RSA Encryption Using Polynomial Rings

Michelle Freed

April 13, 2018

Abstract

This paper examines some of the complexities of the public key encryption system RSA. RSA provides the necessary steps to encrypt messages using modular arithmetic. It doesn't require separate parties to exchange keys because the encrypting key is published for anyone to use. This creates a secure system whose strength lies in the difficulty to factor the product of two large (500-600 digits) prime numbers. RSA was originally written to be used over the ring of integers. This paper gives the necessary mathematical background needed to extend RSA to be used over the ring of polynomials using the quotient ring $\mathbb{F}_2[x]/n(x)$.

Contents

1 Introduction

The RSA public key encryption algorithm (named after its inventors Rivest, Shamir, and Adleman) was created because of the growing need for privacy and authentication when using digital communication channels as the use of technology throughout the world increases steadily [4]. Public key encryption like RSA is important because a courier isn't required to exchange needed encrypting and decryption keys for long distance transmission. This courier isn't needed with public key because the encryption key is made public for anyone to use. The publisher can be confident that any messages sent to him or her using this encryption key will remain private because he or she is the only one with the corresponding decryption key.

The remainder of this paper is organized as follows: Section 2 provides background information including different mathematical definitions and theorems which are needed to understand the implementation of RSA over the ring of integers $\mathbb{Z}$; Section 3 details how the RSA encryption process is implemented and used with integers; Section 4 provides background information including different mathematical definitions and theorems which are needed to understand the implementation of RSA over the ring of polynomials using the ring $\mathbb{F}_2[x]/n(x)$; Section 5 details how the RSA encryption process is implemented and used with polynomials; Section 6 presents areas where future work can expand the study; and Section 7 provides concluding remarks.

2 Properties of the Ring of Integers

In order to understand the main ideas and implementation of the RSA algorithm, we must first understand some elements of the algebraic structure of the ring of integers $\mathbb{Z}$. This section examines the relevant definitions and theorems.

2.1 Division Algorithm for $\mathbb{Z}$

Definition 1. Let a and $b \neq 0$ be two integers. We say that b **divides** a (written $b \,|\, a$) if and only if there exists an integer c such that $a = b \cdot c$.

Definition 2. Let a and $b \neq 0$ be two integers. The **greatest common divisor** of a and b, denoted $\gcd(a, b)$, is the greatest positive integer that divides both a and b.

Theorem 3 (Division Algorithm)**.** *Let a and $b > 0$ be two integers. There exists unique integers q and r such that $a = b \cdot q + r$ with $0 \leq r < b$.*

Proof. See Theorem 6.3 in [1]. □

Lemma 4. Let a and $b \neq 0$ be two integers. Assume that $a = bq + r$ as in the division algorithm. Then $\gcd(a, b) = \gcd(b, r)$.

Proof. See Lemma 1.5 in [2] □

2.2 Euclidean Algorithm

The Euclidean Algorithm is used to find the gcd $= d$ of integers a and b when $a > b$. We use Theorem 3 to divide b into a as follows,

$$a = q_1 b + r_1 \quad \text{with} \quad 0 \leq r_1 < b.$$

If $r_1 = 0$ then $b|a$ and $d = b$. If not, then we divide r_1 into b and write,

$$b = q_2 r_1 + r_2 \quad \text{with} \quad 0 \leq r_2 < r_1.$$

Since $\gcd(a, b) = \gcd(b, r_1)$ by Lemma 4, if $r_2 = 0$ then $d = r_1$. If not, then we continue in this way until the remainder equals 0.

$$\begin{aligned} a &= q_1 b + r_1 \\ b &= q_2 r_1 + r_2 \\ r_1 &= q_3 r_2 + r_3 \\ &\vdots \\ r_{k-2} &= q_k r_{k-1} + r_k \end{aligned}$$

Finally, when $r_k = 0$, $d = \gcd(a, b) = r_{k-1}$.

Example 5. Let us find the greatest common divisor of 36 and 15. We use the Euclidean Algorithm as follows:

$$\begin{aligned} 36 &= 2 \cdot 15 + 6 \\ 15 &= 2 \cdot 6 + 3 \\ 6 &= 2 \cdot 3 + 0 \\ 3 &= \gcd(36, 15) \end{aligned}$$

2.3 Bezout's Identity and The Extended Euclidean Algorithm

Theorem 6 (Bezout's Identity)**.** *Let a and b be two nonzero integers then there exists integers u and v such that* $\gcd(a, b) = ua + vb$.

Proof. See Theorem 1.7 in [2]. □

In order to find the Bezout numbers u and v, we must perform the Extended Euclidean Algorithm which is essentially the Euclidean Algorithm in reverse order. Start with the $\gcd(a, b) = d$, which was the last non-zero remainder r_{k-1} found by preforming the Euclidean Algorithm. Solve for r_{k-1} in the second to last equation to get

$$r_{k-1} = r_{k-3} - q_{k-1} \cdot r_{k-2}$$

which expresses d as a multiple of r_{k-3} and r_{k-2}. Next, use the previous equation to solve for r_{k-2},

$$r_{k-2} = r_{k-4} - q_{k-2} \cdot r_{k-3}.$$

Eliminate r_{k-2} by substituting the second equation into the first to express d as a multiple of r_{k-3} and r_{k-4}. Repeat this until all of the remainders $r_{k-3}, r_{k-4}, ...$ are eliminated. This results in an equation that expresses d as a linear combination of a and b with integer coefficients u and v, $d = ua + vb$.

Example 7. Find the Bezout numbers for 15 and 36. Returning to our example from above where the $\gcd(15, 36) = 3$, we have

$$\begin{aligned} 3 &= 1 \cdot 15 - 2 \cdot 6 \\ &= 1 \cdot 36 - 2 \cdot 15 \\ &= 1 \cdot 15 - 2(1 \cdot 36 - 2 \cdot 15) \\ &= 5 \cdot 15 - 2 \cdot 36. \end{aligned}$$

So we see that the coefficients in the Bezout identity for 15 and 36 are $u = 5$ and $v = -2$ since

$$\gcd(15, 36) = 3 = 5 \cdot 15 - 2 \cdot 36.$$

2.4 Congruences

Definition 8. Let a and n be an integers where n is called the **modulus**. To find the class of a modulo n, denoted as $a \bmod (n)$, divide n into a and take the remainder as your answer.

Example 9. Find 22 mod (10). In this case, $a = 22$ and $n = 10$. Using the division algorithm, we can write 22 divided by 10 as $22 = 2 \cdot 10 + 2$. Since 2 is the remainder, the answer is $22 \bmod (10) = 2$

Definition 10. Let a, b and $n > 0$ be three integers. We define the relation a **is congruent to** b **modulo** n, written as $a \equiv b \bmod (n)$ if and only if $n|(a - b)$.

The relation a is congruent to b is an equivalence relation on the ring of integers. The set of equivalence classes will be denoted as $\mathbb{Z}_n$ and by abuse of notation the class of a (denoted $[a]$) will still be denoted by a. We define addition and multiplication of classes as: $[a] + [b] = [a + b]$ and multiplication $[a] \cdot [b] = [a \cdot b]$. These two operations are well defined and that $\mathbb{Z}_n$ endowed with these two binary operation is a commutative ring with unity. The set of elements of $\mathbb{Z}_n$ with multiplicative inverses is denoted as $\mathbb{Z}_n^*$.

Lemma 11. The integer $x \in \mathbb{Z}_n$ is in $\mathbb{Z}_n^*$ if and only if the $\gcd(x, n) = 1$.

Proof. If the $\gcd(x, n) = 1$ then $1 = ux + vn$. If we look at this equation modulo n we get $1 = ux + v \cdot 0 = ux$. This implies that u is the multiplicative inverse of x. Conversly, if $\gcd(x, n) = d \neq 1$ then $x = dy$ for some y and $n = dm$ for some m with $0 < m < n$. It follows that $x \cdot m = (dy)m = y(dm) = yn = 0 \bmod n$. If x has an inverse u, then $u \cdot 0 = u \cdot (xm) = (ux)m = 1 \cdot m \bmod n$. □

Theorem 12 (Chinese Remainder Theorem)**.** *Let $n_1, n_2, \ldots, n_k$ be positive integers with $\gcd(n_i, n_j) = 1$ whenever $i \neq j$ and let $a_1, a_2, \ldots, a_k$ be any integers. Then there exists a simultaneous solution x to all of the congruences,*

$$\begin{aligned} x &\equiv a_1 \bmod (n_1), \\ x &\equiv a_2 \bmod (n_2), \\ &\vdots \\ x &\equiv a_r \bmod (n_k) \end{aligned}$$

and they form a single congruence class $\bmod(n)$ where $n = n_1 n_2 \ldots n_k$.

Proof. See Theorem 3.10 in [2]. We remark that if $N = \prod_{i=1}^{k} n_i$ then the Chinese Remainder Theorem is equivalent to the fact that the ring $\mathbb{Z}_N$ is isomorphic to the ring $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_k}$ □

Example 13. Find a solution to the congruences,

$$x \equiv 1 \bmod (2),$$
$$x \equiv 2 \bmod (3),$$
$$x \equiv 1 \bmod (5).$$

We see that $\gcd(2,3) = 1$, $\gcd(2,5) = 1$, and $\gcd(3,5) = 1$. Also, $n = 2 \cdot 3 \cdot 5 = 30$, $c_1 = 30/2 = 15$, $c_2 = 30/3 = 10$, and $c_3 = 30/5 = 6$. Now we need to find d_1, d_2, and d_3 satisfying $c_1 d_1 \equiv 1 \bmod (n_1)$, $c_2 d_2 \equiv 1 \bmod (n_2)$, and $c_3 d_3 \equiv 1 \bmod (n_3)$. So $15d_1 \equiv 1 \bmod (2)$ implies that $d_1 \equiv 1 \bmod (2)$ and we may take $d_1 = 1$. Similarly, $d_2 = 1$ and $d_3 = 1$. So a solution is

$$x = a_1 c_1 d_1 + a_2 c_2 d_2 + \ldots + a_k c_k d_k = 1(15)(1) + (2)(10)(1) + 1(6)(1) = 41 \bmod (30) = 11.$$

2.5 Euler Phi Function

Definition 14 (**Euler Phi Function**)**.** Let n be a positive integer. We define $\varphi(n)$ as the number of elements in the set $\mathbb{Z}_n^*$. Equivalently, $\varphi(n)$ is the number of units in $\mathbb{Z}_n$, or $\varphi(n)$ is the number of non-negative integers less than n which are prime to n.

Theorem 15. *Let m and n be two integer such that $m \geq 2$ and $n \geq 2$. Then, $\mathbb{Z}_{mn}$ is isomorphic to $\mathbb{Z}_m \times \mathbb{Z}_n$ if and only if* $\gcd(m,n) = 1$.

Proof. See Theorem 11.5 in [1]. □

Lemma 16. We have the following properties of the function $\varphi(n)$.

1. If p is a prime number and r is a positive integer then $\varphi(p^r) = p^r - p^{r-1}$.

   Proof. See Lemma 5.4 in [2] □

2. If $m > 1$ and $n > 1$ are integers then $\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$.

   Proof. See Corollary 5.7 in [2] □

3 RSA over $\mathbb{Z}$

RSA is a public key encryption system meaning that there is a **public** pair of positive integers which is the encryption key $(\mathcal{E}, n)$ and a **private** pair which is the decryption key $(\mathcal{D}, n)$ as shown below. The encryption key is published for public use, and the decryption key is kept private. This means that there isn't a need for a courier to exchange the keys because anyone can access the key to encrypt. Because of this, public key encryption systems are essential to digital mail privacy. The numbers $\mathcal{E}$ and $\mathcal{D}$ are carefully chosen by the publisher as explained below. The number n is created by taking the product of two random large primes p and q, so $n = p \cdot q$. The number n is made public, but the primes p and q are kept private. The security of RSA depends on the difficulty to factor n.

3.1 Creating the Keys

As mentioned above, RSA uses a public encryption key $(\mathcal{E}, n)$ and a private decryption key $(\mathcal{D}, n)$. To create these keys, we must first randomly choose two large prime numbers p and q (p in the range of 500 digits and q in the range of 600 digits). These are then multiplied together to create the number $n = p \cdot q$. We must calculate $\varphi(n)$. By the Chinese Remainder Theorem (Theorem 12) we know that,

$$\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1).$$

To create the encrypting key, we must find a large random number $\mathcal{E}$ that has $\gcd(\mathcal{E}, \varphi(n)) = 1$ and is between $\max(p, q)$ and $\varphi(n)$. Once we have $\mathcal{E}$, we must find the multiplicative inverse $\mathcal{D}$ which is the key to decrypt the message and satisfies the equation,

$$\mathcal{E} \cdot \mathcal{D} \equiv 1 \bmod (\varphi(n))$$

To find the inverse $\mathcal{D}$, we use the Euclidean Algorithm. Let $a = \varphi(n)$ and $b = \mathcal{E}$ so that,

$$\begin{aligned} a &= q_1(b) + r_1 \\ b &= q_2(r_1) + r_2 \\ r_1 &= q_3(r_2) + r_3 \\ &\vdots \\ r_{k-2} &= q_k(r_{k-1}) + r_k \end{aligned}$$

Perform these steps until the remainder becomes $r_k = 1$ (we know this will be the case because $\gcd(\mathcal{E}, \varphi(n)) = 1$). Now we can preform the Extended Euclidean Algorithm to find numbers that satisfy Bezout's identity $\gcd(\mathcal{E}, \varphi(n)) = ua + vb$. By substitution, $1 = u\varphi(n) + v\mathcal{E}$ which can be rewritten as $v \cdot \mathcal{E} \equiv 1 \bmod (\varphi(n))$. So we see that our inverse number $\mathcal{D} = v$.

3.2 Encryption

To encrypt a message, user A (the sender) first takes their plaintext message M and converts it into decimal format (the number representation of the message) using a prearranged algorithm with user B (the receiver) which becomes m. To encrypt the message m, user A uses the published encryption or enciphering key $(\mathcal{E}, n)$ to raise m to the $\mathcal{E}^{\text{th}}$ power modulo n as seen in the following modulus function,

$$\mathcal{E}(m) = m^{\mathcal{E}} \bmod (n) = c.$$

This gives us the ciphertext (which is encrypted message) c. The final step is to convert the ciphertext from decimal c into binary and send to user B.

3.3 Decryption

The decryption process is very similar to the encryption process except user B (the receiver) uses the decryption or deciphering key $(\mathcal{D}, n)$ which has been kept private. The received message c is raised to the $\mathcal{D}^{\text{th}}$ power modulo n instead of the $\mathcal{E}^{\text{th}}$ power as seen bellow,

$$\mathcal{D}(c) = c^{\mathcal{D}} \bmod (n) = m$$

First, user B converts the received binary code into decimal format and preforms $\mathcal{D}(c)$ to find the original message m. Next, user B converts $\mathcal{D}(c) = m$ back into the message M.

3.4 Example

Assume Jill wants to send a message $M = YES$ through the RSA encryption process to Paul. They have pre-arranged that they will use an alphabet of 26 letters, which are each represented by a number 0-25. Paul has made the enciphering key ($n = 46927$, $\mathcal{E} = 39423$) public while he keeps the numbers $p = 281$ and $q = 167$ private along with the decryption key $\mathcal{D} = 26767$ (which is the multiplicative inverse of $\mathcal{E}$ mod $(\varphi(n))$). Jill knows that $Y = 24$, $E = 4$ and $S = 18$. She writes the message YES in decimal format by

$$24 \cdot 26^2 + 4 \cdot 26 + 18 = 16346 = m$$

Jill then encrypts the message m by using the encryption formula as seen below,

$$\mathcal{E}(m) = m^{\mathcal{E}} \bmod (n) = 16346^{39423} \bmod (46927) = 21166 = c,$$

The final step for Jill is to change $c = 21166$ back to base 26 from the current decimal format by dividing 26 into c and taking the remainders.

$$\begin{aligned} 21166 &= 814(26) + 2 \\ 814 &= 31(26) + 8 \\ 31 &= 1(26) + 5 \\ 1 &= 0(26) + 1 \end{aligned}$$

The remainders $1 = B$, $5 = F$, $8 = I$ and $2 = C$ give the coded message $C = BFIC$ which she sends to Paul.

Paul receives the message $C = BFIC$ and follows the same steps Jill followed to decrypt it only with a slight difference. The difference is that Paul will use the decryption key $\mathcal{D} = 26767$ instead of the encryption key $\mathcal{E}$. So, $C = BFIC$ becomes $c = 21166$ and,

$$\mathcal{D}(c) = c^{\mathcal{D}} \bmod (n) = 21166^{26767} \bmod (46927) = 16346 = m,$$

Finally, Paul converts m to the original base 26 by dividing by 26 and taking the remainders as the digits. So,

$$\begin{aligned} 16346 &= 628(26) + 18 \\ 627 &= 24(26) + 4 \\ 24 &= 0(26) + 24 \end{aligned}$$

The remainders are $24, 4, 18$ which Paul knows represent the letters $Y = 24$, $E = 4$ and $S = 18$. So he uncovers the original message $M = YES$.

Note: Keep in mind that this example uses prime numbers that are three digits each. When RSA is used, the primes are hundreds of digits long.

This example is taken from [3]

3.5 Strength of RSA

The strength of RSA lies in the prime factorization. Recall that the numbers being made public are n and $\mathcal{E}$. The number n is an extremely large number (large meaning 500-600 digits each) since n is the product of two large primes, and $\mathcal{E}$ is rather large itself since $\mathcal{E} > n$. In order to decrypt a message, we must find the inverse $\mathcal{D}$ of $\mathcal{E}$ mod $(\varphi(n))$. This is only possible by first finding the value $\varphi(n)$. To find $\varphi(n)$, we must be able to factor n because $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$. Once $\varphi(n)$ is found, we can then use the Euclidean Algorithm to find the desired number $\mathcal{D}$. This prime factorization is extremely difficult because $n = p \cdot q$ is such a large number.

4 Properties of the Ring of Polynomials over a Field $\mathbb{F}$

We have now looked at RSA encryption over the ring of integers, and we want look at the implementation process of RSA over a ring of polynomials. This section examines the different necessary definitions and theorems for understanding in RSA over polynomials.

4.1 Division Algorithm for $\mathbb{F}[x]$

Definition 17. Let $a(x)$ and $b(x)$ be two polynomials with degree greater than zero. We say that $b(x)$ **divides** $a(x)$ (written $b(x)|a(x)$) if and only if there exists a polynomial $c(x)$ such that $a(x) = b(x) \cdot c(x)$.

Definition 18. Let $a(x)$ and $b(x)$ be two polynomials with degree greater than zero. The **greatest common divisor** of $a(x)$ and $b(x)$, denoted $\gcd(a(x), b(x))$, is the polynomial with the greatest degree which divides both $a(x)$ and $b(x)$.

Definition 19. Let $a(x) = a_0 + a_1x + a_2x^2 + ... + a_dx^d$ be a polynomial. The **degree** d of $a(x)$ is the highest power of x with nonzero coefficients, denoted as $\deg(a(x)) = d > 0$.

Theorem 20. *Let $a(x)$ and $b(x)$ be two polynomials with degree greater than zero. There exists unique polynomials $q(x)$ and $r(x)$ in $\mathbb{F}[x]$ such that $a(x) = b(x) \cdot q(x) + r(x)$ with $r(x) = 0$ or the degree of $r(x)$ is less than the degree of $b(x)$.*

Proof. Theorem 23.1 in [1] □

Lemma 21. Let $a(x)$ and $b(x)$ be two polynomials with degree greater than zero. Assume that $a(x) = b(x)q(x) + r(x)$ as in the division algorithm. Then $\gcd(a(x), b(x)) = \gcd(b(x), r(x))$.

Proof. Imitate proof for integers as in Lemma 1.5 in [2]. □

4.2 Euclidean Algorithm

To find the greatest common divisor $d(x)$ of polynomials $a(x)$ and $b(x)$ when $deg(a(x)) > deg(b(x))$, use Theorem 20 to divide $b(x)$ into $a(x)$ as follows,

$$a(x) = q_1(x)b(x) + r_1(x) \quad \text{with} \quad \deg(r_1(x)) = 0 \quad \text{or} \quad \deg(r_1(x)) < \deg(b(x)).$$

If $r_1(x) = 0$ then $b(x)|a(x)$ and $d(x) = b(x)$. If not, then we divide $r_1(x)$ into $b(x)$ and write,

$$b(x) = q_2(x)r_1(x) + r_2(x) \quad \text{with} \quad \deg(r_1(x)) = 0 \quad \text{or} \quad \deg(r_2(x)) < \deg(r_1(x)).$$

Since $\gcd(a(x), b(x)) = \gcd(b(x), r_1(x))$ by Theorem 21, if $r_2(x) = 0$ then $d(x) = r_1(x)$. If not, then we continue in this way until the remainder equals 0.

$$\begin{aligned} a(x) &= q_1(x)b(x) + r_1(x) \\ b(x) &= q_2(x)r_1(x) + r_2(x) \\ r_1(x) &= q_3(x)r_2(x) + r_3(x) \\ &\vdots \\ r_{k-2}(x) &= q_k(x)r_{k-1}(x) + r_k(x) \end{aligned}$$

Finally, when $r_k(x) = 0$, $d(x) = \gcd(a(x), b(x)) = r_{k-1}(x)$.

We can use the Euclidean Algorithm and Theorem 20 to find the greatest common divisor of any two polynomials.

Example 22. Let us find the greatest common divisor of the polynomials $a(x) = x^4 - 4x^3 + 6x^2 - 4x + 1$ and $b(x) = x^3 - x^2 + x - 1$. Similarly to the Euclidean Algorithm for integers, we use the Euclidean Algorithm for polynomials as follows:

$$\begin{aligned} a(x) &= (x-3)b(x) + (2x^2 - 2) \\ b(x) &= (2x^2 - 2)(\frac{1}{2}x - \frac{1}{2}) + (2x - 2) \\ (2x^2 - 2) &= (2x - 2)(x + 1) + 0 \\ (2x - 2) &= \gcd(a(x), b(x)) \end{aligned}$$

It is customary to represent this polynomial using a monic polynomial. A monic polynomial is defined to be the polynomial where the coefficient of the term with the highest degree is equal to 1. So the monic polynomial for $(2x - 2) = (x - 1)$. Therefore, the $\gcd(a(x), b(x)) = (x - 1)$.

4.3 Bezout's Identity and The Extended Euclidean Algorithm

Theorem 23 (Bezout's Identity)**.** *Let $a(x)$ and $b(x)$ be two polynomials with degree greater than zero. There exists polynomials $u(x)$ and $v(x)$ such that* $\gcd(a(x), b(x)) = u(x)a(x) + v(x)b(x)$.

Proof. Imitate proof for integers as in Theorem 1.7 in [2]. □

In order to find the Bezout polynomials $u(x)$ and $v(x)$, we use the Extended Euclidean Algorithm. Start with the $\gcd(a(x), b(x))) = r_{k-1}(x)$, which was the last non-zero remainder found by preforming the Euclidean Algorithm. Solve for $r_{k-1}(x)$ in the second to last equation to get

$$r_{k-1}(x) = r_{k-3}(x) - q_{k-1}(x)r_{k-2}(x)$$

which expresses the gcd as a multiple of $r_{k-3}(x)$ and $r_{k-2}(x)$. Next, use the previous equation to solve for $r_{k-2}(x)$,

$$r_{k-2}(x) = r_{k-4}(x) - q_{k-2}(x)r_{k-3}(x)$$

Eliminate $r_{k-2}(x)$ by substituting the second equation into the first to express the gcd as a multiple of $r_{k-3}(x)$ and $r_{k-4}(x)$. Repeat this until all of the remainders $r_{k-3}(x), r_{k-4}(x), ...$ are eliminated. This results in an equation that expresses the gcd as a linear combination with polynomial coefficients of $a(x)$ and $b(x)$, or more specifically, gcd $= u(x)a(x) + v(x)b(x)$.

Example 24. Find the Bezout polynomials for $a(x) = x^4 - 4x^3 + 6x^2 - 4x + 1$ and $b(x) = x^3 - x^2 + x - 1$. Returning to our example from above where the $\gcd(a(x), b(x)) = (2x - 2)$, we have

$$\begin{aligned} (2x - 2) &= b(x) - (\frac{1}{2}x - \frac{1}{2})(2x^2 - 2) \\ &= b(x) - (\frac{1}{2}x - \frac{1}{2})(a(x) - b(x)(x - 3)) \\ &= (\frac{1}{2}x^2 - 2x + \frac{5}{2})b(x) - (\frac{1}{2}x - \frac{1}{2})a(x) \end{aligned}$$

So we see that the coefficient polynomials in the Bezout identity for $a(x)$ and $b(x)$ are $u(x) = -(\frac{1}{2}x - \frac{1}{2})$ and $v(x) = (\frac{1}{2}x^2 - 2x + \frac{5}{2})$.

4.4 Congruences

Definition 25. Let $a(x)$ and $n(x)$ be any polynomials with degree greater than zero where $n(x)$ is called the **modulus**. To find the class of $a(x)$ modulo $n(x)$, denoted as $a(x) \bmod (n(x))$, divide $n(x)$ into $a(x)$ and take the remainder as the class representative.

Example 26. Find $(x^3+2x^2+3) \bmod (x^2-1)$. In this case, $a(x) = x^3+2x^2+3$ and $n(x) = x^2-1$. Using the division algorithm, we can write $a(x)$ divided by $n(x)$ as $a(x) = (x+2)n(x) + (3x+1)$. Since $(3x+1)$ is the remainder, the class representative is $(3x+1)$

Definition 27. Let $a(x)$, $b(x)$ and $n(x)$ be three polynomials with degree greater than zero. We define the relation $a(x)$ **is congruent to** $b(x)$ **modulo** $n(x)$, written as $a(x) \equiv b(x) \bmod (n(x))$ if and only if $n(x)$ divides $a(x) - b(x)$ and we will write $n(x)|(a(x) - b(x))$.

The relation $a(x)$ is congruent to $b(x)$ is an equivalence relation on the ring of polynomials. The set of equivalence classes will be denoted as $\mathbb{F}[x]/n(x)$ and the class of $a(x)$ will still be denoted by $[a(x)]$. We define addition and multiplication of classes as: $[a(x)] + [b(x)] = [a(x) + b(x)]$ and multiplication $[a(x)] \cdot [b(x)] = [a(x) \cdot b(x)]$. It is not hard to check that these two operations are well defined and that $\mathbb{F}[x]/n(x)$ endowed with these two binary operation is a commutative ring with unity. The set of elements of $\mathbb{F}[x]/n(x)$ that have multiplicative inverse is denoted as $[\mathbb{F}[x]/n(x)]^\star$. By abuse of notation we will not use the brackets to represent classes in further explanations.

Lemma 28. The class of the polynomial $a(x)$ in $\mathbb{F}[x]/n(x)$ is in $[\mathbb{F}[x]/n(x)]^*$ if and only if $\gcd(a(x), n(x)) = 1$.

Proof. This follows from Bezout's Identity. □

Theorem 29 (Chinese Remainder Theorem)**.** *Let $n_1(x), n_2(x), \ldots, n_k(x)$ be polynomials such that $\gcd(n_i(x), n_j(x)) = 1$ whenever $i \neq j$ and let $a_1(x), a_2(x), \ldots, a_k(x)$ be any polynomials. Then there exists a simultaneous solution $m(x)$ to all of the congruences,*

$$\begin{aligned} m(x) &\equiv a_1(x) \bmod (n_1(x)), \\ m(x) &\equiv a_2(x) \bmod (n_2(x)), \\ &\vdots \\ m(x) &\equiv a_r(x) \bmod (n_k(x)) \end{aligned}$$

and they form a single congruence class $\bmod(n(x))$ where $n(x) = n_1(x)n_2(x)\ldots n_k(x)$. As in the case of the ring of integers the Chinese Remainder Theorem is equivalent to the fact that the ring $\mathbb{F}[x]/n(x)$ is isomorphic to the ring

$$\mathbb{F}[x]/n_1(x) \times \mathbb{F}[x]/n_2(x) \times \cdots \times \mathbb{F}[x]/n_k(x)$$

Proof. Imitate the proof given for the ring of integers as in Theorem 3.10 in [2]. □

Definition 30. Let $f(x)$ be a non-constant polynomial in $\mathbb{F}[x]$ with degree greater than zero. We sat that $f(x)$ is **irreducible** over $\mathbb{F}$ or is an **irreducible polynomial** in $\mathbb{F}[x]$ if $f(x)$ cannot be expressed as a product of two polynomials $g(x)$ and $h(x)$ each of degree greater than one.

4.5 Euler Phi Function for Polynomials

Definition 31. Let p be a prime number, $\mathbb{F}_k$ be a finite field with $k = p^r$ elements. Let $\mathbb{F}_k[x]$ be the ring of polynomials with coefficients in $\mathbb{F}_k$, and let $f(x) = f_0 + f_1x + ... + f_dx^d$ be a polynomial in $\mathbb{F}_k[x]$ where $\deg(f(x)) = d$. Consider the quotient ring $R_f = \mathbb{F}_k[x]/f(x)$. We define $\varphi(f(x))$ to be the number of elements in R_f^* where

$$R_f^* = \{a(x) \in R \,|\, a(x) \text{ has a multiplicative inverse in } R\}.$$

It follows from Theorem 23 that $a(x) \in R_f^*$ if and only if $\gcd(a(x), f(x)) = 1$.

Theorem 32. *Let $\mathbb{F}$ be a field (not necessarily finite) and $\mathbb{F}[x]$ the ring of polynomials over $\mathbb{F}$. Let $m(x)$ and $n(x)$ be two polynomials such that $\deg(m(x)) \geq 2$ and $\deg(n(x)) \geq 2$. Then, $R_{m(x)n(x)}$ is isomorphic to $R_{m(x)} \times R_{n(x)}$ if and only if $\gcd(m(x), n(x)) = 1$.*

Proof. Imitate proof for integers as in Theorem 11.5 in [1]. □

Lemma 33. Let $f(x)$ be any irreducible polynomial of degree d in $\mathbb{F}_k[x]$. Then,

$$\varphi(f(x)^n) = k^{dn} - k^{d(n-1)}$$

Proof. By abuse of notation, f^n will be used to represent $f(x)^n$. To find the cardinality (or number of elements) of $R_{f^n}^*$, we are going to find the cardinality of $R_{f^n} \setminus R_{f^n}^*$ and subtract that from the cardinality of the set R_{f^n}. First observed that $b(x) \in R_{f^n} \setminus R_{f^n}^*$ if and only $\gcd(b(x), f(x)^n) \neq 1$. Since $f(x)$ is irreducible $f(x)$ must divide $b(x)$. Therefore, $b(x)$ must be of the form $b(x) = c(x)f(x)$ for some $c(x) = c_0 + c_1d + ... + c_sx^s \in \mathbb{F}[x]$ where s must satisfy

$$\begin{aligned} s + d &< nd \text{ or} \\ s &< nd - d = d(n-1). \end{aligned}$$

So there are $d(n-1)$ possible coefficients for the polynomial $c(x)$ because its coefficients need to stop at $c_{d(n-1)-1}$. Therefore,

$$\left|R_{f^n} \setminus R_{f^n}^*\right| = k^{d(n-1)}$$

We know that $|R_{f^n}| = (k^d)^n = k^{dn}$ because any element in R_{f^n} has at the most degree $dn - 1$. Since $R_{f^n} = \left[R_{f^n} \setminus R_{f^n}^*\right] \cup R_{f^n}^*$ and $\left[(R_{f^n} \setminus R_{f^n}^*)\right] \cap R_{f^n}^* = \varnothing$, that is R_{f^n} and $R_{f^n} - R_{f^n}^*$ do not intersect, we have

$$\begin{aligned} \left|R_{f^n}\right| &= \left|(R_{f^n} \setminus R_{f^n}^*)\right| + \left|R_{f^n}^*\right| \\ \varphi(f(x)^n) = \left|R_{f^n}^*\right| &= \left|R_{f^n}\right| - \left|(R_{f^n} \setminus R_{f^n}^*)\right| = k^{dn} - k^{d(n-1)} \end{aligned}$$

□

Theorem 34. *If $deg(m(x)) > 1$ and $deg(n(x)) > 1$ are polynomials then*

$$\varphi(m(x) \cdot n(x)) = \varphi(m(x)) \cdot \varphi(n(x)).$$

Proof. The proof follows from Lemma 33 and the Chines Remainder Theorem. □

5 RSA over $\mathbb{F}_k[x]$

Now that we have explored how RSA works with messages represented as integers and the mathematics about integers and polynomials, let us examine how RSA works over the ring of polynomials. From now on we will be using $k = 2$ when talking about the field $\mathbb{F}_2[x]$.

5.1 Creating the Keys

In RSA with integers we need to select two random prime numbers. However, when using RSA with polynomials, instead of prime numbers we find two irreducible polynomials $p(x)$ with $\deg(p(x)) = d_p$ and $q(x)$ with $\deg(q(x)) = d_q$ in the ring $\mathbb{F}_2[x]$ and we set $n(x) = p(x)q(x)$. We select the encryption key $\mathcal{E}$, to be a random integer satisfying $\gcd(\mathcal{E}, \varphi(n(x))) = 1$ and that

$$\max(p(x), q(x)) < \mathcal{E} < \varphi(n(x)).$$

In order to find the value for $\varphi(n(x))$, which we know that to be, $\varphi(n(x)) = \varphi(p(x))\varphi(q(x))$ we need to define the value for $\varphi(p(x))$.

By theorem 33 we see that for the irreducible polynomials $p(x)$ and $q(x)$,

$$\varphi(p(x)) = 2^{d_p \cdot (1)} - 2^{d_p \cdot (1-1)} = 2^{d_p} - 2^0 = 2^{d_p} - 1$$
$$\varphi(q(x)) = 2^{d_q \cdot (1)} - 2^{d_q \cdot (1-1)} = 2^{d_q} - 2^0 = 2^{d_q} - 1$$
$$\varphi(n(x)) = \varphi(p(x) \cdot q(x)) = \varphi(p(x)) \cdot \varphi(q(x)) = (2^{d_p} - 1) \cdot (2^{d_q} - 1).$$

Now that we have calculated the value of $\varphi(n(x))$, we must find the encryption key $\mathcal{E}$ and the decryption key $\mathcal{D}$. These are found using the same method from when we were using the integers. Choose a random number $\mathcal{E}$ and find $\mathcal{D}$ (which is the multiplicative inverse) by using the Euclidean Algorithm which satisfies the equation

$$\mathcal{E} \cdot \mathcal{D} = 1 \bmod (\varphi(n(x)))$$

5.2 Encryption

The steps are very similar for encrypting as explained above in regards to integers, however there is one step added. User A (the sender) takes their plaintext message M and converts it all the way into a binary message m. They then create a polynomial $m(x)$ from the binary number. Each digit of message m is the coefficient to the corresponding term. For example, let $m = 101011$. The corresponding polynomial would be $m(x) = 1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0 = x^5 + x^3 + x + 1$. To encrypt this polynomial we just raise $m(x)$ to the $\mathcal{E}^{\text{th}}$ power like before as seen below,

$$\mathcal{E}(m(x)) = m(x)^{\mathcal{E}} \bmod (n(x)) = c(x).$$

The coefficients are extracted from $c(x)$ to create the binary number or ciphertext c which is finally sent to user B (the receiver).

5.3 Decryption

The binary ciphertext c is received by user B who creates the corresponding polynomial $c(x)$, and raises it to the $\mathcal{D}^{\text{th}}$ power as seen below,

$$\mathcal{D}(c(x)) = c(x)^{\mathcal{D}} \bmod (n(x)) = m(x),$$

User B takes $m(x)$ and extracts the coefficients to find the binary digits m and converts m to the original message M.

5.4 Example

Let us look at an example of RSA using the polynomial ring $\mathbb{F}_2[x]/n(x)$. Because these numbers and polynomials quickly become to large to calculate by hand, the free online software CoCalc has been utilized to write code which performs the encryption and decryption [5]. We have included the lines of code in this example with explanations of the action each line preforms.

`R = PolynomialRing(GF(2),'x')` This line of code creates the polynomial ring.
`x = R.gen()` This line of code shows that x is the variable for the polynomials.
`p`=$x^{101} + x^7 + x^6 + x + 1$ Here we are defining our first irreducible polynomial.
`q`=$x^{503} + x^3 + 1$ Here we are defining our second irreducible polynomial.
`n=p*q` This line of code creates our polynomial $n(x)$ which is,

$$n(x) = x^{604} + x^{510} + x^{509} + x^{504} + x^{503} + x^{104} + x^{101} + x^{10} + x^9 + x^7 + x^6 + x^4 + x^3 + x + 1.$$

`S = R.quotient(n, 'a')` This line of code creates the quotient ring.
`a = S.gen()` Now a will be the variable for the quotient ring which are polynomials in a.
`e=71` This is the encryption key $\mathcal{E} = 71$.
`phiOfN`=$(2^{503} - 1) * (2^{101} - 1)$ This calculates the $\varphi(n(x))$ which is the following number,
663922491020958873361985258190323912913359705809188848823558964818327118462381858240
794464522088916382399287893417464661797151839493813114531695167777471456394113692511
94530751840257.

Now we have implemented the needed code to preform the encryption. Jan wants to send the message $M = HELP$ to Norm. Both users have pre-arranged that any message sent must be converted to binary where each binary digit represent the coefficients of the message polynomial $m(x)$. So Jan converts the message M to binary,

$$M = HELP = (126037)_{26} = (11110110001010101)_2.$$

From this binary number, she then creates the corresponding message polynomial,

$$m(x) = a^{16} + a^{15} + a^{14} + a^{13} + a^{11} + a^{10} + a^6 + a^4 + a^2 + 1.$$

In the code we assign this polynomial to the variable `M` and raise the message to the encryption key $\mathcal{E} = 71$ which produces the encrypted polynomial `eM` as below.
`eM=M`$^e = a^{601}+a^{600}+a^{598}+a^{597}+a^{596}+a^{594}+a^{593}+a^{591}+a^{590}+a^{587}+a^{586}+a^{584}+a^{583}+a^{582}+a^{581}+$
$a^{580}+a^{578}+a^{577}+a^{576}+a^{573}+a^{571}+a^{570}+a^{568}+a^{567}+a^{565}+a^{560}+a^{559}+a^{557}+a^{555}+a^{554}+a^{553}+$
$a^{551}+a^{550}+a^{547}+a^{546}+a^{545}+a^{544}+a^{543}+a^{541}+a^{540}+a^{536}+a^{535}+a^{534}+a^{533}+a^{530}+a^{526}+a^{523}+$
$a^{519}+a^{517}+a^{516}+a^{513}+a^{511}+a^{509}+a^{507}+a^{506}+a^{505}+a^{504}+a^{502}+a^{501}+a^{499}+a^{498}+a^{497}+a^{496}+$
$a^{494}+a^{492}+a^{491}+a^{488}+a^{487}+a^{486}+a^{485}+a^{484}+a^{479}+a^{478}+a^{473}+a^{471}+a^{469}+a^{467}+a^{464}+a^{462}+a^{461}+$
$a^{457}+a^{455}+a^{454}+a^{452}+a^{451}+a^{450}+a^{446}+a^{444}+a^{442}+a^{440}+a^{439}+a^{434}+a^{433}+a^{432}+a^{431}+a^{430}+a^{429}+$
$a^{427}+a^{424}+a^{417}+a^{414}+a^{413}+a^{412}+a^{411}+a^{407}+a^{406}+a^{403}+a^{401}+a^{399}+a^{398}+a^{396}+a^{395}+a^{392}+a^{391}+$
$a^{390}+a^{389}+a^{386}+a^{382}+a^{373}+a^{372}+a^{369}+a^{368}+a^{365}+a^{364}+a^{363}+a^{362}+a^{361}+a^{358}+a^{357}+a^{355}+a^{353}+$
$a^{351}+a^{350}+a^{348}+a^{345}+a^{343}+a^{341}+a^{339}+a^{338}+a^{337}+a^{335}+a^{334}+a^{331}+a^{325}+a^{324}+a^{323}+a^{322}+a^{320}+$
$a^{316}+a^{314}+a^{312}+a^{311}+a^{309}+a^{308}+a^{306}+a^{304}+a^{302}+a^{294}+a^{291}+a^{289}+a^{287}+a^{284}+a^{282}+a^{275}+a^{271}+$
$a^{265}+a^{264}+a^{263}+a^{262}+a^{261}+a^{258}+a^{254}+a^{252}+a^{251}+a^{249}+a^{246}+a^{245}+a^{243}+a^{242}+a^{241}+a^{240}+a^{238}+$
$a^{236}+a^{235}+a^{232}+a^{231}+a^{230}+a^{229}+a^{228}+a^{223}+a^{222}+a^{217}+a^{215}+a^{213}+a^{211}+a^{208}+a^{206}+a^{205}+a^{201}+$
$a^{199}+a^{198}+a^{196}+a^{195}+a^{194}+a^{190}+a^{187}+a^{186}+a^{185}+a^{184}+a^{183}+a^{182}+a^{180}+a^{179}+a^{177}+a^{176}+a^{175}+$

$a^{171}+a^{170}+a^{169}+a^{167}+a^{164}+a^{163}+a^{158}+a^{157}+a^{155}+a^{151}+a^{150}+a^{147}+a^{146}+a^{144}+a^{142}+a^{141}+a^{140}+$
$a^{137}+a^{135}+a^{132}+a^{130}+a^{126}+a^{125}+a^{124}+a^{123}+a^{122}+a^{121}+a^{119}+a^{117}+a^{116}+a^{114}+a^{113}+a^{112}+a^{105}+$
$a^{101}+a^{100}+a^{97}+a^{96}+a^{95}+a^{94}+a^{91}+a^{89}+a^{87}+a^{86}+a^{84}+a^{81}+a^{80}+a^{79}+a^{78}+a^{76}+a^{75}+a^{72}+a^{71}+$
$a^{69}+a^{67}+a^{66}+a^{65}+a^{62}+a^{61}+a^{58}+a^{57}+a^{56}+a^{53}+a^{51}+a^{47}+a^{46}+a^{43}+a^{40}+a^{39}+a^{36}+a^{35}+a^{31}+$
$a^{30}+a^{28}+a^{27}+a^{26}+a^{23}+a^{22}+a^{21}+a^{20}+a^{19}+a^{17}+a^{15}+a^{13}+a^{12}+a^{8}+a^{7}+a^{5}+a^{4}+a^{3}+a^{2}+a+1$

Jan extracts the coefficients from this encrypted polynomial `eM` creating the binary message,
1101110110110011011111011100101101101000011010111011001111101100011110010001001000101
1001010101111011011110101100111111000011000010101010010110001011011100010101011000011 1
1101001000000100111100011001010110110011111001000100000000110011001111100110101011010
0101010111011001000001111010001010110110101010000000100101010010100000010001000001111
1001000101101001101111010110011111000011000010101010010110001011011100010011111101101
1100011101001100001101000110011010111001010010100011111101011011100000010001100111100
1010110100111101100110101110011001110010100011001001100110001101110011111010101100011
0111111 which she sends to Norm. Norm receives this binary number and uses the decryption key to retrieve the original message. To find the decryption key $\mathcal{D}$ (which is the multiplicative inverse of $\mathcal{E} = 71$) we find the Bezout number v as seen in the below line of code.

```
d,u,v=xgcd(fiOfN,e)
```

In this case, $\mathcal{D} = 29923267200944625278286659524070936920038747304076117130075897005896433508163689385500595584094148343399546086280191390025001247727296059107748485263222406564029808191603133780364631 3$. Norm raises the received message `eM` to the $\mathcal{D}^{th}$ power which results in the original message polynomial.

`dM=eM`^{d}

$$\begin{aligned} m(x) &= a^{16}+a^{15}+a^{14}+a^{13}+a^{11}+a^{10}+a^{6}+a^{4}+a^{2}+1 \\ &= 11110110001010101 \\ &= 126037 \\ &= HELP \end{aligned}$$

5.5 Strength of RSA

As we saw above, the strength of RSA lies in the difficulty for an outside source to factor the number n. In the case of this polynomial ring, it seems that to factor the polynomial $n(x)$ is also difficult since it is the product of two unique irreducible polynomials both with large degree (at least 500 or 600 digits).

6 Future Work

It is important to note that RSA could be implemented using any other finite field. For example, we could use $\mathbb{F}_5[x]$ instead of $\mathbb{F}_2[x]$. However, one must then find irreducible polynomials of high degree in this finite field, a difficult problem we have not discussed here. Another area for future study would be the difficulty to factor the polynomial $n(x)$ compared with the difficulty to factor the integer n. This would tell us how secure polynomial RSA is compared to the classical RSA over the integers.

7 Conclusion

In this paper we explored the implementation of RSA over the ring of integers $\mathbb{Z}$ and extended the RSA algorithm to the ring of polynomials over the field $\mathbb{F}_2[x]$. In the case of integers we pick random prime numbers p and q in the range of (500 to 600 digits each) with $p < q$ and create the modulus $n = pq$. In the case of polynomials we pick two irreducible polynomials $p(x)$ and $q(x)$ with degree in the range (500 to 600 digits each) with $\deg(p(x)) < \deg(q(x))$ and create the modulus polynomial $n(x) = p(x)q(x)$. In both cases we select a random prime number $\mathcal{E}$ (with requirements listed in section 3.2 and 5.2) to be the encryption key. Next, we find the decryption key $\mathcal{D}$ which is the multiplicative inverse of $\mathcal{E}$ modulo $\varphi(n)$ or $\varphi(n(x))$ (see sections 3.1 and 5.1 for explanation of finding $\mathcal{D}$). In section 4.5, we defined and found a formula to calculate $\varphi(n(x))$ of a given polynomial $n(x)$ over a finite field $\mathbb{F}_k[x]$, Lemma 33. The implementation of polynomial RSA hinges on the formula for $\varphi(n(x))$.

References

[1] Fraleigh, John B., *A First Course in Abstract Algebra*, Pearson Education. Inc, 2003.

[2] Jones, Gareth A., and Jones, J. Mary, *Elementary Number Theory*, Springer Undergraduate Mathematics Series, 1998.

[3] Koblitz, Neal, *A Course in Number Theory and Cryptography*, Department of Mathematics University of Washington, 1994.

[4] Rivest R. L., Shamir A., Adleman L., *A Method for obtaining Digital Signatures and Public-Key Cryptosystems*, 1977.

[5] Stein, William A. et al. Sage Mathematics Software (Version 8.1). The Sage Development Team, 2011. http://www. sagemath.org