Linnæus University
Sweden

Bachelor Thesis

Polynomial based RSA

Author: Izabela Beatrice Gafitoiu
Supervisor: Per-Anders Svensson
Examiner: Marcus Nilsson
Date: 2015-06-10
Course Code: 2MA11E
Subject: Mathematics
Level: Bachelor

Department Of Mathematics

Abstract

The RSA public-key cryptosystem has a major role in information security even today, after more than three decades since it was invented. The reason why is that the security provided by this algorithm relies on the fact that integer factorization is considered to be a hard problem. The day someone finds an efficient algorithm to factor integers, is the day when the RSA cryptosystem, used millions of times every day, will not be secure anymore. A good perspective is to look ahead of that day, and start thinking about alternatives. A polynomial version of the RSA cryptosystem is one such idea someone can think about. The idea will be analyzed from three main points of view, namely whether it is easy to encrypt messages, whether it is correct, that is whether the private key will always restore the plaintext, and the last point to be analyzed is the degree of difficulty of breaking the code, or in other words how secure the algorithm is. The original RSA cryptosystem and the polynomial version will be discussed in parallel all the time.

It will result that the mathematical operations used in the encryption and decryption processes become more complex in the case of the polynomial RSA, and the problem behind its security has been well and successfully studied along the years.

That being said, it will follow that the polynomial version of the RSA public-key cryptosystem is not a good alternative if one day someone succeeds breaking the original RSA.

Contents

Chapter 1

Introduction

Keeping information private while communicating over the internet and digital data secure are major concerns in the virtual world. Personal information that goes into the wrong hands might have devastating effects. The art and science of secure transmission of confidential information defines cryptography. Encryption is the process of transforming information so that it is unreadable to anyone but the intended receiver, whereas decryption is the process of restoring the message that was sent. Asymmetric cryptography, or public-key cryptography, defines a class of cryptographic algorithms which have two kind of keys, a public key, used for encryption, and a private key, used for decryption. An algorithm having a public key implies that everybody can have access to it, everybody can encrypt messages using that key, while only the holder of the private key can decrypt messages. A real-world analogy to public-key cryptography is a padlock. Everyone can easily close a padlock, while doing the opposite becomes a challenge if you do not have the right key that opens the padlock. Imagine that Alice sends an open padlock to Bob. Bob writes a message, puts it in a box, and then he locks the box with the padlock received from Alice and sends the box to her. It does not matter through how many hands the box will travel, no one but Alice can open it since she has the key to the padlock.

Such kind of algorithm ensures nowadays the authenticity and privacy of the email and the security of electronic credit-card payment systems.

Back in 1977, Ronald Rivest, Adi Shamir, and Leonard Adleman invented what we call today the RSA Public-Key Cryptosystem. The latter authors published their work in 1978 [26]. The public key in this cryptosystem consists of the value n, which is called the modulus, and the value e, which is called the public exponent. The private key consists of the two prime factors of n, p and q, and the value d, which is called the private exponent. In other words the pair (n, e) denotes the public key used for encryption, and being available to everyone, and the triple (p, q, d) denotes the private key, used for decryption, and being available only to the intended receiver. More than thirty years since its invention, the cryptosystem is used millions of times every day, passing thus, the test of time.

What is therefore the reason behind its longevity? Not difficult to imagine, the combination between the easiness to encrypt messages and the difficulty to break them has made this coding method so popular and reliable.

At the heart of RSA's security is the mathematical problem of integer factorization. Despite hundreds of years of study, nobody has yet discovered an efficient algorithm to find the prime factors of a presumably large integer. What is interesting is that a fast method for integer factorization exists actually in theory, but it runs on computers that have not yet been built [30]. On such a computer the task of integer factorization would be as easy as integer multiplication.

An important area in cryptography is the alternatives one has, in case one of the current public-key cryptosystems is broken in the future. What if one day someone finds an efficient algorithm for integer factorization? We have to find another solution quickly, another way to maintain the security of our private information, which is so important for us.

This is therefore the reason why, in this report we will take a look at the polynomial version of the RSA public-key cryptosystem. In the second chapter we will firstly explain how the original algorithm works, as well as the polynomial version of it. Proceeding to the third chapter, we will see why both versions of the same idea work for encrypting messages and successfully decrypting them, while in the fourth chapter we will analyze the theory behind the cryptosystem's security. In particular, algorithms

for integer factorization as well as for polynomial factorization will be discussed. In the fifth chapter we will find a comparison between the original RSA and the polynomial RSA, displaying key strengths of them both and raising the conclusion on whether the version of RSA, proposed in this report, meets the requirements to replace the original version of RSA. In the sixth chapter an interactive computer application will be presented. The application allows any user to build up values for the polynomial RSA, testing encryption and decryption for an input of his choice. How to use the application as well as examples of an execution of it, will be also shown there.

Chapter 2

RSA Construction

RSA is a public key cryptosystem, meaning that the key used for encryption is made public, while the key used for decryption is kept private. Everyone can encrypt messages but only the holder of the private key can decrypt them.

2.1 Key-Construction

2.1.1 Original RSA

In order to understand how the algorithm works, as well as easily follow the steps involved in the encryption and decryption process, we will recall some fundamental topics from elementary number theory.

Theorem 2.1. *Let a and b be integers with $b > 0$. Then there exists unique integers q and r such that*

$$a = bq + r$$

with $0 \leq r < b$.

For a proof of the theorem see [16].

Definition 2.1. For any positive integer $n > 1$, let $\mathbb{Z}_n$ denote the set $\{0, 1, \ldots, n-1\}$. An element $a \in \mathbb{Z}_n$ is **invertible** modulo n if and only if there is an element $b \in \mathbb{Z}_n$ such that $ab \equiv 1 (\text{mod } n)$. The element b exists if and only if the greatest common divisor of a and n, namely the largest integer d such that $d|a$ and $d|n$, denoted by $\gcd(a, n)$, is equal to 1.

Theorem 2.2. *The **Euclidean algorithm** is a method for computing the greatest common divisor of two positive integers. The first step in finding the* gcd *between two integers, say a and b with $a > b$, is dividing a by b, hence represent a in the form*

$$a = q_1 b + r_1.$$

If $r_1 = 0$ then b divides a and the greatest common divisor is b. If $r \neq 0$, then continue by representing b in the form

$$b = q_2 r_1 + r_2.$$

Continue in this way until the remainder is zero

$$r_1 = q_3 r_2 + r_3$$

$$\vdots$$

$$r_{k-2} = q_k r_{k-1} + r_k$$

$$r_{k-1} = q_{k+1} r_k.$$

It results that $\gcd(a, b) = r_k$.

The ***extended Euclidean algorithm*** *finds integers* x *and* y *such that*

$$ax + by = \gcd(a, b).$$

The first step is writing the last non-zero remainder from the next-to-last equation in the Euclidean algorithm, as a linear combination of the other two terms, namely $r_k = r_{k-2} - q_k r_{k-1}$. *The next step is writing the remainder obtained before the last non-zero remainder as a linear combination of the other terms, and substituting it in the equation obtained previously. More precisely, write* $r_{k-1} = r_{k-3} - q_{k-1} r_{k-2}$, *and substitute it in* $r_k = r_{k-2} - q_k r_{k-1}$. *Continue in this way until* r_k *will be a linear combination of* a *and* b.

For a proof of the theorem see [27] (p.98-99, p.104).

Let us illustrate this with an example.

Example 2.1. Consider the set $\mathbb{Z}_{28}$. Find the inverse, in case it exists, for the elements 2 and 5 respectively, in $\mathbb{Z}_{28}$.

Solution: Finding the inverse of the element 2 in $\mathbb{Z}_{28}$ is equivalent to solving the equation:

$$2 \cdot x \equiv 1 \mod 28,$$

for $x \in \mathbb{Z}_{28}$. The first thing we need to do is to check whether the inverse of 2 exists. How do we do this? We compute the $\gcd(2, 28)$ and if the result is equal to 1 then we proceed to finding the inverse. In this particular case we see the **prime factorizations** of 2 and 28, namely $2 = 2 \cdot 1$ and $28 = 2^2 \cdot 7$, from which we conclude that $\gcd(2, 28) = 2$. It follows that we cannot find the inverse of 2 because it does not exist.

Let us now attempt to find the inverse of the element 5 in $\mathbb{Z}_{28}$. This is equivalent to solving the congruence:

$$5 \cdot x \equiv 1 \mod 28,$$

for $x \in \mathbb{Z}_{28}$. Does the inverse exist? In order to answer this question we firstly compute the $\gcd(5, 28)$, using this time the **Euclidean algorithm** since it is more efficient than prime factorization method used previously. In what follows we will show how the Euclidean algorithm is used.
First, divide 28, the larger of the two integers, by 5, the smaller, to obtain

$$28 = 5 \cdot 5 + 3.$$

Any divisor of 5 and 3 must also be a divisor of $28 = 5 \cdot 5 + 3$. Thus, the greatest common divisor of 5 and 28 is the same as the greatest common divisor of 5 and 3. This means that the problem of finding $\gcd(28, 5)$ has been reduced to the problem of finding $\gcd(5, 3)$.

Next, divide 5 by 3 to obtain

$$5 = 3 \cdot 1 + 2.$$

Any divisor of 3 and 2 must also be a divisor of $5 = 3 \cdot 1 + 2$. It follows that $\gcd(5, 3) = \gcd(3, 2)$.

Next, divide 3 by 2 to obtain

$$3 = 2 \cdot 1 + 1.$$

Continue by dividing 2 by 1, to obtain

$$2 = 2 \cdot 1 + 0.$$

Because 1 divides 2, it follows that $\gcd(2, 1) = 1$. Furthermore, because $\gcd(5, 28) = \gcd(5, 3) = \gcd(3, 2) = \gcd(2, 1) = 1$, the original problem has been solved.

Finding the greatest common divisor of 28 and 5 being equal to 1, it means that we can go on and find the inverse of 5, since we know now that it exists. For this we will use the **extended Euclidean algorithm** that will be explained step by step in what follows.

To find the greatest common divisor of 5 and 28, the Euclidean algorithm uses 4 divisions as we previously saw. Using the second-to-last division (the third one), we can express $\gcd(5, 28) = 1$ as a linear combination of 3 and 2. We find that

$$1 = 3 - 2 \cdot 1.$$

The second division tells us

$$2 = 5 - 3 \cdot 1.$$

Substituting this expression for 2 into the previous equation, we can express 1 as a linear combination of 3 and 5. We get

$$1 = 3 - 2 \cdot 1 = 3 - 1 \cdot (5 - 3 \cdot 1) = -5 + 2 \cdot 3.$$

The first division tells us that

$$3 = 28 - 5 \cdot 5.$$

Substituting this expression for 3 into the previous equation, we can express 1 as a combination of 28 and 5. We conclude that

$$1 = -5 + 2(28 - 5 \cdot 5) = 2 \cdot 28 - 11 \cdot 5.$$

Now writing this equation as a congruence equation modulo 28 we get

$$2 \cdot 28 - 11 \cdot 5 \equiv 1 \mod 28.$$

But $2 \cdot 28$ is equivalent to 0 modulo 28, therefore

$$-11 \cdot 5 \equiv 1 \mod 28.$$

Hence we have now found an integer belonging to $\mathbb{Z}_{28}$ such that when multiplied by 5 modulo 28 it gives the result 1. As such -11, which is congruent to 17 modulo 28 , is the inverse of 5.

Definition 2.2. Let $n \geq 1$ be an integer. The **Euler phi function** denoted by $\phi(n)$ is the number of positive integers less than n that are relatively prime to n.

Example 2.2. Calculate the value of the Euler phi function for 12, 13 and 14.

Solution: In order to calculate $\phi(12)$ we look for the integers smaller than 12 , which are relatively prime to 12. The number of such integers represents the value of the Euler phi function.

$$\phi(12) = |\{1, 5, 7, 11\}| = 4.$$

In the same way

$$\phi(13) = |\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}| = 12$$

and

$$\phi(14) = |\{1, 3, 5, 9, 11, 13\}| = 6.$$

We now recall how one forms both the public and private key for the RSA cryptosystem:

1. Choose two large primes p and q such that $p \neq q$. Finding a large range of primes is usually done with the help of a **prime number sieve**, which works by creating a list of all integers up to a desired limit and progressively removing the composite numbers until only primes are left. The oldest such sieve is the **sieve of Eratosthenes**. For generating large primes used in cryptography, a random range of odd numbers of a desired size is sieved against small primes (usually all primes less than 65000). The remaining probable primes are randomly tested using primality tests, such as the **Miller-Rabin primality test**, which states whether the given input is either not a prime or a probable prime [17].

2. Put $n = pq$.

3. Calculate the number of elements in $\mathbb{Z}_n$ that are invertible modulo n. Denote it s. We notice here that the number of invertible elements in $\mathbb{Z}_n$ is in fact the number of elements which are relatively prime to n.

   Theorem 2.3. *The number of invertible elements in $\mathbb{Z}_n$ denoted by s, with n being the product of two primes p and q, is given by the following formula:*

   $$s = (p - 1)(q - 1).$$

Proof. In order to calculate the number of invertible elements in $\mathbb{Z}_n$, simply enough to imagine, we have to delete from this set those elements that are not invertible. We recall that an element a belonging to a set $\mathbb{Z}_m$ has an inverse, say a_{inv} in this set, such that

$$aa_{inv} \equiv 1 \mod m,$$

if and only if $\gcd(a, m) = 1$. See [7].

This means that we have to delete those elements that are not relatively prime to n. Since n is the product of the two primes p and q it results that the greatest common divisor of n and p or of n and a multiple of p will always be p .The same happens for q. Thus, n and any multiple of p or q are not relatively prime. Therefore in order to find the invertible elements, we have to delete all multiples of p and all multiples of q and of course add the common multiples of p and q because we do not want to delete one element twice.

Firstly, we find the multiples of p and q respectively in the set $\mathbb{Z}_{pq}$.The multiples of p are: $0 \cdot p, 1 \cdot p, \ldots, (q-1) \cdot p$. The multiples of q are: $0 \cdot q, 1 \cdot q, \ldots, (p-1) \cdot q$. We see here that there are q multiples of p and p multiples of q. Now we have to find the common multiples. Before doing that we recall the **Fundamental Theorem of Arithmetic** which states that every integer greater than 1 either is prime itself or is the product of prime numbers. The factorization into primes is unique [19]. We know therefore that the factorization of n into the product of the prime numbers p and q is unique. As such this is the reason why the only common multiple of p and q can be 0.

We recall now that we said we would find the number of invertible elements in $\mathbb{Z}_n$ by deleting from the number of all elements in $\mathbb{Z}_n$, namely n elements, the number of elements that are **not** invertible. In other words, in our case s is :

s =number of elements in $\mathbb{Z}_n-$ multiples of $p-$ multiples of $q+$ common multiples of p and q.

Having found all the values we need in the formula above, we can replace them, getting thus the following:

$$s = n - q - p + 1 = pq - p - q + 1 = (p-1)(q-1).$$

This proves the theorem.

□

4. Choose $e \in \mathbb{Z}_s$ such that $\gcd(e, s) = 1$.
5. Compute the multiplicative inverse $d = e^{-1} \mod s$.

The pair (n,e) will be the public key, while the triple (p,q,d) represents the private key. To encrypt, the plaintext (the message to be sent) is first encoded into a sequence of integers m_1, m_2,...,$m_k \in \mathbb{Z}_n$. Each m_i is then encrypted by computing

$$c_i = m_i^e \mod n.$$

To decrypt, one has to compute

$$m_i = c_i^d \mod n.$$

How and why the decryption formula works, will be explained in Chapter 3.

2.1.2 Polynomial RSA

In order to fully understand how the polynomial version of RSA, proposed in this report, works, one needs a quick refreshment of several concepts found in abstract algebra.

Definition 2.3. Let M be a set. By a **binary operation** $*$ on M we mean a mapping

$$M \times M \ni (a, b) \longmapsto a * b \in M.$$

Example 2.3. Which of the following arithmetic operations $+,-,\times,\div$ are binary operations on $\mathbb{N}$?

Solution: Let us first check whether addition is a binary operation on $\mathbb{N}$. The definition above tells us that for the operation to be binary then for any two numbers in $\mathbb{N}$, their addition should also belong to $\mathbb{N}$, namely for $a, b \in \mathbb{N}$, $a + b \in \mathbb{N}$. This holds for every number in $\mathbb{N}$, therefore addition is a binary operation on $\mathbb{N}$.

For subtraction, the question is whether the difference of two natural numbers is always a natural number, namely if for $a, b \in \mathbb{N}$, $a - b \in \mathbb{N}$. This is not true for all $a, b \in \mathbb{N}$, since if a is smaller than b, then their difference would be a negative number, which is not in $\mathbb{N}$. Therefore subtraction is not a binary operation on $\mathbb{N}$.

For multiplication to be a binary operation, then $a \times b$ should belong to $\mathbb{N}$ for any $a, b \in \mathbb{N}$. This is true so multiplication is in fact a binary operation on $\mathbb{N}$.

In the case of division, the question is whether any two natural numbers, when divided, give a natural number, namely if for $a, b \in \mathbb{N}$, $a \div b \in \mathbb{N}$. The answer is no since for two relatively prime natural numbers, their division is definitely not a natural number. Therefore division is not a binary operation on $\mathbb{N}$.

Definition 2.4. Let G be a non-empty set and $*$ a binary operation on G. The pair $(G, *)$ is called a **group** if

- $*$ is associative, which means that for all a, b, $c \in G$ $(a * b) * c = a * (b * c)$,
- G contains an identity element with respect to $*$, meaning that there is an element $e \in G$ such that for all $a \in G$ $a * e = e * a = a$,
- each element in G has an inverse in G with respect to $*$, meaning that for every $a \in G$, there exists an element $a' \in G$ such that $a * a' = a' * a = e$.

If in addition $*$ is commutative, meaning that for all a,$b \in G$ $a * b = b * a$, then the group is called an **Abelian group**.

The group G is said to be **finite** if it has a finite number of elements. In this case, the number of elements in G is called the **order of** G and is denoted by $|G|$. A group with infinitely many elements is said to have **infinite order**.

Example 2.4. Show that $(\mathbb{Z}_5, +)$, where addition is a binary operation defined as addition modulo 5, forms an Abelian group.

Solution: In order to prove that the given structure is an Abelian group, we have to show that the three axioms defined in the definition above, are true.

1. Associativity

   We easily see that if for any a, b, $c \in \mathbb{Z}_5$, $(a + b) + c = a + (b + c)$ holds.

2. Identity element

   There exists an element $e \in \mathbb{Z}_5$ such that for any $a \in \mathbb{Z}_5$, $e + a = a + e = a$, namely $e = 0$.

3. Inverse element

   There exists an element $a' \in \mathbb{Z}_5$ for every $a \in \mathbb{Z}_5$, such that $a + a' = a' + a = e$, namely $a' = -a$.

   Because the addition modulo 5 is also commutative, since for any a, $b \in \mathbb{Z}_5$ $a + b = b + a$, and because we have proved that the three axioms above hold, we conclude that $(\mathbb{Z}_5, +)$ is an Abelian group.

Definition 2.5. If a subset H of a group G is closed under the binary operation of G, and if H with the induced operation from G is itself a group, then H is a **subgroup** of G.

Example 2.5. One can see that $(\mathbb{Q}^+, \cdot)$ is a subgroup of $(\mathbb{R}^+, \cdot)$.

Definition 2.6. Let H be a subgroup of a group G. The subset $aH = ah | h \in H$ of G is the **left coset** of H containing a, while the subset $Ha = ha | h \in H$ is the **right coset** of H containing a.

Example 2.6. Exhibit the left and the right cosets of $5\mathbb{Z}$ the subgroup of $\mathbb{Z}$.

Solution: The notation here is additive so the left coset of $5\mathbb{Z}$ containing l is $l + 5\mathbb{Z}$. Taking $l = 0$, we see that

$$5\mathbb{Z} = \{\dots, -15, -10, -5, 0, 5, 10, 15, \dots\}$$

is itself one of its left cosets, the coset containing 0. To find another left coset, we select an element of $\mathbb{Z}$, not in $5\mathbb{Z}$, say 1, and find the left coset containing it. We have

$$1 + 5\mathbb{Z} = \{\dots, -14, -9, -4, 1, 6, 11, 16, \dots\}.$$

The two left cosets, $5\mathbb{Z}$ and $5\mathbb{Z} + 1$, do not exhaust $\mathbb{Z}$. For example, 2 is in none of them. The left coset containing 2 is

$$2 + 5\mathbb{Z} = \{\dots, -13, -8, -3, 2, 7, 12, 17, \dots\}.$$

The left coset containing 3 is

$$3 + 5\mathbb{Z} = \{\dots, -12, -7, -2, 3, 8, 13, 18, \dots\}.$$

The left coset containing 4 is

$$4 + 5\mathbb{Z} = \{\dots, -11, -6, -1, 4, 9, 14, 19, \dots\}.$$

It is clear that these five left cosets we have found do exhaust $\mathbb{Z}$, so they are the partition of $\mathbb{Z}$ into left cosets of $5\mathbb{Z}$.

Since $\mathbb{Z}$ is abelian, the left and right cosets are the same, so the partition of $\mathbb{Z}$ into right cosets is the same.

Definition 2.7. The order of an element a belonging to a finite group G, denoted by $o(a)$ is the smallest positive integer k such that

$$a^k = 1_G,$$

where 1_G is the identity element of the group.

Theorem 2.4. *The order of an element of a finite group divides the order of the group.*

For a proof of the theorem see [8] (p.101).

Definition 2.8. A **ring** is a triple $(R, +, \cdot)$, where R is a set and where $+$ and $\cdot$ are binary operations on R, called addition and multiplication respectively, such that

- $(R, +)$ is an Abelian group
- $\cdot$ is associative
- $\cdot$ is distributive over $+$, meaning that for all a, b, $c \in G$, the following two equalities hold

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

$$(b + c) \cdot a = (b \cdot a) + (c \cdot a).$$

Example 2.7. Show that the structure $(\mathbb{Z}_5, +, \cdot)$, where addition and multiplication are binary operations defined as addition and multiplication modulo 5, forms a ring.

Solution: In order to prove what it is required, we have to show that the three axioms, defined in the above definition, hold.

1. $(\mathbb{Z}_5, +)$ forms an Abelian group

   We have proved this in the previous example.

2. Associativity of $\cdot$.

   It is clear that for any a, b, $c \in \mathbb{Z}_5$ the associativity law holds $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

3. Distributivity

   We have learned in our first years of school that multiplication is distributive over addition, namely that $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(b + c) \cdot a = b \cdot a + c \cdot a$ both are true. The same happens also for addition and multiplication modulo an integer.

   Therefore we conclude that that $(\mathbb{Z}_n, +, \cdot)$ is a ring.

Definition 2.9. An **integral domain D** is a commutative ring with unity $1 \neq 0$ (the multiplicative identity element), in which the product of any two nonzero elements is nonzero.

Example 2.8. We notice that $\mathbb{Z}$ and $\mathbb{Z}_p$ for any prime p, are integral domains.

Definition 2.10. An additive subgroup N of a ring R satisfying the properties

$$aN \subseteq N$$

and

$$Nb \subseteq N$$

for all $a, b \in R$ is an **ideal**.

Definition 2.11. If R is a commutative ring with unity and $a \in R$, the ideal $ra|r \in R$ of all multiples of a is the **principal ideal** generated by a and is denoted by $\langle a \rangle$. An ideal N of R is a principal ideal if $N = \langle a \rangle$ for some $a \in R$.

Example 2.9. Every ideal of the ring $\mathbb{Z}$ is of the form $n\mathbb{Z}$, which is generated by n, so every ideal of $\mathbb{Z}$ is a principal ideal.

Definition 2.12. Let N be an ideal of a ring R. Then the additive cosets of N form a ring R/N with the binary operations defined by

$$(a + N) + (b + N) = (a + b) + N$$

$$(a + N)(b + N) = ab + N,$$

called the **factor ring** of R by N.

Example 2.10. Consider the factor group $\mathbb{Z}/5\mathbb{Z}$ with the cosets shown in Example 2.6. We can add $(3 + 5\mathbb{Z}) + (4 + 5\mathbb{Z})$ by choosing 3 and 4, finding $3 + 4 = 7$, and noticing that 7 is in the coset $2 + 5\mathbb{Z}$. We could as well add these two cosets by choosing -12 in $3 + 5\mathbb{Z}$ and 19 in $4 + 5\mathbb{Z}$. The sum $-12 + 19 = 7$ is still in the coset $2 + 5\mathbb{Z}$.

Definition 2.13. A **polynomial in one determinant x, over a ring** R is an infinite formal sum

$$f(x) = \sum_{k=0}^{\infty} a_k x^k = a_0 + a_1 x + \cdots + a_n x^n + \ldots,$$

where $a_k \in R$ for all k, and where $a_k = 0_R$ for all but a finite number of values of k. We denote the set of such polynomials by $R[x]$.

Definition 2.14. Let R be a ring, and let $f(x) = \sum_{k=0}^{\infty} a_k x^k$ and $g(x) = \sum_{k=0}^{\infty} b_k x^k$ be two polynomials in $R[x]$. Then the **sum** of $f(x)$ and $g(x)$ is defined as

$$f(x) + g(x) = \sum_{k=0}^{\infty} a_k x^k,$$

where $c_k = a_k + b_k \in R$ for all k.

The **product** of $f(x)$ and $g(x)$ is defined as

$$f(x)g(x) = \sum_{k=0}^{\infty} d_k x^k,$$

where $d_k = \sum_{j=0}^{k} a_j b_{k-j} \in R$ for all k.

Theorem 2.5. *If R is a ring, then the set $R[x]$ of all polynomials over R is also a ring with respect to addition and multiplication of polynomials.*

Example 2.11. Let $f(x) = 2x^2 + 3x + 5$ and $g(x) = 4x^2 + 2x + 1$ be two polynomials in $\mathbb{Z}_5$. Compute their sum and their product.

Solution:
Their sum is

$$f(x) + g(x) = (2x^2 + 3x + 1) + (4x^2 + 2x + 1)$$
$$f(x) + g(x) = (2 + 4)x^2 + (3 + 2)x + (1 + 1)$$
$$f(x) + g(x) = x^2 + 2$$

Their product is

$$f(x)g(x) = (2x^2 + 3x + 1)(4x^2 + 2x + 1)$$
$$f(x)g(x) = (2 \cdot 4)x^4 + (3 \cdot 4 + 2 \cdot 2)x^3 + (1 \cdot 4 + 2 \cdot 3 + 2 \cdot 1)x^2 + (1 \cdot 2 + 3 \cdot 1)x + 1$$
$$f(x)g(x) = 3x^4 + x^3 + 3x^2 + 1.$$

Definition 2.15. A **field** is a triple $(\mathbb{F}, +, \cdot)$, where $\mathbb{F}$ is a set and where $+$ and $\cdot$ are binary operations on $\mathbb{F}$, called addition and multiplication respectively, such that

- $(\mathbb{F}, +)$ is an Abelian group
- $(\mathbb{F} \setminus \{0\}, \cdot)$ is an Abelian group
- $\cdot$ is distributive over $+$

Example 2.12. Show that the algebraic structure $(\mathbb{Z}_5, +, \cdot)$, where addition and multiplication are binary operations defined as addition and multiplication modulo 5, forms a field.

Solution: In order to show that a structure is a field, we have to show that the three axioms defined above are true.

We have previously showed (Example 2.5), that $(\mathbb{Z}_5, +)$ is an Abelian group and that multiplication modulo 5 is distributive over addition modulo 5.

Thus the only thing left to show is that $(\mathbb{Z}_5 \setminus \{0\}, \cdot)$ is an Abelian group. In the following we will use the notation $\mathbb{Z}_5^*$ for $\mathbb{Z}_5 \setminus \{0\}$. The binary operation $\cdot$ fulfills the associative law $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, the commutative law $a \cdot b = b \cdot a$, for all $a, b, c \in \mathbb{Z}_5^*$,there exists an identity element $e \in \mathbb{Z}_5^*$ such that for any $a \in \mathbb{Z}_5^*$ $a \cdot e = e \cdot a = a$, namely $e = 1$, and also there exists an element $a' \in \mathbb{Z}_5$ for every $a \in \mathbb{Z}_5$, such that $a \cdot a' = e = 1$. In order to attest the existence of such a', we shall recall that an equation $ax \equiv 1 \mod n$ with $a \in \mathbb{Z}_n$ has a solution for x if the $\gcd(a, n) = 1$. In our case we shall demonstrate that every element from the set $\mathbb{Z}_5^* = \{1, 2, 3, 4\}$ has an inverse in this set, more precisely that the equation $ax \equiv 1 \mod 5$ has a solution for every $a \in \{1, 2, 3, 4\}$. Since in this specific case $n = 5$, a prime number, we see directly that the greatest common divisor between 5 and any of the elements in the set $\mathbb{Z}_5^*$ is equal to 1. Therefore there is an inverse for every element in the set.

Hence we have now proved that the structure $(\mathbb{Z}_5, +, \cdot)$ is a field.

Definition 2.16. Let $\mathbb{F}$ be a field. A non-constant polynomial $f(x) \in \mathbb{F}[x]$ is said to be **irreducible over** $\mathbb{F}$ if it is **not** possible to write $f(x)$ as a product $f(x) = g(x)h(x)$ of two polynomials $g(x)$, $h(x) \in \mathbb{F}[x]$, both of smaller degree than $f(x)$. If $f(x)$ is **not** irreducible, then it is said to be **reducible**.

Example 2.13. Show that the polynomial $x^2 + 3x + 2$ is reducible over $\mathbb{Z}_5$. In addition, show that the polynomial $x^2 + 2$ is irreducible over $\mathbb{Z}_5$.

Solution: In order to show that the polynomial $x^2 + 3x + 2$ is reducible over $\mathbb{Z}_5$, we have to show that it can be written as a product of two polynomials, say $h(x), g(x) \in \mathbb{Z}_5[x]$, both of degree smaller than the degree of the given polynomial. This means that the polynomials $h(x)$ and $g(x)$ should be both of degree one. It is easy to see that for $h(x) = x + 1$ and $g(x) = x + 2$, then the product $h(x)g(x)$ gives exactly the polynomial $x^2 + 3x + 2$. Hence, the polynomial $x^2 + 3x + 2$ is reducible over $\mathbb{Z}_5$.

In order to show that the polynomial $x^2 + 2$ is irreducible over $\mathbb{Z}_5$, we have to fail to find a way to write it as a product of two polynomials, say $m(x)$ and $n(x) \in \mathbb{Z}_5[x]$, both of degree smaller than the

degree of the given polynomial. This means that for any two polynomials of degree one, that belong to $\mathbb{Z}_5[x]$, their product will never be x^2+2. Consider $m(x)=ax+b$ and $n(x)=cx+d$, with $a,b,c,d\in\mathbb{Z}_5$. Suppose that $x^2+2=m(x)n(x)$, that is $x^2+2=acx^2+(b+d)x+bd$. This is possible only if we succeed finding elements $a,b,c,d\in\mathbb{Z}_5$, such that the product ac is congruent to one modulo five, the sum $b+d$ is congruent to zero modulo five, and the product bd is congruent to two modulo five. In other words the elements a,b,c,d must fulfill the following system of congruences:

$$ac\equiv 1 \mod 5$$

$$b+d\equiv 0 \mod 5$$

$$bd\equiv 2 \mod 5.$$

From the second congruence it follows that $b\equiv -d \bmod 5$, which in turn means that $b\equiv(-1)\cdot d \bmod 5$. Since $-1\equiv 4 \bmod 5$, we get that $b\equiv 4\cdot d \bmod 5$. Substituting b in the third congruence of the system, we get that $4d^2\equiv 2 \bmod 5$. Multiplying this equation with the inverse of four, which is in fact four ($4\cdot 4\equiv 1 \bmod 5$), we get $d^2\equiv 3 \bmod 5$. Now, the problem of showing that the polynomial x^2+2 is irreducible, boils down to showing that there is no element $d\in\mathbb{Z}_5$, such that its square root is congruent to three modulo five. With the five computations $0^2\equiv 0 \bmod 5$, $1^2\equiv 1 \bmod 5$, $2^2\equiv 4 \bmod 5$, $3^2\equiv 4 \bmod 5$ and $4^2\equiv 1 \bmod 5$, we have showed that there is no element in $\mathbb{Z}_5$ such that when raised to the power of two is congruent to three modulo five, which implies that the polynomial x^2+2 is irreducible over $\mathbb{Z}_5$.

Definition 2.17. We say that $f(x),g(x)\in\mathbb{Z}_p[x]$ are **associated polynomials** if $f(x)=ug(x)$ for some unit (invertible element) $u\in\mathbb{Z}_p$.

Theorem 2.6. *Let*

$$f(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_0$$

and

$$g(x)=b_mx^m+b_{m-1}x^{m-1}+\cdots+b_0$$

be two elements of a field $\mathbb{F}[x]$, with a_n and b_m both nonzero elements of $\mathbb{F}$ and $m>0$. Then there are unique polynomials $q(x)$ and $r(x)$ such that $f(x)=g(x)q(x)+r(x)$, where either $r(x)=0$ or the degree of $r(x)$ is less than the degree m of $g(x)$.

For a proof of the theorem see [8] (p.210).

Definition 2.18. Let $\mathbb{F}$ be a field and let $f(x)$, $g(x)\in\mathbb{F}[x]$. A **common divisor** of $f(x)$ and $g(x)$ is a polynomial $d(x)\in\mathbb{F}[x]$ such that $d(x)|f(x)$ and $d(x)|g(x)$. The common divisor of highest degree is called the **greatest common divisor**.

Example 2.14. Compute the $\gcd(x^3+4x^2+x+4,x^2+4)$ in $\mathbb{Z}_5[x]$.

Solution: Just as in the case of integer greatest common divisor computation, the Euclidean algorithm will be used here as well. The algorithm follows the same steps, as described in Theorem 2.2, just that here instead of integer divisions one will deal with polynomial divisions.

First, divide x^3+4x^2+x+4, the polynomial with higher degree, by x^2+4, the polynomial with lower degree, to obtain

$$x^3+4x^2+x+4=(x^2+4)(x+4)+2x+3.$$

Any divisor of x^2+4 and $2x+3$ must also be a divisor of $x^3+4x^2+x+4=(x^2+4)(x+4)+2x+3$. It follows that $\gcd(x^3+4x^2+x+4,x^2+4)=\gcd(x^2+4,2x+3)$. Next, divide x^2+4 by $2x+3$ to obtain

$$x^2+4=(2x+3)(3x+3)+0.$$

Because $3x+3$ divides x^2+4, it follows that $\gcd(x^2+4,2x+3)=3x+3$. Furthermore, because $\gcd(x^3+4x^2+x+4,x^2+4)=\gcd(x^2+4,2x+3)=3x+3$, the original problem has been solved.

Definition 2.19. Let $f(x)$, $g(x)$ and $n(x)$ belong to a field $\mathbb{F}[x]$ with $n(x)\neq 0$. We say that $f(x)$ **is congruent to** $g(x)$ **modulo** $n(x)$ if $n(x)$ divides $f(x)-g(x)$, and we write $f(x)\equiv g(x)(\bmod n(x))$.

Example 2.15. Show that the polynomials x^2+2 and $2x$ belonging to $\mathbb{Z}_5[x]$ are congruent modulo $x+1$.

Solution: The definition above tells us that x^2+2 is congruent to $2x$ modulo $x+1$ if $x+1$ divides x^2-2x+2. Since $-2 \equiv 3 \mod 5$, the polynomial x^2-2x+2 can be written as x^2+3x+2. We see that when dividing x^2+3x+2 by $x+1$ we get the quotient $x+2$ and the remainder zero. Hence, we have showed that $x+1$ divides x^2+3x+2 which implies that

$$x^2+2 \equiv 2x \mod (x+1).$$

Definition 2.20. A polynomial $f(x) \in \mathbb{F}[x]$ is said to be **invertible** modulo $n(x) \in \mathbb{F}[x]$ if there is a polynomial $g(x) \in \mathbb{F}[x]$ such that $f(x)g(x) \equiv 1 (\mod n(x))$.

Example 2.16. Show that the polynomial $f(x) = x+2 \in \mathbb{Z}_5[x]$ is invertible modulo x.

Solution: In order to show that $f(x)$ is invertible modulo x, we have to find a polynomial $g(x)$ such that the product $f(x)g(x)$ is congruent to one modulo x. In other words, we have to find $g(x) \in \mathbb{Z}_5[x]$ such that

$$(x+2)g(x) \equiv 1 \mod x.$$

This means that the polynomial x has to divide the difference $(x+2)g(x)-1$. If we choose $g(x)=x+3$, then the difference becomes $(x+2)(x+3)-1 = x^2+5x+6-1 = x^2$. We notice that x^2 is divisible by x and since the polynomial $n(x)=x$ divides $f(x)g(x)-1 = (x+2)(x+3)-1$, it follows that

$$(x+2)(x+3) \equiv 1 \mod x.$$

Theorem 2.7. *Let $\mathbb{F}$ be a field, and $f(x)$ a nonzero member of $\mathbb{F}[x]$. Then $f(x)$ can be written as a product $f(x) = c\prod_{k=1}^{n} f_k(x)$ of a nonzero constant c and a collection of monic irreducible polynomials $f_k(x)$. This factorization is unique up to the order in which the irreducibles $f_k(x)$ are taken.*

For a proof of the theorem see [18].

Whereas in the original version of RSA cryptosystem the plaintext blocks and the ciphertext blocks respectively are encoded as elements in $\mathbb{Z}_n$, which is a ring with respect to addition and multiplication modulo n, in the version proposed in this report, the RSA implementation will be done in the polynomial ring

$$\mathbb{Z}_p[x] = \left\{a_0 + a_1x + \cdots + a_kx^k | k \geq 0, a_i \in \mathbb{Z}_p\right\}$$

where p is a prime and the operations addition and multiplication are done modulo a polynomial.

We now pattern after the RSA construction as follows:

1. Pick two irreducible polynomials $P(x)$, $Q(x) \in \mathbb{Z}_p[x]$; $P(x)$ and $Q(x)$ not associated. Generating irreducible polynomials of degree n, over the finite field $\mathbb{F}_q$ is done following the same path as in the case of generating primes, randomizing at first a monic polynomial of degree n, and then applying an irreducibility test on that specific polynomial, such as **Rabin's test for irreducibility**. The algorithm takes as input a polynomial $f(x) \in \mathbb{F}_q[x]$ of degree n. Let $p_1, \ldots, p_k$ be all the prime divisors of n, and denote $n_i = n/p_i$, for $1 \leq i \leq k$. The Rabin's irreducibility test is based on the fact that a polynomial $f(x) \in \mathbb{F}_q[x]$ is irreducible in $\mathbb{F}_q[x]$ if and only if $\gcd(f(x), x^{q^{n_i}} - x \mod f) = 1$ for $1 \leq i \leq k$, and $f(x)$ divides $x^{q^n} - x$. See [25].

2. Compute $N(x) = P(x)Q(x)$ in $\mathbb{Z}_p[x]$.

3. Let $R = \mathbb{Z}_p[x]/\langle N(x)\rangle$ denote the set consisting of all possible remainders when any polynomial in $\mathbb{Z}_p[x]$ is divided by $N(x)$. Calculate the number of elements in R that are invertible modulo $N(x)$. Denote it s.

   In other words, the set R represents all the polynomials in $\mathbb{Z}_p$ of degree smaller than the degree of $N(x)$. The question now is how to find the number of all the polynomials in R that are actually invertible. Since finding the polynomials that are not invertible proves to be an easier task we will find the value of s, denoting the number of invertible elements, by subtracting from the set R all the polynomials that are not invertible.

Theorem 2.8. *Let $A(x)$ and $N(x)$ be two polynomials in $\mathbb{Z}_p[x]$. The polynomial $A(x)$ has an inverse $A^{-1}(x) \in \mathbb{Z}_p[x]$ modulo $N(x)$, for which*

$$A(x)A^{-1}(x) \equiv 1 \mod (N(x)),$$

if and only if $\gcd(A(x), N(x)) = 1$.

Proof. Suppose the following congruence holds

$$A(x)A^{-1}(x) \equiv 1 \mod (N(x))$$

It follows that there exists $Q(x) \in \mathbb{Z}_p[x]$ such that $A(x)A^{-1}(x) = N(x)Q(x) + 1$. Having $D(x)$ denoting the $\gcd(A(x), N(x))$, we know that $D(x)$ divides both $A(x)$ and $N(x)$. Therefore $D(x)$ divides any linear combination of $A(x)$ and $N(x)$, and hence $D(x)$ divides $1 = A(x)A^{-1}(x) - Q(x)N(x)$. Since $D(x)|1$, the largest it could possibly be is the constant polynomial 1 itself. □

Since $N(x)$ is the product of the two irreducible polynomials $P(x)$ and $Q(x)$ it follows that the greatest common divisor between $N(x)$ and any multiple of $P(x)$ or $Q(x)$ is different from 1. Therefore the number of invertible polynomials in R is calculated by subtracting from R the multiples of $P(x)$, the multiples of $Q(x)$ and add the common multiples of the polynomials because we do not want to subtract any polynomial twice.

By assuming that the two irreducible polynomials, factors of $N(x)$, are **not** associated and due to the fact that the factorization is unique, as stated in Theorem 2.6 we may conclude that the only common multiple of $P(x)$ and $Q(x)$ is the zero polynomial, just as the only common multiple of two primes is zero.

Theorem 2.9. *The number of invertible polynomials in the ring R, denoted by s, is given by the following formula*

$$s = (p^m - 1)(p^n - 1).$$

Proof. First of all, we have to calculate the number of polynomials in the set R. That is, find all polynomials in $\mathbb{Z}_p[x]$ of degree smaller than the degree of $N(x)$. We consider

$$P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$
$$Q(x) = b_m x^m + b_{m-1}x^{m-1} + \cdots + b_1 x + b_0$$
$$N(x) = P(x)Q(x)$$

with degree n of $P(x)$, degree m of $Q(x)$ and degree $m+n$ of $N(x)$.

A polynomial $K(x) \in \mathbb{Z}_p[x]$ of degree smaller than $m+n$ is of the form $K(x) = \sum_{i=0}^{m+n-1} k_i x^i$ for some $k_i \in \mathbb{Z}_p$. We see here that $K(x)$ can be chosen in p^{m+n} different ways since we have $m+n$ coefficients $k_0, k_1, \ldots, k_{m+n-1}$ to choose, and each of them can be chosen in p different ways.

Next, we will find out the number of multiples of $P(x)$ and $Q(x)$ respectively. A multiple of $P(x)$ is of the form $P(x)A(x)$, where $\deg A(x) < m$. Therefore $A(x) = \sum_{i=0}^{m-1} k_i x^i$ for some $k_i \in \mathbb{Z}_p$. We see here that the polynomial $A(x)$ can be chosen in p^m different ways since we have m coefficients $k_0, k_1, \ldots, k_{m-1}$ to chose, and each of them can be chosen in p different ways.

The same situation is when we compute the number of multiples of $Q(x)$. A multiple of $Q(x)$ is of the form $Q(x)B(x)$, where $\deg B(x) < n$. Therefore $B(x) = \sum_{i=0}^{n-1} k_i x^i$ for some $k_i \in \mathbb{Z}_p$. We see here that the polynomial $B(x)$ can be chosen in p^n different ways since we have n coefficients $k_0, k_1, \ldots, k_{n-1}$ to choose, and each of them can be chosen in p different ways.

Gathering all what we have found previously , we find a general formula for computing the number of invertible polynomials in the set R

$$s = p^{m+n} - p^m - p^n + 1 = (p^m - 1)(p^n - 1).$$

□

4. Choose $e \in \mathbb{Z}_s = \{0, 1, 2, 3, 4, \ldots s-1\}$ such that $\gcd(e, s) = 1$.

5. Compute the multiplicative inverse $d = e^{-1} \mod s$, in other words find $d \in \mathbb{Z}_s$ such that $ed \equiv 1 \mod s$.

Chapter 3

Why does it work?

In this chapter we will explain why the decryption algorithm successfully restores the plaintext as well as how the mathematics that stand behind the decryption process assures the cryptosystem's correctness.

To begin with we will look at the original RSA. As we have seen earlier one encrypts a message m in the following way

$$m^e \mod n.$$

This message, denoting the ciphertext c, might then be decrypted by performing

$$c^d \mod n.$$

The question we might ask ourselves is how and why does this procedure work. First of all we know that the ciphertext c represents the eth power of the initial message m modulo n, namely

$$c = m^e \mod n.$$

The plaintext is then restored by taking the dth power of the ciphertext modulo n, namely

$$c^d = (m^e)^d = m^{ed} \mod n.$$

When calculating the decryption exponent d, we solve the congruence $ed \equiv 1 \bmod s$. From this equation we can write the product ed in terms of s in this way $ed = sk + 1$, where k is an integer. As such the decryption process becomes

$$c^d = m^{ed} = m^{sk+1} = m^{sk}m = (m^s)^k m \mod n.$$

In what follows we will show that for any message $m \in \mathbb{Z}_n$ $m^s \equiv 1 \bmod n$, where s denotes as we remember the number of elements in $\mathbb{Z}_n$ that are invertible modulo n.

Definition 3.1. A **ring with identity** is a ring R that contains an element 1_R satisfying this axiom

$$a \cdot 1_R = 1_R \cdot a = a,$$

for all $a \in R$.

Theorem 3.1. *If R is a ring with identity, then the set U of all units (invertible elements) in R is a group under multiplication.*

For a proof of the theorem see [8] (p.186).

Theorem 3.2. *Let $(G, \cdot)$ be a finite group under multiplication with $|G| = n$ and let 1_G denote the identity element. Then for any $a \in G$ the following holds*

$$a^n = 1_G.$$

Proof. Let $(G, \cdot)$ be a finite group under multiplication with $|G| = n$ and let 1_G denote the identity element. The order of any element $a \in G$ is the smallest k such that $a^k = 1_G$. By Theorem 2.4 we know that the order of an element $a \in G$ divides the order of the group which is in this case n. Hence, it follows that $o(a)|n$, from which it results that $n = o(a) \cdot m$ for some integer m. Now, raising an element a to the power of n, we get

$$a^n = a^{o(a)\cdot m} = a^{o(a)^m} = 1_G{}^m = 1_G,$$

which proves the theorem. □

The residue class $\mathbb{Z}_n$ together with the two binary operations addition and multiplication modulo n, forms the structure of a ring. Since there exists an element $e_1 \in \mathbb{Z}_n$ such that $e_1 \cdot a = a \cdot e_1 = a$, namely $e_1 = 1$, it means that the ring $(\mathbb{Z}_n, +, \cdot)$ is a ring with identity.

The theorem above tells us that the set of all invertible elements in $\mathbb{Z}_n$, denoted by $\mathbb{Z}_n^*$, is a group under multiplication. In other words the algebraic structure $(\mathbb{Z}_n^*, \cdot)$ represents a group. In addition it is a finite group with $|\mathbb{Z}_n^*| = s = \phi(n) = (p-1)(q-1)$. By Theorem 3.2 it follows that any element belonging to $\mathbb{Z}_n^*$ raised to the power s is equal to the identity element with respect to multiplication modulo n, which in this case is equal to 1. Therefore we have showed that for any $a \in \mathbb{Z}_n^*$

$$a^s \equiv 1 \mod n.$$

Going back to our decryption equation, we have

$$c^d = (m^s)^k m = 1^k m = m \mod n.$$

We have now showed that the decryption formula works for any message a belonging to $\mathbb{Z}_n^*$.

One might ask now what if a chosen plaintext is not encoded in $\mathbb{Z}_n^*$. In other words what happens if the message to be sent is first encoded as an integer in $\mathbb{Z}_n$, which is not relatively prime to n. Will the decryption formula still work?

We write the decryption formula as before, but this time modulo p and q respectively

$$c^d = m^{ed} = m^{sk+1} = m^{sk} m = (m^s)^k m \mod p,$$

$$c^d = m^{ed} = m^{sk+1} = m^{sk} m = (m^s)^k m \mod q.$$

As we have earlier proved, the number of integers less than n, that are relatively prime to n, denoted by s, representing Euler's Phi function $\phi(n)$, is given by the formula $s = \phi(n) = (p-1)(q-1)$. Substituting s in the two formulas above, we get

$$m^{ed} = (m^{(p-1)(q-1)})^k m = (m^{p-1})^{(q-1)k} m \mod p$$

$$m^{ed} = (m^{(p-1)(q-1)})^k m = (m^{q-1})^{(p-1)k} m \mod q.$$

Theorem 3.3. ***(Fermat's Little Theorem)*** *Let a be a positive integer and p a prime. If a is not divisible by p, then the following is true*

$$a^{p-1} \equiv 1 \mod p.$$

For a proof of the theorem see [27] (p.217-218).

By using the above theorem we see that

$$(m^{p-1})^{(q-1)k} m \equiv 1^{(q-1)k} m \equiv m \mod p,$$

$$(m^{q-1})^{(p-1)k} m \equiv 1^{(p-1)k} m \equiv m \mod q.$$

Theorem 3.4. ***(Chinese Remainder Theorem)*** *Let $m_1, \ldots, m_r$ be positive integers that are pairwise relatively prime. The system of congruences*

$$x \equiv a_1 \mod m_1$$

$$\vdots$$

$$x \equiv a_r mod m_r$$

has then a unique solution modulo $M = m_1 \cdot \cdots \cdot m_r$.

For a proof of the theorem see [27] (p. 159).

We have now the system of the two congruences

$$m^{ed} \equiv m \mod p$$

$$m^{ed} \equiv m \mod q.$$

The Chinese Remainder Theorem tells us that the solution for m^{ed} modulo pq is **unique**. We can rewrite the above system of congruences as

$$m^{ed} - m \equiv 0 \mod p$$

$$m^{ed} - m \equiv 0 \mod q.$$

We see that $m^{ed} - m$ is both divisible by p and q. Since p and q are two different primes, it follows that $m^{ed} - m$ is also divisible by the product of p and q

$$m^{ed} - m \equiv 0 \mod pq.$$

Hence we have now proved that

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

This means that the decryption of the ciphertext will always restore the plaintext.

Just as for the original RSA, the reasoning behind the correctness of the polynomial based RSA follows the same path. In this case we will implement RSA in the polynomial ring $(\mathbb{Z}_p[x], +, \cdot)$ where p is a prime and the operations addition and multiplication are done modulo a polynomial. We have to show here that for any plain-text $M(x)$ belonging to $R = \mathbb{Z}_p[x] / \langle N(x) \rangle$, it is true that

$$(M(x)^e)^d \equiv M(x) \mod N(x).$$

That is the dth power of the ciphertext $C(x)$, which is in turn equal to the eth power of the message $M(x)$ should restore the message $M(x)$ that was sent.

When calculating the decryption exponent d, we solve the congruence $ed \equiv 1 \bmod s$, where e is the encryption exponent and s denotes as we remember the number of invertible polynomials in R. From this equation we can write the product ed in terms of s in this way $ed = sk + 1$, where k is an integer. As such the decryption process becomes

$$C(x)^d = M(x)^{ed} = M(x)^{sk+1} = M(x)^{sk} M(X) = (M(x)^s)^k M(x) \mod N(x).$$

Since there exists an element $E(x) \in R$ such that for any $A(x) \in R$, $A(x) \cdot E(x) = E(x) \cdot A(x) = A(x)$, namely the constant polynomial $E(X) = 1$, it follows that the polynomial ring is a ring with unity. Theorem 3.1 tells us that the set of all invertible elements belonging to the ring with unity R together with the operation multiplication modulo a polynomial, form the algebraic structure of a group. That is $(R^*, \cdot)$ is a group, where R^* denotes the set of units. The number of elements in R^* is represented by s, in other words $|R^*| = s$. Using the result of Theorem 3.2 we know that for any element $B(x) \in R^*$

$$B(x)^{|R^*|} = B(x)^s = E(x) = 1.$$

This means that if someone wants to send a message $M(x)$ and this message belongs to R^*(the message $M(x)$ is invertible modulo $N(x)$) then the sth power of it will always be equal to the identity element of the group $(R^*, \cdot)$, which is equal to the constant polynomial $E(x) = 1$. In the decryption process we will have

$$C(x)^d = (M(x)^s)^k M(x) = 1^k M(x) = M(x) \mod N(x).$$

We have now seen that in this case the message that was sent will be restored by the holder of the decryption exponent.

Now, if someone wants to send a message $M(x)$ which does not belong to R^*, but instead it belongs to R, will the receiver be able to restore the message that was sent?

In order to show that the decryption still works we have to prove that for any plaintext $M(x) \in R$ the following congruence holds

$$(M(x)^s)^k M(x) \equiv M(x) \mod N(x).$$

Substituting s with the expression we have earlier found and, and writing the congruence modulo $P(x)$ and $Q(x)$ respectively, we get

$$(M(x)^{(p^m-1)(p^n-1)})^k M(x) \equiv (M(x)^{p^n-1})^{k(p^m-1)} M(x) \mod P(x)$$
$$(M(x)^{(p^m-1)(p^n-1)})^k M(x) \equiv (M(x)^{p^m-1})^{k(p^n-1)} M(x) \mod Q(x).$$

It follows that

$$M(x)^{ed} \equiv 1^{k(p^m-1)} M(x) \equiv M(x) \mod P(x)$$
$$M(x)^{ed} \equiv 1^{k(p^n-1)} M(x) \equiv M(x) \mod Q(x).$$

It results that

$$M^{ed}(x) - M(x) \equiv 0 \mod P(x)$$
$$M^{ed}(x) - M(x) \equiv 0 \mod Q(x).$$

We see that $M(x)^{ed} - M(x)$ is both divisible by $P(x)$ and $Q(x)$. Since $P(x)$ and $Q(x)$ are two irreducible polynomials that are not associated, it follows that $M^{ed}(x) - M(x)$ is also divisible by the product $P(x)Q(x)$

$$M^{ed} - M(x) \equiv 0 \mod P(x)Q(x).$$

Hence, we have now proved that

$$M^{ed} \equiv M(x) \mod N(x).$$

This means that the decryption formula will always restore the plaintext.

Chapter 4

Security

When the topic of any discussion is a certain cryptosystem, the question of security comes naturally, being actually the most important aspect when considering whether one can use it in practice or not.

4.1 Integer factorization

For over 30 years, the RSA cryptosystem was found to be secure, supporting nowadays most of the electronic commercial communications. At the heart of its security are the two mathematical, still unsolved problems, namely efficient factorization of large numbers and the RSA problem.

Integer factorization (**prime factorization**) is the decomposition of a composite number into its prime divisors. For example the prime factorizations of $n = 65$ and $m = 20$ are $n = 5 \cdot 13$ and $m = 2^2 \cdot 5$ respectively. Not all integers of a given length are equally hard to factor. The most difficult to factor are **semiprimes**, the product of two primes, especially when they are large, and about the same size. This is exactly the case here, where the security of RSA is relying on the inability of even the fastest computers to factorize a large semiprime n, representing the product of the two primes p and q.

Definition 4.1. An **algorithm** is a clearly specified set of instructions to be followed to solve a problem.

When an algorithm has been specified for an operation, we can consider the amount of time required to perform this algorithm on a computer. If any algorithm is correct but takes 10 years to solve the problem, is hardly of any use. The main factors that affect the running time of an algorithm are the algorithm used and the input to the algorithm.

Over the past years factoring algorithms were developed, but none of them proved to be efficient enough when used in practice. These algorithms that were invented fall into two classes: **special-purpose** factoring algorithms and **general purpose** ones.

4.1.1 Special-purpose factoring algorithms

The special-purpose algorithms are suitable for integers with specific types of factors. Therefore the running time of a special-purpose algorithm depends on the size of the number n being factored as well as on the properties of the factors of n. None of these is useful to factor composites used in cryptosystems. In order to understand how and when this kind of algorithms can be used we will briefly discuss one of the following such special-purpose factoring algorithms:

1. Trial division
2. Pollard's ρ algorithm
3. Pollard's $p - 1$ algorithm
4. The elliptic curve method
5. Fermat's factorization algorithm

6. Euler's factorization algorithm

Fermat's factorization algorithm can be used for a composite integer n whose factors are close to each other. The approach is trying to find two integers a, b such that their squares are congruent modulo n. In other words, it finds a, b with

$$a^2 \equiv b^2 \mod n.$$

The congruence is equivalent to saying that $n|(a^2 - b^2)$. By replacing $a^2 - b^2 = (a - b)(a + b)$, we have that $n|(a - b)(a + b)$. The right a and b can be found by trying each of the following values for $a = [\sqrt{n}] + 1, [\sqrt{n}] + 2$, until we get $a^2 - n$ to be a perfect square. Then $b^2 = a^2 - n$. Then by computing $\gcd(a + b, n)$ and $\gcd(a - b, n)$ we might get the non-trivial factors of the given integer n.

Example 4.1. Find the factors of the integer 323 using Fermat's method.

Solution: We begin by trying to find the two integers a, b such that

$$a^2 \equiv b^2 \mod n.$$

The first try for a is $a = \lfloor\sqrt{323}\rfloor + 1 = 17 + 1 = 18$. We check if $b^2 = a^2 - n$ is a perfect square. By replacing the corresponding values we get $b^2 = 18^2 - 323 = 324 - 323 = 1$, which is what we hoped for. Hence we have found $a = 18$ and $b = 1$. We now compute $\gcd(a + b, n)$ and $\gcd(a - b, n)$. When substituting the values for a, b and n, we find $\gcd(18+1, 323) = 19$ and $\gcd(18-1, 323) = 17$, which being non-trivial are in fact the actual factors of the given integer n. Hence, by using Fermat's factorization method we have found the factors of $n = 323$, namely $323 = 17 \cdot 19$.

4.1.2 General-purpose factoring algorithms

The running time of a general-purpose factoring algorithm depends only on the size of the number n to be factored. The algorithms belonging to this group are used in cryptography. Some of the most important to mention are:

1. Dixon's algorithm
2. Quadratic sieve
3. General number field sieve.

Since they are of great interest right here in our discussion about the RSA security, we will take a closer look at one of them in what follows.

Definition 4.2. A positive integer is called **B-smooth** if none of its prime factors is greater than B.

Example 4.2. Show that the integer 150 is 5-smooth.

Solution: In order to show that the given integer is 5-smooth, first of all we have to write its prime factorization. We have $150 = 2 \cdot 3 \cdot 5^2$. We see that none of the integer's prime factors is greater than 5, therefore we conclude that 150 is 5-smooth.

Dixon's algorithm

Let n be the integer to be factored. Dixon's factorization approach tries to find two elements a and $b \in \mathbb{Z}_n$, such that their squares are congruent modulo n:

$$a^2 \equiv b^2 \mod n.$$

In other words, it looks for elements $a, b \in \mathbb{Z}_n$ such that $a^2 - b^2$ (this being in turn equal to $(a-b)(a+b)$) is a multiple of n. Therefore, n will divide $(a - b)(a + b)$, from which it follows that n divides $\gcd(a - b, n) \cdot \gcd(a + b, n)$.

Now, going back to finding congruences of squares Dixon's algorithm proposes to pick some random v, for which s_v, the remainder of v^2 modulo n, is smooth for a bound B. The algorithm continues until it has found sufficiently many different pairs v, s_v for which s_v is smooth.

Example 4.3. Find the factors of the composite integer 323, using Dixon's algorithm.

Solution: We will use the smoothness bound $B = 5$ (that is s_v is 5-smooth meaning that s_v can be factored using only the primes 2,3 and 5).

We begin by selecting v at random. Let $v = 26$. We find that

$$s_v = v^2 = 26^2 = 676 \equiv 30 \mod 323.$$

Since $30 = 2 \cdot 3 \cdot 5$, we find s_{26} to be smooth. Hence, we will keep the pair $(26, s_{26})$ and the identity

$$26^2 \equiv 2^1 \cdot 3^1 \cdot 5^1 \mod 323.$$

Next, we randomly choose $v = 32$. We have that

$$s_{32} = 32^2 = 378 \equiv 55 \mod 323.$$

Since $55 = 5 \cdot 11$, and thus it is not 5-smooth, it is not of further interest.

Proceeding to the next random v, say $v = 33$, we get that

$$s_{33} = 33^2 = 1089 \equiv 120 \mod 323.$$

Since $120 = 2^3 \cdot 3 \cdot 5$, we find s_{33} to be 5-smooth. Hence, we will keep the pair $(33, s_{33})$ and the identity

$$33^2 \equiv 2^3 \cdot 3^1 \cdot 5^1 \mod 323.$$

Having the two identity relations, we can now try to find the factors of the given integer. If we are lucky, the two relation may suffice in achieving our goal to factorize 323. As we remember we found the following two relations:

$$26^2 \equiv 2^1 \cdot 3^1 \cdot 5^1 \mod 323$$
$$33^2 \equiv 2^3 \cdot 3^1 \cdot 5^1 \mod 323.$$

By multiplying these two identity relations we get

$$(26 \cdot 33)^2 \equiv 2^4 \cdot 3^2 \cdot 5^2 \mod 323.$$

As we observe on the left hand side of the congruence, we have a perfect square, namely the perfect square of $26 \cdot 33$, and on the right hand side we also have a perfect square, namely the perfect square of $2^2 \cdot 3 \cdot 5$. Hence we have found some a and b for which $a^2 \equiv b^2 \mod n$, as it was specified in the description of the algorithm. Now, when computing the $\gcd(a - b, n)$ and $\gcd(a + b, n)$, we might find the factors of n. In this case we have found $a = 26 \cdot 33 = 858$ and $b = 2^2 \cdot 3 \cdot 5 = 60$. When computing the respective greatest common divisors we get that $\gcd(858 + 60, 323) = 17$ and $\gcd(858 - 60, 323) = 19$. Since both are non-trivial divisors, we have indeed found the factorization of the given integer, namely $323 = 17 \cdot 19$.

4.2 The RSA problem

First of all let us recall that in the RSA public-key cryptosystem the pair (n, e) represents the public key, whereas the triple (p, q, d) represents the private key. When encrypting a message m, one performs $m^e \mod n$, which will be the corresponding ciphertext c. When decrypting, the holder of the private key, more specifically the holder of the private exponent d, will be able to perform $c^d \mod n$, restoring thus the message m that was sent.

The **RSA problem** is finding the plaintext m given the ciphertext c and the public key (n, e). More precisely, it requires finding the e^{th} root of an arbitrary ciphertext c modulo n (that is find $\sqrt[e]{c} \mod n$).

Just as there is no firm mathematical ground on which the assumption that factoring a composite integer is a hard problem, neither is there an evidence showing that the RSA problem is intractable. Still, whether solving the RSA problem is equivalent, as for complexity, to factoring the integer n remains an open question.

4.3 Polynomial factorization

Factoring an univariate polynomial (that is, a polynomial in a single variable) refers to writing it as a product of irreducible polynomials. Since the polynomial version of RSA is implemented in the polynomial ring $\mathbb{Z}_p[x] = \{a_0 + a_1x + \cdots + a_kx^k | k \geq 0, a_i \in \mathbb{Z}_p\}$, in this section we will look at the factorization of univariate polynomials over the finite field $\mathbb{Z}_p$.

As the security of the RSA public key cryptosystem is based on our failure to find an efficient algorithm for integer factorization, the security of the polynomial RSA, proposed here, is based on the grade of difficulty to factorize polynomials over finite fields.

The time complexity of an algorithm is estimated by counting the number of operations performed by the algorithm. It is usually expressed using the big O notation, which excludes coefficients and lower order terms. In the 60's, Berlekamp described a probabilistic algorithm which factorizes univariate polynomials with coefficients in a finite field, say $\mathbb{F}_q$. Its complexity is polynomial in the size of the input polynomial, namely $O(d^2 \log q)$, where d is an integer such that the degree of the input polynomial is smaller than d. In 1995, Kaltofen and Shoup , have come up with a better complexity bound for the polynomial factorization with coefficients in the finite field $\mathbb{F}_q$, namely $O(d^{1.815} \log q)$. Two years later, they have given a lower bound of order $O(d \log^2 q)$. For further details on the algorithms mentioned above and their time complexities see [2].

In order to have an idea on how such a polynomial factoring algorithm works, we will present the "basic" algorithm for solving the problem, which dates back in 1967 [23].

4.3.1 Berlekamp's algorithm

Let $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_mx^m$ be a polynomial of degree m, with coefficients over the finite field $\mathbb{Z}_p$, where p is a prime. The algorithm takes as input a *monic* (the leading coefficient is equal to one, that is $a_m = 1$), *squarefree polynomial* (that is a polynomial with no repeated factors). Let $f_1, f_2, \ldots, f_r \in \mathbb{Z}_p[x]$ be the irreducible monic factors that we shall compute. The algorithm uses basic polynomial operations such as products, divisions, greatest common divisor calculations, powers of one polynomial modulo another.

Theorem 4.1. *Let $f \in \mathbb{F}_q[x]$ be a monic polynomial and $h \in \mathbb{F}_q[x]$ is such that $h^q \equiv h \bmod f$. Then the following holds*

$$f(x) = \prod_{c \in \mathbb{F}_q} \gcd(f(x), h(x) - c).$$

Theorem 4.1 gives a way to calculate the factors of $f(x) \in \mathbb{F}_q[x]$, by computing q greatest common divisors. Finding a polynomial $h(x)$ such that $h(x)^q \equiv h(x) \bmod g = f(x)$ is done by reducing this congruence to a system of linear equations. Let the degree of $f(x)$ be equal to n. Then we construct the matrix $B_{n \times n}$ by calculating the powers $x^{iq} \bmod f(x)$ with $0 \leq i \leq n-1$. Each row entry in the matrix corresponds to the coefficients of the resulting polynomials when computing $x^{iq} \bmod f(x)$. Finding the vectors that form the null space of the matrix $B - I$, where I is the identity matrix, is the way to find such polynomials $h(x)$ satisfying $h(x)^q \equiv h(x) \bmod f(x)$. The solutions of the equation $(B - I)X = 0$ correspond therefore to such polynomials $h(x)$. Then, by computing the $\gcd(f(x), h(x) - c)$ for all $c \in \mathbb{F}_q[x]$, we will have found the factors of $f(x)$. We shall see this in the following example, where the Berlekamp's method will be presented step by step. Before that we will need the following theory.

Definition 4.3. The **rank** of a matrix is defined as the maximum number of linearly independent column vectors in the matrix (or the maximum number of linearly independent row vectors in the matrix).

Example 4.4. Find the factorization of $f(x) = x^3 + 1$ over $\mathbb{Z}_2$ using Berlekamp's algorithm.

Solution: First of all we check whether the given polynomial is squarefree. We do this by computing the greatest common divisor between f and it derivative by using the same algorithm as we do for integer gcd's calculations. We see that

$$\gcd(x^3 + 1, 3x^2) = 1.$$

The result means that there are actually no repeated factors in f. The next step is to compute the powers $x^{2i} \bmod f(x)$ for $0 \leq i \leq 2$. We have that

$$x^0 \equiv 1 \mod f(x)$$

$$x^2 \equiv x^2 \mod f(x)$$
$$x^4 \equiv x \mod f(x).$$

We now form the matrix B of order 3×3, with each row entry representing the coefficients of the resulting polynomials in the congruences above:

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

We compute then the difference $B - I$, where I represents the identity matrix

$$B - I = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

We now subtract the second row from the third one and the matrix becomes equivalent to

$$B - I \sim \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

We see that there are 2 free variables, namely x_1 and x_3. Since there is one non-zero row, it follows that the rank of the matrix is equal to 1. It results that f has $3 - 1$ irreducible factors. The next step is finding the vectors that form the null space of the matrix $B - I$. That is, we have to solve the equation

$$(B - I) \cdot X = 0.$$

More explicitly, we have to solve

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

It leads to solving the following system

$$0 \cdot x_1 + 0 \cdot x_2 + 0 \cdot x_3 = 0$$
$$0 \cdot x_1 + x_2 + x_3 = 0$$
$$0 \cdot x_1 + 0 \cdot x_2 + 0 \cdot x_3 = 0.$$

From the second equation we find $x_2 = -x_3$. We can now write the solution as:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = x_1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + x_3 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}.$$

Hence, the two vectors (1,0,0) and (0,1,1) form the basis for the null space of $B - I$. The corresponding vectors are

$$h_1(x) = 1$$
$$h_2(x) = x + x^2.$$

The next step is to make use of Theorem 4.1 and compute

$$\gcd(f(x), h_2(x) - 0) = \gcd(x^3 + 1, x^2 + x) = x + 1$$
$$\gcd(f(x), h_2(x) - 1) = \gcd(x^3 + 1, x^2 + x + 1) = x^2 + x + 1.$$

Since we have seen that f has two irreducible factors, we can now conclude that we have found the factorization of f, which is

$$f(x) = (x + 1)(x^2 + x + 1).$$

As we have understood the Berlekamp's algorithm finds the factors of a polynomial $f(x) \in \mathbb{F}_q[x]$ by computing q greatest common divisors. Well, that is fine as long as the field we are working in, is small, having thus few values c could be. If q is large, that is the basis field is large, we would be forced to do a lot of computation. To avoid computing all possible greatest common divisors, the **Zassenhaus algorithm** is used to characterize the elements $c \in \mathbb{F}_q$ for which $\gcd(f(x), h(x) - c) \neq 1$. For a description of the Zassenhaus algorithm see [11].

Chapter 5

Discussions

We have now reached the point where, after analyzing the most important aspects regarding the original RSA cryptosystem as well as the polynomial version of it, we shall refresh all we have learnt until now, and try to make a comparison between them both, displaying advantages and disadvantages.

In the original RSA public key cryptosystem we start with the ring of integers $\mathbb{Z}$ (which as we earlier saw it is an integral domain). We then choose two primes p and q in this ring, and move on to the ring $\mathbb{Z}_n$, where $n = pq$. Now, the ring $\mathbb{Z}_n$ can be seen as the factor ring of $\mathbb{Z}$ modulo the principal ideal $\langle n \rangle = n\mathbb{Z}$ of all multiples of n. In the same way, in the polynomial version of the RSA public key cryptosystem, we start with the polynomial ring $\mathbb{Z}_p[x]$, which is also an integral domain. We then choose two irreducible polynomials $P(x)$ and $Q(x)$ in this ring, and move on to the factor ring of $\mathbb{Z}_p[x]$ modulo the principal ideal $\langle N(x) \rangle$ of all multiples of $N(x) = P(x)Q(x)$. The encryption and decryption for the original RSA occur in $\mathbb{Z}_n$ (the factor ring $\mathbb{Z}/n\mathbb{Z}$) and so similarly the encryption and decryption for the polynomial RSA occur in the factor ring $\mathbb{Z}_p[x] / \langle N(x) \rangle$.

As we have experienced, adding and multiplying integers modulo an integer, as in the case of the original RSA implementation, is easier than adding and multiplying polynomials, modulo a polynomial, as it is in the case of the polynomial RSA. As such even the encryption and decryption processes, where the modular exponentiation computations are done, become easier and faster in the case of integers.

Regarding the most important aspect when considering any cryptosystem, namely its security, we recall that the problem of polynomial factorization over finite fields was successfully studied, and with today's algorithms and computers we can factor large polynomials. On the other side, the problem of integer factorization is still unsolved. We have not yet discovered an algorithm that efficiently finds the factors of an integer, especially of a semiprime. Since integer factorization and polynomial factorization stand behind the security of the cryptosystem, we can, from this point, conclude that the polynomial version of RSA cannot be used in real life, since it would be insecure.

Since implementing the polynomial RSA in the polynomial ring $\mathbb{Z}_p[x]$ proved not to be effective regarding the security, an interesting idea would be that instead of using a finite base field (as in this case $\mathbb{Z}_p$), we can replace it by an infinite field such as $\mathbb{Q}$. Considering this idea, the problem of security will be equivalent to the problem of factorizing polynomials over infinite fields. Given a nonzero polynomial $f \in \mathbb{Q}[x]$ in one variable with rational coefficients, a polynomial-time algorithm to solve the problem is given by Lenstra et. al. [21].

Once again, the simplicity and elegance which describes the RSA cryptosystem proves to pass the test of time, winning over the more complicated version of the same idea.

Chapter 6

Application

In order to have a better insight into how the polynomial version of the RSA public key cryptosystem works, we will use an interactive computer application to encrypt and decrypt at will. The application is implemented in Java, and we will see here how and under which circumstances can be used.

6.1 Encoding

To begin with, in this project we implement RSA in the polynomial ring $\mathbb{Z}_{29}[x]$. That is, when encoding the plaintext, the corresponding polynomial will be a member of the polynomial ring

$$\mathbb{Z}_{29}[x] = \left\{a_0 + a_1x + a_2x^2 + \cdots + a_kx^k | k \geq 0, a_i \in \mathbb{Z}_{29}\right\}.$$

The public key is represented by the pair $(N(x), e)$, and the private key is represented by the triple $(P(x), Q(x), d)$. It will be possible to encode all the letters in the Swedish alphabet, which contains 29 letters. That being said, we see that this is the reason behind the choice of encoding the plaintext into polynomials with coefficients belonging to $\mathbb{Z}_{29}$.

The plaintext will be encoded as a sequence of polynomials of degree smaller than the degree of $N(x)$, $N(x)$ representing the product of the two irreducible polynomials $P(x)$ and $Q(x)$. For this specific application the two polynomials $P(x)$ and $Q(x)$ will have degree at most three.

Each letter will be encoded into its corresponding position in the alphabet, starting with the letter **a** having position **zero**, and the first letter of the plaintext will be the coefficient of the highest degree term in the polynomial.

For example, if we are using two irreducible polynomials of degree two, say $P(x) = ax^2 + bx + c$ and $Q(x) = dx^2 + ex + f$, having all coefficients in $\mathbb{Z}_{29}$, $N(x) = P(x)Q(x)$ is a polynomial of degree four. It follows that for encoding the plaintext *hello*, blocks of four letters will be encoded at a time. If we cannot make all blocks of size four letters, then we add extra $x's$ until we can encode.

The plaintext *hello* will be encoded as a sequence of two polynomials, namely one corresponding to the first four letters in the plaintext $M_1(x) = 7x^3+4x^2+11x+11$, and the other one corresponding to the next letter *o*, to which we add three of x to make it complete, having $M_2(x) = 14x^3 + 23x^2 + 23x + 23$. Then, we perform the modular exponentiation $M_1(x)^e \bmod N(x) = C_1(x)$, and $M_2(x)^e \bmod N(x) = C_2(x)$, obtaining the two polynomials, representing the ciphertext, whose coefficients represent in order the position in the alphabet of each letter of the plaintext.

6.2 Choosing the irreducible polynomials

As we know, in order to start building the coding-scheme, the first step is to choose the polynomials $P(x)$ and $Q(x)$. The user of the application can choose his own polynomials, or he can simply push the *Generate* button leaving this job for the computer.

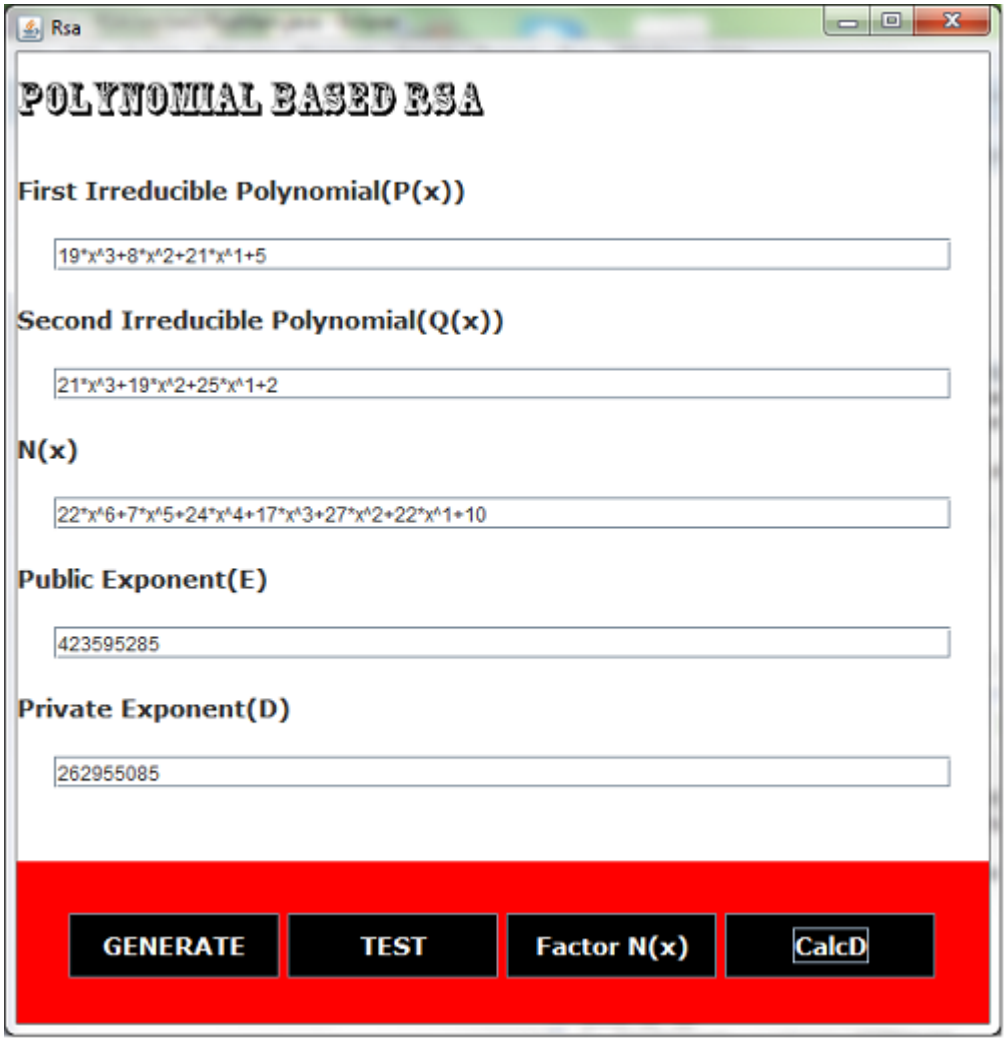
One thing to mention is that when writing your own polynomials, one has to write explicitly every coefficient and every power of each term (i.e. one has to write $1 * x^2 + 5 * x^1 + 7$, even though on paper we do not write coefficients and powers of one).

6.3 Factoring polynomials

Since the application accepts polynomials $P(x)$ and $Q(x)$ of degree at most three, it will only be possible to factor a polynomial $N(x)$ of degree at most six. What we have to do in order to factor a polynomial is simply to type it in the box underneath $N(x)$. The polynomial will be factored, and the corresponding factors will be put in the corresponding boxes (i.e. underneath the labels $P(x)$ and $Q(x)$). Then one can proceed to calculating the public and private exponent and then to encrypting and decrypting messages.

6.4 Encrypting

After having chosen the polynomials $P(x)$ and $Q(x)$, the next thing to do is to push the button *CalcD*. If the two polynomials are not irreducible then the computer will ask you to choose another polynomial(s). After the button is pushed, $N(x)$ will automatically be calculated, as well as the encryption and decryption exponents. The encryption exponent e will be randomly chosen from the set $\mathbb{Z}_s = \{0, 1 \ldots, s-1\}$, such that it fulfills the condition $\gcd(e, s) = 1$ (as we shall recall s denotes the number of invertible polynomials modulo $N(x)$ in $\mathbb{Z}_p[x] / \langle N(x) \rangle$). The decryption exponent will then be found by computing the multiplicative inverse of e modulo s. After this, the button *Test* shall be pushed in order to start encrypting. An encryption window will appear, where we will type the plaintext to be encrypted, and the corresponding ciphertext will be shown underneath. In order to test that the encryption and decryption processes worked as they should, we can then push the button *Decrypt* and restore the plaintext. An execution of the application in three steps, namely, generating the polynomials $P(x)$ and $Q(x)$ and computing the exponents, encryption of a message and decryption of the message that was encrypted, look like this :


Rsa
POLYNOMIAL BASED RSA
First Irreducible Polynomial(P(x))
19*x^3+8*x^2+21*x^1+5
Second Irreducible Polynomial(Q(x))
21*x^3+19*x^2+25*x^1+2
N(x)
22*x^6+7*x^5+24*x^4+17*x^3+27*x^2+22*x^1+10
Public Exponent(E)
423595285
Private Exponent(D)
262955085
GENERATE
TEST
Factor N(x)
CalcD


Encrypt
Message to encrypt:
izabela
Ciphertext:
nzrpcvpsjkoh
ENCRYPT
DECRYPT


Encrypt
Message to encrypt:
izabela
Ciphertext:
izabelaxxxxx
ENCRYPT
DECRYPT

Appendices

.1 Encryption

```
if(e.getSource().equals(encrypt)){
 int splitBy=polynomialP.getDegree()+
 polynomialQ.getDegree();
 //the length of each block of letters
 message=plaintextField.getText();
 //if the length of the message is
 //odd then we add an extra letter to
 //the plaintext
 // a letter which doesn't make sense
 //in the context, namely "x";
 //if the  message cannot be split into equal bloks
 while(message.length()%splitBy!=0){
        message+="x";
  }

 strings = new ArrayList<String>();
 int index = 0;
 while (index < message.length()) {
        strings.add(message.substring(index,
        Math.min(index+splitBy,message.length())));
        index += splitBy;
                }
 coeffPlaintext=new int[splitBy];
 coeffCiphertext=new int[splitBy];
 Poly coeffSplit=new Poly();
 Poly ciphertext=new Poly();
 String str="";
 for(int j=0;j<strings.size();j++){
  for(int i=0;i<splitBy;i++)
         coeffPlaintext[i]=alphabet.indexOf(
         strings.get(j).charAt(i));
         coeffSplit=new Poly(coeffPlaintext);
         ciphertext=power1(coeffSplit,
         encryptExp,polynomialN);
         coeffCiphertext=ciphertext.getCoeff();

                        for(int m=0;m<splitBy;m++)
                                str+=alphabet.charAt(
                                coeffCiphertext[m]);
        }

 ciphertextField.setText(str);

}
```

.2 Decryption

```
else if(e.getSource().equals(decrypt)){
 code=ciphertextField.getText();
 int splitBy=polynomialP.getDegree()+
 polynomialQ.getDegree();
 strings1 = new ArrayList<String>();
 int index = 0;
 while (index <code.length()) {
         strings1.add(code.substring(index,
         Math.min(index+splitBy,code.length())));
         index +=splitBy;
         }
coeffPlaintext1=new int[splitBy];
coeffCiphertext1=new int[splitBy];
Poly coeffSplit=new Poly();
Poly plaintext=new Poly();
String str="";
for(int j=0;j<strings1.size();j++){
         for(int i=0;i<splitBy;i++)
                 coeffCiphertext1[i]=alphabet.indexOf(
                 strings1.get(j).charAt(i));

                 coeffSplit=new Poly(coeffCiphertext1);
                 plaintext=power1(coeffSplit,decryptExp,
                 polynomialN);
                 coeffPlaintext1=plaintext.getCoeff();
    for(int m=0;m<splitBy;m++)
                  str+=alphabet.charAt(coeffPlaintext1[m]);
 }
 ciphertextField.setText(str);

}
```

Bibliography

[1] Manindra Agrawal, Lecture notes on Dixon's Algorithm for Factoring Integers, http://www.cse.iitk.ac.in/users/manindra/CS681/2005/Lecture21.pdf, 15.07.2014

[2] Ali Ayad, A lecture on the Complexity of Factoring Polynomials over Global Fields, 2010, http://www.m-hikari.com/imf-2010/9-12-2010/ayadIMF9-12-2010.pdf, 20.01.2015

[3] Majid Bakhtiari and Mohd Aizaini Maarof, Serious Security Weakness in RSA Cryptosystem, http://ijcsi.org/papers/IJCSI-9-1-3-175-178.pdf, 18.07.2014

[4] Kostas Bimpikis and Ragesh Jaiswal, Modern Factoring Algorithms, http://www.cs.columbia.edu/~rjaiswal/factoring-survey.pdf, 15.07.2014

[5] Dan Boneh, Twenty Years of Attacks on the RSA Cryptosystem, http://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf, 18.07.2014

[6] Pete L. Clark, Arithemtical Functions I: Multiplicative Functions, http://math.uga.edu/~pete/4400arithmetic.pdf, 26.06.2014

[7] Dr. Marcel B. Finan, Lecture notes on Elementary Modern Algebra, http://faculty.atu.edu/mfinan/4033/absalg12.pdf, 23.06.2014

[8] John B. Fraleigh, A First Course in Abstract Algebra, 7th Edition, Pearson, 2002

[9] Shuhong Gao and Daniel Panario, Tests and Constructions of Irreducible Polynomials over Finite Fields, http://www.math.clemson.edu/~sgao/papers/GP97a.pdf, 30.06.2014

[10] Joachim Von Zur Gathen and Daniel Panario, Factoring Polynomials Over Finite Fields: A Survey, http://people.csail.mit.edu/dmoshkov/courses/codes/poly-factorization.pdf, 20.07.2014

[11] Sajid Hanif and Muhammad Imran, Factorization Algorithms for Polynomials over Finite Fields, Linnaeus University, 2011

[12] Thomas W. Hungerford, Abstract Algebra An Introduction third edition, Cengage Learning, 2013

[13] Avi Kak, Public-Key Cryptography and the RSA Algorithm https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture12.pdf

[14] Burt Kaliski, The Mathematics of the RSA Public-Key Cryptosystem,http://www.mathaware.org/mam/06/Kaliski.pdf, 10.06.2014

[15] Piyush P. Kurur, Cantor-Zassenhaus Algorithm, http://www.cmi.ac.in/~ramprasad/lecturenotes/comp_numb_theory/lecture11.pdf, 20.07.2014

[16] Lecture notes, https://www.math.okstate.edu/~binegar/3613/3613-l07.pdf

[17] Lecture notes on The Miller-Rabin Randomized Primality Test, http://www.cs.cornell.edu/courses/cs482/2008sp/handouts/mrpt.pdf, 21.07.2014

[18] Lecture notes (Theorem 9.16), https://www.math.okstate.edu/~binegar/3613/3613-l19.pdf, 22.07.2014

[19] Lecture notes, http://www.math.hawaii.edu/~lee/courses/fundamental.pdf, 25.06.2014

[20] Arjen K. Lenstra, Integer Factoring, http://modular.math.washington.edu/edu/124/misc/arjen_lenstra_factoring.pdf, 15.07.2014

[21] A. K. Lenstra, H. W. Lenstra and L. Lovasz, Factoring polynomials with rational coefficients, http://www.math.elte.hu/~lovasz/scans/lll.pdf, 29.08.2014

[22] A. Menezes, P. van Oorschot and S. Vanstone, Handbook of Applied Cryptography, http://cacr.uwaterloo.ca/hac/about/chap8.pdf, 22.06.2014

[23] Robert J. McEliece, Factorization of Polynomials over Finite Fields, http://www.ams.org/journals/mcom/1969-23-108/S0025-5718-1969-0257039-X/S0025-5718-1969-0257039-X.pdf, 22.07.2014

[24] Robert J. McEliece, Finite Fields for Computer Scientists and Engineers, Springer, 1987, 22.07.2014

[25] Daniel Panario, Boris Pittel, Bruce Richmond and Alfredo Viola, Analysis of Rabin's irreducibility test for polynomials over finite fields, http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0116.pdf

[26] R.L. Rivest, A. Shamir, and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, http://people.csail.mit.edu/rivest/Rsapaper.pdf, 10.06.2014

[27] Kenneth H. Rosen, Elementary Number Theory and Its Applications (5th International Edition), Addison Wesley, 2005

[28] David Seal, Lecture notes,http://www.math.msu.edu/~seal/teaching/sp12/4-4.8.pdf, 27.07.2014

[29] Dr. David Singer and Mr Ari Singer, The Role Played by Mathematics in Internet Commerce, http://www.case.edu/affil/sigmaxi/files/CryptoslidesSinger.pdf, 28.07.2014

[30] Peter W. Shor, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, http://arxiv.org/pdf/quant-ph/9508027v2.pdf, 15.07.2014

[31] Barry Steyn, Why RSA Works: Three Fundamental Questions Answered, http://doctrina.org/Why-RSA-Works-Three-Fundamental-Questions-Answered.html, 29.06.2014

[32] Per-Anders Svensson, Lecture notes on Algebraic Structures, Fall 2013

[33] Lerk R. Vermani, Elements of Algebraic Coding Theory, Chapman Hall, 1996

[34] Dr Francis J. Wright, Mathematics and Algorithms for Computer Algebra, http://www.cs.berkeley.edu/~fateman/282/F\%20Wright\%20notes/week6.pdf, 1.07.2014

Lnu.se

Linnæus University
Sweden

Faculty of Technology
SE-391 82 Kalmar | SE-351 95 Växjö
Phone +46 (0)772-28 80 00
teknik@lnu.se
Lnu.se/faculty-of-technology?l=en