j = 0; j < c2; ++j)
{
cout << "f" << i << "=" << mult[i][j] % n;
if (j == c2 - 1)
cout << endl;
}
return 0;
        }
```

If $R = Z_{100001}$, $m = 10$, $\omega = 26364$ and $f(x) = 1 + 2x + 3x^2 + 4x^3 + 5x^4 + 6x^5 + 7x^6 + 8x^7 + 9x^8 + x^9 \in R[x]$, then $\begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \\ F_9 \end{pmatrix} = \begin{pmatrix} 46 \\ 19019 \\ 3314 \\ 10082 \\ 48017 \\ 4 \\ 80347 \\ 18172 \\ 68413 \\ 52627 \end{pmatrix}$.

Also, we can verify the inverse DFT using the above data.

The following proposition from number theory is very useful in the next section.

Proposition 13. *[3] Let $n = p_i^{e_1} p_2^{e_2} ....p_k^{e_k}$ be the standard form of the integer n and let d,e satisfy $ed \equiv 1$ mod $\phi(n)$. Then for all integer x,*

$$x^{ed} \equiv x \ mod \ n.$$

Therefore, if $c = x^e \mod n$, we have $x \equiv c^d$ mod n.

Let

$$u = \sum_{i=1}^{m} \alpha_i g_i$$

be an element in the group algebra RG and let

$$\lambda_r = \sum_{i=1}^{m} \alpha_{m-i+2} (\omega^{(i-1)})^{(r-1)} \tag{1}$$

where $\omega \in R$ is a primitive m^{th} root of unity. Then u is said to be *depending* on $\lambda_1, \lambda_2, ..., \lambda_m$.

Theorem 14. *Let*

$$u = \sum_{i=1}^{m} \alpha_i g_i \in U(RG)$$

be depending on $\lambda_1, \lambda_2, ..., \lambda_m$. Let

$$v = \sum_{i=1}^{m} \beta_i g_i$$

be the multiplicative inverse of u in RG. Then

$$\beta_i = \frac{1}{m} \sum_{r=1}^{m} \lambda_r^{-1} (\omega^{i-1})^{r-1}.$$

Computer programme-4

```
#include<iostream>
#include<cmath>
using namespace std;
int main() {
cout << "To find the units in Z(n)G;" <<
"G is a cyclic group of order m through " <<
"lamda take units in R." << endl;
long long int  a[1000], l[1000], w1[1000], m = 1, t = 0, x = 1, y = 1, s = 0,
s1 = 0, m1 = 1, n = 1, i = 1, p=1,k = 0, r = 1, w = 1;

cout << "Enter n =" << endl;
cin >> n;
cout << "Enter index m =" << endl;
cin >> m;
cout << "Enter m^(-1) =" << endl;
cin >> m1;
cout << "Enter primitive m th root w =" << endl;
cin >> w;
w1[0] = 1; cout << "w1[0]=" << w1[0] << endl;
for (i = 1; i < m; ++i) {

w1[i] = p * w % n; p = w1[i];
cout << "w1[" << i << "]=" << w1[i] << endl;
}
cout << "Enter lamda values which are units" << endl;
for (int i = 1; i < m + 1; ++i) {
cout << "l[" << i << "]=" << endl;
cin >> l[i];
}
for (int r = 1; r < m + 1; ++r) {
for (int j = 1; j < m + 1; ++j)
{
x = ((j - 1) * (r - 1)) % m;
k = k + (m1 * l[j] * w1[x]) % n; k = k % n;
// cout << "k=" << k << endl;
} a[r] = k; cout << "a[" << r << "]=" << a[r] << endl;
k = 0;
}
cout << "The unit in RG is  u= ";
s = 1;
mylabel:
cout << a[s] << "g^(" << s - 1 << ") + ";
s++;
if (s < m + 1) goto mylabel; cout << endl;
cout << endl;
```

```
cout << "Note: Please neglect the last + as it is unavoidable for a for loop.";
return 0;
}
```

Computer Programme-5

```
#include<iostream>
#include<cmath>
using namespace std;
int main() {
cout << "To check whether an element in Z(n)G;" <<
"G is a cyclic group of order m" <<"has a multiplicative inverse or not" << endl;
long long int a[1000], b[1000], c[1000], d[1000], e[1000], w1[1000], m = 1,
t = 0, x = 1, s = 0, s1 = 0, l = 0, m1 = 1, hcf = 1,
n = 1, i = 1, k = 0, q = 1, p = 1, r = 1, w = 1;
cout << "Enter n =" << endl;
cin >> n;
cout << "Enter index m =" << endl;
cin >> m;
cout << "Enter  m^(-1) =" << endl;
cin >> m1;
cout << "Enter primitive m th root w =" << endl;
cin >> w;
w1[0] = 1;
for (i = 1; i < m; ++i)
{

w1[i] = p * w % n; p = w1[i];
cout << "w1[" << i << "]" << w1[i] << endl;
}
for (int i = 1; i < m + 1; ++i) {
cout << "Enter a[" << i << "]=" << endl;
cin >> a[i];
}
a[0] = a[m];
for (int r = 1; r < m + 1; ++r) {
for (int j = 1; j < m + 1; ++j)
{
l = (m - j + 2) % m;
x = ((j - 1) * (r - 1)) % m;
k = k + (a[l] * w1[x]) % n; k = k % n;
// cout << "k=" << k << endl;
} c[r] = k; cout << "lambda[" << r << "]=" << c[r] << endl;
k = 0;
```

```
}
for (r = 1; r < m + 1; ++r) {
for (int i = 1; i <= n; ++i) {
if (c[r] % i == 0 && n % i == 0) {
hcf = i;
}
}
if (hcf == 1) {
cout << "lambda[" << r << "] is a unit" << endl;
}
else {
cout << "lambda[" << r <<
"] is a not unit. So there is no multiplicative inverse." <<
endl; t = 1;
}
}
for (r = 1; r < m + 1; ++r) {
for (int i = 1; i <= n; ++i) {
e[r] = (c[r] * i) % n;
if (e[r] == 1) {
b[r] = i;
cout << " The inverse of lambda[" << r << "] is " << b[r] << endl;
}
}
}
b[0] = b[m];
for (int r = 1; r < m + 1; ++r) {
for (int j = 1; j < m + 1; ++j)
{
l = (m - j + 2) % m;
x = (m * m - (j - 1) * (r - 1)) % m;  //cout << "x= " << x << endl;
k = k + (m1 * b[l] * w1[x]) % n; k = k % n;
//cout << "k=" << k << endl;
}
d[r] = k; //cout << "d[" << r << "]=" << d[r] << endl;
k = 0;
}
if (t == 1) {
s = m;
mylabel2:
cout << a[m - s + 1] << "g^(" << m - s << ") + ";
s--;
if (s > 0) goto mylabel2; cout <<
"has no multiplicative inverse." << endl;
}
```

```
else {
cout << "The inverse of ";
s = m;
mylabel:
cout << a[m - s + 1] << "g^(" << m - s << ") + ";
s--;
if (s > 0) goto mylabel; cout << "is" << endl;
s1 = m;
mylabel1:
cout << d[m - s1 + 1] << "g^(" << m - s1 << ") + ";
s1--;
if (s1 > 0) goto mylabel1; cout << "." << endl;
}
return 0;
}
```

The computer programme-3 can be used to test whether a given element u in RG is a unit or not. If it is a unit, then the programme will give the multiplicative inverse v in RG.

Theorem 15. *Let*

$$u = \sum_{i=1}^{m} \alpha_i g_i \in RG$$

be depending on $\lambda_1, \lambda_2, ..., \lambda_m$. *Then*

1. $u \in U(RG)$ *if and only if each* $\lambda_i \in U(R)$,
2. $u \in E(RG)$ *if and only if each* $\lambda_i \in E(R)$, *where* $E(RG)$ *is the set of idempotents in* RG.

More over, $|U(RG)| = |U(R)|^{|G|}$ *and* $|E(RG)| = |E(R)|^{|G|}$.

4 RSA-DFT Cryptosystem

Let m be the length of the message including the blank spaces between the words. If the message has a length more than m, we can split the message into blocks with lengths less than m. For a message of length less than m, we can add blank spaces after the period to make it a message with length m. Choose large prime numbers such that $p_i = mt_1 + 1$ where $i = 1, 2, 3, ...k$ and t_i's are relatively prime. Let ω be a primitive m^{th} root of unity in $\mathbb{Z}_n$, where $n = p_1^{e_1} p_2^{e_2} ....p_k^{e_k}$ for some positive integers e_i. We know that $\mathbb{Z}_n$ is a halidon ring with maximum index $gcd(p_1 - 1, p_2 - 1, ...p_k - 1)$(theorem 8) and since $m|p_1 - 1, p_2 - 1, ...p_k - 1$, $\mathbb{Z}_n$ is also a halidon ring with index m(proposition 3).

Here we are considering a cryptosystem based on modulo n. The following table gives numbers and the corresponding symbols.

Numbers assigned	Symbols
0 to 9	0 to 9
10 to 35	A to Z
36	blank space
37	colon
38	period
39	hyphen

In this cryptosystem, there are two stages. In stage 1, we shall compute the value of ω which Bob keeps secret and in stage 2, we shall decrypt the message sent by Bob.

Stage 1-RSA

Cryptosystem setup

1. Alice chooses large primes $p_1, p_2, ...p_k$ and positive integers $e_1, e_2, ...e_k$, and calculates $n = p_1^{e_1} p_2^{e_2} ... p_k^{e_k}$ and $\phi(n) = p_1^{e_1-1} p_2^{e_2-1} ... p_k^{e_k-1}(p_1 - 1)(p_2 - 1)...(p_k - 1)$.
2. Alice chooses an e so that $gcd(e, \phi(n)) = 1$.
3. Alice calculates d with property $ed \equiv 1 mod\ \phi(n)$.
4. Alice makes n and e public and keeps the rest secret.

Cryptosystem Encryption(Programme-1)

1. Bob looks up Alice's n and e .
2. Bob chooses an arbitrary $\omega\ mod\ n$ and kept secret.
3. Bob sends $c \equiv \omega^e\ mod\ n$ to Alice.

Cryptosystem Decryption(Proposition 13)

1. Alice receives c from Bob.
2. Alice computes $\omega \equiv c^d mod\ n$.

Stage 2-Discrete Fourier Transform

Cryptosystem setup

1. Alice chooses Discrete Fourier Transform as the encryption key.
2. Alice chooses Inverse Discrete Fourier Transform as the decryption key.

Cryptosystem Encryption(Programme-2)

1. Bob looks up Alice's encryption key.
2. Bob writes his message x.
3. Bob computes $y = DFT\ x$ his chosen ω.
4. Bob sends y to Alice.

Cryptosystem Decryption(Programme-3)

1. Alice receives y from Bob.
2. Alice computes $x = Inverse\ DFT\ y$ with ω calculated in stage 1.

The above cryptosystem is an **asymmetric cryptosystem** as Alice and Bob share different information. For the practical application, we must choose very large prime numbers (more than 300 digits) so that the calculation of $\phi(n)$ must be very difficult

and the probability of choosing the primitive m^{th}root of unity ω should tends to zero. We exhibit the working of the RSA-DFT cryptosystem using a simple choice of prime numbers in which the probability of choosing the primitive m^{th}root of unity ω is $\frac{1}{10000} = 0.0001 = 0.01\%$ in the following example.

Example 3. ***Stage 1***

The length of the message has been fixed as $m = 202$. *Alice chooses two primes* $p_1 = 607$*and* $p_2 = 809$ *and two positive integers* $e_1 = 1$ *and* $e_2 = 1$, *and calculates* $n = 491063$ *and* $\phi(n) = 489648$.

Alice chooses an $e = 361123$ *so that* $gcd(e, \phi(n)) = 1$.

Alice calculates $d = 18523$ *with property* $ed \equiv \ 1 \ mod \ \phi(n)$. *Alice shared Bob* $n = 491063$ *and* $e = 361123$ *and rest kept secret.*

Bob looks up Alice's $n = 491063$ *and* $e = 361123$.

Bob chooses an arbitrary 202^{th} *root of unity* ω *mod n(there are* $\phi(202)^2 = 10000$ ω*'s possible and they can be found by running the programme* 1 *and it will take around* 8 *hours, and its probability is* $\frac{1}{10000} = 0.0001$) *kept secret.*

Bob sends $c \equiv \omega^e \ mod \ n \equiv 142638 \ \ mod \ 491063$ *to Alice. Alice receives c from Bob.*

Alice computes $\omega \equiv \ c^d mod \ n \equiv 239823 \ mod \ 491063$.

Stage 2

Alice shared Bob the encryption key Discrete Fourier Transform and $n = 491063$. *Suppose Bob sends the following secret message to Alice.*

MY BANK DETAILS: NAME: JACK CARD NUMBER: 4125678 SORT CODE:20-30-41 ACCOUNT NUMBER:20164 BANK:OVERSEAS.

The length of the message is 101 and to make it 202 we need to add 101 blank spaces. This can be translated into a 202 component vector

$$\left(\begin{array}{c}
22\ 34\ 36\ 11\ 10\ 23\ 20\ 36\ 13\ 14\ 29\ 10\ 18\ 21\ 28\ 37 \\
36\ 23\ 10\ 22\ 14\ 37\ 36\ 19\ 10\ 12\ 20\ 36\ 12\ 10\ 27\ 13 \\
36\ 23\ 30\ 22\ 11\ 14\ 27\ 37\ 4\ 1\ 2\ 5\ 6\ 7\ 8 \\
36\ 28\ 24\ 27\ 29\ 36\ 12\ 24\ 13\ 14\ 37\ 2\ 0\ 39\ 3\ 0\ 39\ 4\ 1 \\
36\ 10\ 12\ 12\ 24\ 30\ 23\ 29\ 23\ 30\ 22\ 11\ 14\ 27\ 37\ 2\ 0\ 1\ 6\ 4 \\
36\ 11\ 10\ 23\ 20\ 37\ 24\ 31\ 14\ 27\ 28\ 14\ 10\ 28\ 38 \\
36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36 \\
36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36 \\
36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36 \\
36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36 \\
36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36
\end{array}\right)$$

Bob chooses a primitive 202^{th} *roots of unity* ω *in stage 1 using programme 1 and kept secret in the halidon ring* $\mathbb{Z}_{491063}$ *with index 202. Applying DFT (see programme 2) to the plain text to get the following cipher text using the chosen value of* ω*:*

$$\begin{pmatrix} 5640\ 28875\ 82477\ 377806\ 380572\ 399487\ 350120\ 214346 \\ 101686\ 277011\ 93173\ 220930\ 72573\ 42514\ 361289 \\ 476177\ 371780\ 243907\ 179047\ 292166\ 427665\ 243623\ 344397 \\ 155022\ 360049\ 312478\ 305875\ 392901\ 193460\ 440042 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 216155\ 440701\ 157904\ 342869\ 348795\ 159340\ 140193\ 222089 \\ 326519\ 95581\ 431250\ 15009\ 166938\ 384271\ 452109 \end{pmatrix}$$

The readers can check the above results by copying the programmes and paste in Visual Studio 2022 c++ projects.

Alice receives the above cipher text and she uses $\omega = 239823$ *from stage 1. Applying the inverse DFT (see programme 3)Alice gets the original message back. Also, we can assign letters and numbers in 40! ways which will also make the adversaries their job difficult. For messages with length more than* m*, split the message into blocks with length less than* m*.*

The Security of RSA [6]

Five possible approaches to attacking the RSA algorithm are:

- Brute force: This involves trying all possible private keys. To defend against this attack, use a large key space.
- Mathematical attacks: There are several approaches, all equivalent in effort to factoring n into standard form. To overcome this threat take n as product of two large primes with at least 300 digits.
- Timing attacks: These depend on the running time of the decryption algorithm. They can be countered by constant exponentiation time, random delays and blinding.
- Hardware fault-based attack: This involves inducing hardware faults in the processor that is generating random digital signatures. This is not a serious threat as it requires that the attacker have physical access to the target machine and that the attacker is able to directly control the input power to the processor .
- Chosen ciphertext attacks: This type of attack exploits properties of the RSA algorithm. To overcome this simple attack, practical RSA-based crptosystems randomly pad the plaintext prior to encryption.

5 RSA-HGR Cryptosystem

Here we are considering a cryptosystem based on modulo n. The following table gives numbers and the corresponding symbols.

Symbols	u values which are units $\frac{\phi(n)!}{(\phi(n)-40)!}$ assignments uniquely
0 to 9	u_1 to u_{10}
A to Z	u_{11} to u_{36}
blank space	u_{37}
colon	u_{38}
period	u_{39}
hyphen	u_{40}

Note: For $n = 100, \phi(n) = 40$. So there are $40! = 815,915,283,247,897,734,345,611,269,596,115,894,272,000,000,000$ assignments of units to symbols.

In this cryptosystem also, there are two stages. In stage 1, we shall compute the value of ω which Bob keeps secret and in stage 2, Alice shall decrypt the message sent by Bob.

In RSA, the challenge of adversaries is to find the value of $\phi(n)$. But here they have an extra challenge of locating or calculating the value of ω.

Stage 1-RSA

Stage 1 is same as above.

Stage 2-Halidon Group Ring (HGR)

Cryptosystem setup

1. Alice chooses programme 4 as the encryption key.
2. Alice chooses programme 5 as the decryption key.

Cryptosystem Encryption(Equation 1)

1. Bob looks up Alice's encryption key.
2. Bob writes his message $x = x_1x_2x_3 \cdots x_m$.
3. Bob translates x into $y = \lambda_1\lambda_2\lambda_3 \cdots \lambda_m$ using the table.
4. Bob calculates the coefficients of the corresponding unit using the programme 4 and the chosen value of ω in stage 1.
5. Bob sends coefficients to Alice.

Cryptosystem Decryption (Theorem 14)

1. Alice receives coefficients from Bob.
2. Alice computes $\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_m$ using programme 5 with ω calculated in stage 1.
3. Alice recovers the message using the table.

Example 4. ***Stage 1***

Same as example 3.

Stage 2

Public Key

Symbols	*Unit in* $\mathbb{Z}_{491063}$ *assigned*	*Symbols*	*Unit in* $\mathbb{Z}_{491063}$ *assigned*
0	*221373*	*K*	*80303*
1	*389086*	*L*	*52853*
2	*21415*	*M*	*80303*
3	*428230*	*N*	*52853*
4	*162920*	*O*	*114288*
5	*126345*	*P*	*473119*
6	*81308*	*Q*	*323343*
7	*490630*	*R*	*26857*
8	*22673*	*S*	*91043*
9	*4004*	*T*	*98057*
A	*162483*	*U*	*150255*
B	*2255*	*V*	*24495*
C	*183775*	*W*	*86867*
D	*4129*	*X*	*176089*
E	*221927*	*Y*	*206140*
F	*437699*	*Z*	*461772*
G	*275130*	*BLANK SPACE*	*348362*
H	*50473*	*COLON*	*90605*
I	*123651*	*PERIOD*	*5932*
J	*114773*	*HYPHEN*	*275062*

Alice shared Bob the encryption key Discrete Fourier Transform and $n = 491063$. *Suppose Bob sends the following secret message to Alice.*

AN IMMINENT ATTACK ON YOU WILL HAPPEN TOMORROW EVENING AT 5:30 PM. BE ALERT AND TAKE PRECAUTIONS.

The length of the message is 97 and to make it 202 we need to add 105 blank spaces. This can be translated into a 202 component vector.

$$\begin{pmatrix} 162483\ 52853\ 348362\ 123651\ 80303\ 80303\ 123651\ 52853\ 221927\ 52853 \\ 98057\ 348362\ 162483\ 98057\ 98057\ 162483\ 183775\ 80303\ 348362\ 114288 \\ 52853\ 348362\ 206140\ 114288\ 150255\ 348362\ 86867\ 123651\ 52853\ 52853 \\ 348362\ 50473\ 162483\ 473119\ 473119\ 221927\ 52853\ 348362\ 98057\ 114288 \\ 80303\ 114288\ 26857\ 26857\ 114288\ 86867\ 348362\ 221927\ 24495\ 221927 \\ 52853\ 123651\ 52853\ 275130\ 348362\ 162483\ 98057\ 348362\ 126345\ 90605 \\ 428230\ 221373\ 348362\ 473119\ 80303\ 5932\ 348362\ 2255\ 221927\ 348362 \\ 162483\ 52853\ 221927\ 26857\ 98057\ 348362\ 162483\ 52853\ 4129\ 348362 \\ 98057\ 162483\ 80303\ 221927\ 348362\ 473119\ 26857\ 221927\ 183775\ 162483 \\ 150255\ 98057\ 123651\ 114288\ 52853\ 91043\ 5932\ 348362\ 348362\ 348362 \\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362 \\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362 \\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362 \\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362 \\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362 \\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362 \\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362 \\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362 \\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362 \\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362\ 348362 \\ 348362\ 348362 \end{pmatrix}$$

Bob chooses a primitive 202^{th} *roots of unity* ω *in stage 1 using programme 1 and kept secret in the halidon ring* $\mathbb{Z}_{491063}$ *with index 202. Using programme 4, Bob converts the plain text into the following cipher text in terms of coefficents.*

$$\begin{pmatrix} 252493\ 450589\ 460479\ 204758\ 233506\ 353306\ 421232\ 356924\ 301091\ 289893 \\ 288179\ 242097\ 326234\ 13515\ 346524\ 267905\ 60544\ 1589\ 224877\ 392891 \\ 393603\ 346149\ 126356\ 374713\ 42452\ 30660\ 444474\ 328107\ 278316\ 320329 \\ 215968\ 8062\ 69501\ 442389\ 463363\ 20437\ 184879\ 111644\ 215157\ 487962 \\ 182507\ 157039\ 200299\ 355976\ 90232\ 362884\ 407252\ 282817\ 324527\ 299628 \\ 83392\ 380613\ 274931\ 455342\ 28745\ 445319\ 430230\ 446985\ 347595\ 201469 \\ 91852\ 53863\ 48802\ 172649\ 95573\ 70434\ 71251\ 95329\ 257149\ 125640 \\ 436246\ 37716\ 452002\ 143402\ 221576\ 137122\ 379802\ 91038\ 217808\ 73515 \\ 245279\ 62765\ 16846\ 473375\ 284904\ 470346\ 392515\ 31311\ 386722\ 228015 \\ 471883\ 95686\ 284880\ 373228\ 282251\ 461945\ 347587\ 372751\ 243942\ 339087 \\ 441737\ 321411\ 205845\ 172853\ 450407\ 431493\ 72268\ 378074\ 403244\ 261526 \\ 363362\ 372773\ 193094\ 61896\ 76335\ 442360\ 12418\ 333213\ 349588\ 137997 \\ 465244\ 464347\ 453371\ 370624\ 414389\ 329819\ 99661\ 168143\ 270109\ 194801 \\ 460848\ 483049\ 98372\ 225436\ 184156\ 147000\ 137130\ 254978\ 435708\ 227589 \\ 126220\ 45283\ 312941\ 108458\ 176782\ 55396\ 134718\ 440134\ 367637\ 450466 \\ 32149\ 44665\ 445959\ 120765\ 447216\ 362999\ 402427\ 210408\ 171884\ 486885 \\ 280531\ 322673\ 116715\ 483483\ 398994\ 31300\ 134031\ 431195\ 434524\ 172474 \\ 198368\ 111628\ 469394\ 198059\ 11214\ 387413\ 93105\ 390274\ 263412\ 304750 \\ 333166\ 415475\ 31915\ 125737\ 36184\ 115899\ 390465\ 6472\ 173688\ 208819 \\ 168514\ 197636\ 136348\ 410545\ 200343\ 316617\ 47292\ 286043\ 112122\ 239726 \\ 361815\ 85601 \end{pmatrix}$$

Alice receives the above cipher text and she uses $\omega = 239823$ from stage 1. Applying the programme 5 Alice gets the original message back.

6 Conclusion

These new cryptosystems have been developed using halidon rings, halidon group rings and and Discrete Fourier Transforms. These systems provides high-level security for communication between ordinary people or classified messages in government agencies. The level of security can be increased by utilising advanced computer technology and powerful codes to calculate the primitive m^{th} root of unity for a very large value of n where the calculation of $\phi(n)$ is difficult. There are scopes for the development of new cryptosystems based on Cyclotomic polynomials..

References

[1] P.J.Davis *Circulant Matrices* John Wiley & Sons, New York(1979).

[2] G. Higman *The Units of Group Rings* Proc. London Math. Soc. **46**(1940), 231-248.

[3] James S. Kraft and Lawrence C. Washington *An Introduction to Number Theory and Cryptogrphy* CRC Press, Taylor & Francis Group, Florida(2014)

[4] Joachim von zur Gathen and Jürgen Gerhard *Modern Computer Algebra, second edition* Cambridge University Press, Cambridge (2003).

[5] G. Karpilovsky *Commuataive Group Algebras* Marcel Dekker, No.78(1983).

[6] William Stalling *Cryptography and Network Security, eighth edition* Perason Education Limited (2023).

[7] A. Telveenus *Circulants and Group Algebras* Hadronic Journal Supplement, Vol. 14, pp 227-288 (1999).

[8] A. Telveenus *Halidon Rings and Group Algebras* Algebras, Groups and Geometries 18, pp 317-330 (2001).

[9] A. Telveenus *Halidon Rings and their Applictions* Algebras, Groups and Geometries 37, pp 193-247 (2021).