LEBANESE AMERICAN UNIVERSITY

SCHOOL OF ARTS AND SCIENCES

DIVISION OF COMPUTER SCIENCE
AND MATHEMATICS

Testing and Evaluating Digital Signature Algorithms in Principal Ideal Domains

by
Bilal Shebaro

Thesis Advisor: **Dr. Ramzi A. Haraty**

November 2005

Testing and Evaluating Digital Signature Algorithms in Principal Ideal Domains

by
Bilal M. Shebaro

Submitted in Partial Fulfillment of the Requirements
for the Degree of Masters in Computer Science

Thesis Advisor: **Dr. Ramzi A. Haraty**

November 2005

Division of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon

LEBANESE AMERICAN UNIVERSITY

School of Arts and Sciences - Beirut Campus

Thesis approval Form (Annex III)

Student Name:	Bilal Shebaro	I.D. #:	199906450
Thesis Title :	Testing & Evaluating Digital Signature Algorithms in Principal Ideal Domains		
Program :	M.S.		
Division/Dept :	Computer Science & Mathematics		
School :	**School of Arts and Sciences**		
Approved by:	Ramzi A. Haraty		
Thesis Advisor:	[redacted]		
Member :	Sanaa Sharafeddine	[redacted]	
Member :	Ahmad Kabbani	[redacted]	
Member :			
Date	25/11/2005	[redacted]	

(This document will constitute the first page of the Thesis)

To be signed by the student:

Bilal Shebaro [redacted]

ABSTRACT

November 2005, Testing and Evaluating Digital Signature Algorithms in Principal Ideal Domains

Digital signature is a mechanism designed to allow secure communication through an unsecure medium and can be traced in many applications where privacy is required. The main purpose of this thesis is to extend some important and useful digital signature schemes from the domain of natural integers **Z** to two principal ideal domains namely the domain of Gaussian integers $\mathbf{Z}[i]$ and the domain of the ring of polynomials over finite fields $F[x]$. This is accomplished by extending arithmetic needed for our extensions to these domains. Each of RSA signature and ElGamal signature schemes is extended to both the domain of Gaussian integers and the domain of polynomials over finite fields. A description of these extensions is given along with algorithms, proofs, numerical examples and computer programs. Moreover, we compare and evaluate the classical and modified ElGamal and RSA algorithms by implementing and running them on a computer. We investigate the issues of complexity, efficiency and reliability by running the programs with different sets of data. Comparisons will be done among these different algorithms given the same data as input. In addition, implementation of an attack algorithm will be presented. The attack algorithm consists of subroutines used to crack signed messages. This is done by applying certain mathematical concepts to find the private key. After finding the key, it will be easy to forge the signature. A study will be done using the results of running the attack algorithm to compare the security of the different classical and modified digital signature algorithms. Finally,

some of the advantages are pointed out.

The thesis consists of the following five chapters.

In chapter one, an introductory material to digital signatures with the basic notions surrounding it is given as well as a general overview of cryptography and a historical background related to digital signature. in order to motivate the digital signature algorithms.

In chapter two, we introduce mathematical tools and concepts needed for the digital signature schemes and mathematical concepts in the domain of natural integers that are relevant to the digital signature algorithms. A survey of various techniques for classical signature algorithms with the reference to their proofs is also given.

In chapter three, a comparitive study of ElGamal based digital signature algorithm is done. We implement the classical and modified ElGamal digital signature algorithms to compare and test their functionality, reliability and security. To test the security of the algorithms we use a famous attack algorithm called Baby-Step-Giant algorithm which works in the domain of natural integers. We enhance the Baby-Step-Giant algorithm to work with the modified ElGamal digital signature algorithms.

In chapter four, a comparitive study of RSA based digital signature algorithm is done. The classical and modified RSA algorithm is implemented to compare and test their functionality, realiability and security. To test the security of the algorithms we implement an attack algorithm to solve the integer factorization problem in $\mathbf{Z}, \mathbf{Z}[i]$, and $F[x]$. After factorization is found, the RSA problem could be solved by finding the private key using the extended Euclidean algorithm.

We conclude this thesis in chapter five by stating some advantages of the ex-

tended schemes and giving some conclusions. Also, we pose some related problems for future research.

DEDICATION

I have the pleasure to dedicate this thesis to my Mom and Dad who gave me their love and support throughout the years of education at the university.

Great thanks to my brothers and sisters who gave me patience and guidance, and who stood beside me in every good and bad.

Acknowledgement

I have great pleasure to express my sincere gratitude to my advisor Dr. Ramzi A. Haraty for his encouragement, guidance throughout my M.S. studies and his assistance in the improvement of this thesis.

Also I have the honor to express humble gratitude and sincere thanks to Dr. Ahmad K. Kabbani and Dr. Sanaa Sharafeddine for being in my thesis committee. Special thanks to Dr. Adbel Naser Kassar for all his assistance in this work.

I should not forget to thank my parents and friends for giving their sincere assistance.

Most important of all, I thank GOD for letting light dawn in moments of extreme darkness.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Cryptography is defined to be the art and the science of preparing coded or protected communications intended to be intelligible only to the person possessing a key. The word "cryptography" is a combination of two Greek words, kryptos, which means secret, and graphos, which means writing. Hence, cryptography is the process or skill of communicating in secret writings (codes, or ciphers) or deciphering secret writings. Also, cryptography refers to the use of codes to convert computerized data so that only a specific recipient will be able to read it using a key (Encryption). In cryptography we call an original text the real message or plaintext. Once the original text has been encrypted (coded), the result is known as the ciphertext. The encryption process usually involves a particular method to cipher origin texts called "Encryption Algorithm". To recover the original text from the ciphered one, we have to use a specific unique inverse method to that of encryption process called "Decryption Algorithm".

Aciently, cryptography was the only tool that gives the ability to encrypt data, and ensure that these data will be secret. So, cryptography was a secure box that gives individuals the power to restore national and personal privacies. Now, with computers, cryptography is important for more than just privacy, however. Cryptography protects the world's banking systems, commerce, communications over telephone lines, financial transactions, medical histories, choices of rental movies, and others. Hence, with technology, cryptography became one of the most essential part of today's information system.

The most striking development in the history of cryptography came in 1976 when Diffie and Hellman published "New Directions in Cryptography", [3]. This paper introduced the revolution in the concept of public-key cryptography and also provided a new and ingenious method overview of cryptography for key exchange. This method is based on the intractability of discrete logarithm problems. Although the authors had no practical realization of a public-key encryption scheme at the time, the idea was clear and it generated extensive interests and activities in the world of cryptography. In 1978 three American computer scientists, Ronald L. Rivest, Adi Shamir, and Leonard Adleman created the Rivest-Shamir-Adleman (RSA) system. The RSA scheme is based on another hard mathematical problem, the intractability of factoring large integers. This application of a hard mathematical problem to cryptography revitalized efforts to find more efficient methods to factor integers. In the 1980s major advances in this area was seen and the RSA system was widely used as a secure system. Another class of powerful and practical public-key schemes was found by ElGamal in 1985. These are also based on the discrete logarithm problem.

1.1 Digital Signatures Concepts

Digital signatures are strong tools applied in order to achieve the security services of authentication (proof of identity of the sender), data integrity (detection of changes to the message) and non-repudiation (prevention of denial of sending the information). They are digital counterpart of handwritten signatures that can be transmitted over a computer network. Only the sender can make the signature, but other people can easily recognize as belonging to the sender. The sender produce a signature consisting of a number associating a message (in digital form) with a secret key. This signature is intended to be unique and it does not necessarily require that a message be encrypted but must be verifiable. Digital signatures are based on mathematical theory and the use of algorithms. They require that the holder of the signature has a two key system for signing and verifying (one private and the other public). The private key should only be available to the user to whom it belongs, and is used by the signing procedure in order to sign the message. The signing procedure consists of a mathematical digital signature generation algorithm, along with a method for formatting data into messages which can be transformed into a tag called a signature. The public key may be available to many users of the system, and is used by the verification procedure. The verification procedure consists of a method to verify that a digital signature is authentic that's to say it really comes from the claimed sender (assuming only the sender knows the private key corresponding to his public key) along with a method for recovering data from the message in particular types. Digital signatures can also be used to testify that a public key belongs to a particular person or entity. This is done by signing the combination of the public key and the user's identity by a trusted key. The public key and the user's identity are called certificates. The digital signature use the owner of the trusted key to bind the identity of an entity to a public key, so that at some later time, other entities can authenticate a public key without assistance from a trusted third party.

The most popular digital signature algorithms are based on certain mathematical properties of the integers such as divisibility, congruencies, factorization, Euclidean algorithm, discrete logarithms, residue systems, quadratic residues, and others. The domain of integers share many similarities with other algebraic systems. The digital signature algorithms can be modified to any principal ideal domain where the needed properties can be fully described and efficiently implemented. Also, we show that these new settings are carefully chosen so that the operations can be performed efficiently and easily applied.

The digital signature schemes are operations combining primitives and other techniques in cryptographic protocols. RSA signature, ElGamal signature, DSA signature, Rabin signature are example of signature schemes that are in use today. All of them are public-key schemes with secret information to sign documents and public information to verify signatures. There are two main classes of digital signature schemes which are the following:

1. Digital signature schemes with appendix : require the original message as input to the verification algorithm. They are the most commonly used in practice and rely on what we called cryptographic hash functions[24],[25].

2. Digital signature schemes with message recovery : do not require the original message as input to the verification algorithm. In this case the original message is retrieved (if the signature verifies) from the signature itself.

These classes of digital signature schemes are based on one-way functions which we will now discuss. A one-way function is a function that is easy to compute in one direction, but very difficult to compute going the other direction. By "difficult", we mean that it would take millions of years to be computed. There are many functions that seem to be one-way. For example, computing square roots in a finite field, factoring in a finite field and calculating discrete logs in a finite field.

1.2 History of Digital Signatures

The concept of a digital signature was introduced in 1976 by Diffie and Hellman. They published their landmark paper "New Directions in Cryptography" [3]. This paper introduced the revolutionary concept of public-key cryptography, provided a new and ingenious method for key exchange and outlined how the difficult problem of solving discrete logarithms in finite fields could be used to develop public-key cryptography which had clear potential for use in data networks. Diffie and Hellman suggested, quite prophetically, that the "one-way authentication" services offered by public-key schemes would ultimately be of more importance to the business community than the confidentiality services for which cryptography had traditionally been used. In 1978, three american scientists Ron Rivest, Adi Shamir and Leonard Adleman invented the first practical and most successful public-key scheme, referred as RSA. This scheme is based on the intractability of factoring large integers. RSA cryptosystem allowed both encryption and the application of digital signatures. Other powerful and practical public-key schemes soon followed including ElGamal technique in 1985 that is based on the discrete logarithm problem. In 1991, the ISO IEC 9796 "Information technology security techniques digital signature scheme giving message recovery" was published by the International Standards Organization as the first international standard for digital signatures. It specifies a digital signature process (as the RSA for digital signatures & others). In 1994, a mechanism based on ElGamal public-key scheme was developed by the U.S. Government for the Digital Signature Standard. This mechanism is then modified with the emphasis on RSA digital signatures to support the de facto standard of financial institutions.

In 2001, ElKassar and Awad implemented El-Gamal public key cryptosystem in the domain of gaussian integers[10]. In 2002, an investigation and comparison between ElGamal public-key cryptosystem and its extension to the gaussian integers domain was done by El-Kassar, see[28].

In 2003, the classical ElGamal cryptosystem and four modifications to it, namely, the ElGamal cryptosystem in Z_n, were presented in the domain of Gaussian integers, $Z[i]$, over finite fields, and over quotient rings of polynomials over finite fields,[16]. The algorithms were implemented and their efficiency, reliability, and security were tested by Haraty and El-Kassar [16]. Attack scenarios that directly aimed at solving the discrete logarithm problem were built for these algorithms to utilize. Recently in 2004, ElGamal public-key cryptosystem using reducible polynomials over a finite field was presented by ElKassar and Haraty [7]. Also in the same year, they presented the classical RSA cryptosystem and two of its modifications, namely, the RSA cryptosystem in the domain of Gaussian integers, $Z[i]$, and over quotient rings of polynomials over finite fields[18]. They also built attack scenarios directly aimed at solving the factorization problem. They observed that the Gaussian method is preferred since it is as secure as the classical one but provides an extension to the message space and to the encryption exponent range.

Cryptographic Hash Functions A cryptographic hash function or simply hash function h is a mathematical function applied to a message. It maps binary input m in binary strings of arbitrary length that will represent the whole message (large strings) to binary output strings of some fixed bit length say n bits. This output which is denoted by $h(m)$ is called the hash value or message digest (generally smaller). By this operation, the message to be signed is reduced to its hash value prior to signature generation. Not the message itself but its hash value is signed, and this represents the digital signature. When the data is received and the hash value is computed, this should match the included hash value. Cryptographic hash functions are used in various contexts, but their main role is in the provision of digital signatures for several reasons:

1. Public-key signing algorithms tend to be very slow, so taking the hash value of a large message and signing the small resulting value is much more efficient than simply signing the whole message.

2. If you want to add multiple signatures to a message without the use of hash functions, the signed message would be many times the size of the original.

Choosing a good hash function is of most importance. There are basic requirements we have for a cryptographic hash function to use in the digital signature schemes to be one-way and collision free. The concept of one-wayness with respect to hash functions is that it is hard to invert, it means that, given a hash value H, it is computationally infeasible to find some input m such that $h(m) = H$. A one-way hash function converts an arbitrary-length message into a fixed-length hash. A strongly collision-free hash function h is one for which it is computationally infeasible to find any two messages m and $m\prime$ such that $h(m) = h(m\prime)$. One of the challenges involved with creating one-way hash functions is that they must operate on input data streams of variable size. Every hash function should be defined in terms of a compression function to deal with the variable length input problem. The hash function is public ; there is no secrecy to the process. The security of a one-way hash function is in its one-wayness. Only the hash value is signed. In this chapter we are going to describe a mechanism that rely on cryptographic hash function which is ElGamal signature scheme in the domain of natural integer. ElGamal signature scheme depends on the discrete logarithm problem and on Diffie-Hellman problem [4] and [22].

Redundancy Function An alternative to hashing is to use a redundancy function which transforms the original message into a larger space. A redundancy function is often used in digital signatures with message recovery. We are going to describe two signature schemes in the domain of natural integers that use a redundancy function and provide digital signature with message recovery which are the RSA signature scheme and the Rabin signature scheme. The RSA signature scheme depends on the integer factorization problem and on RSA problem. The Rabin signature scheme depends also on the integer factorization problem and on the square roots modulo composite n, see [21] and [22] for instance.

Note that the signature schemes in the domain of natural integers will be referred as classical digital signature schemes. Since these signature schemes are based on the number theoretic structure and algebraic foundations, for a better understanding, we are going to present some tools, basic concepts and important number theoretic problems in the domain of natural integers that are relevant to these algorithms.

CHAPTER 2

MATHEMATICAL BACKGROUND

This chapter contains a short mathematical basic knowledge in the area of number theory and algebraic result in the domain of natural integers that are particularly useful for the classical signature schemes described in this chapter. Mathematical proofs and further background can be found in references [1],[27],and [20] for instance.

2.1 Arithmetics in The Domain of Natural Integers

Integer Arithmetic. We begin with a review of a simple but very useful theorem which is the division algorithm theorem. The division algorithm of integers theorem is the following.

Theorem 1 *(Division algorithm in* $\mathbb{Z}$*) If a and b are integers with $b \geq 1$, the ordinary long division of a and b yields integers q (the quotient) and r (the remainder) such that $a = qb + r$, where $0 \leq r < b$. Moreover, q and r are unique.*

Applying the division algorithm to any integer a and a nonzero integer b, we say that b divides a just in case the remainder r is equal to zero. The integer that divides an integer a and an integer b is called a common divisor of a and b. The largest of all common divisors of a and b is called the greatest common divisor of a and b. We denote this integer by $gcd(a, b)$ or (a, b). A basic tool to compute the greatest common divisor of two positive integers is the Euclidean algorithm which is based on the fact that if an integer d divides two integers a and b, it divides their remainder. Thus, $gcd(a, b) = gcd(b, a \bmod b)$.

Algorithm 2 *(Euclidean algorithm in* $\mathbf{Z}$*) Given two positive integers a and b such that $a > b$, the Euclidean algorithm consists of performing the sequence of divisions to produce a decreasing sequence of integers : $r_1 > r_2 > r_3 ... r_n$ as the following :*

$$\begin{array}{lll} a & = bq_1 + r_1 & 0 < r_1 < b \\ b & = r_1 q_2 + r_2 & 0 < r_2 < r_1 \\ . & & \\ . & & \\ r_{k-2} & = r_{k-1} q_k \;\; + r_k & 0 < r_k < r_{k-1} \\ r_{k-1} & = r_k q_{k+1} \;\; + 0 & \end{array}$$

Now,

$gcd(a, b) = gcd(b, r_1) = ...r_k$

It follows that the last nonzero remainder r_k is the $gcd(a, b)$.

An exceeding important property of the greatest common divisor of two integers a and b is that it can be written in the form $ap + bq$ where p and q are both integers. The existence of such integers p and q and how to find them is very useful later when we consider the signature schemes. The algorithm for finding the values p and q in addition to the greatest common divisor of two integers can be stated as follow.

Algorithm 3 *(Extended Euclidean algorithm in* $\mathbf{Z}$*) Let a, b be two integers and let $d = \gcd(a, b)$. The process to compute d and to compute the integers p and q such that $ap + bq = d$ is the following:*

1. Set $a_1 = a$; $a_2 = b$; $x_1 = 1$; $x_2 = 0$; $y_1 = 0$; $y_2 = 1$.

2. Let $q = a_1/a_2$.

3. Set $a_3 = a_1 - qa_2$; $x_3 = x_1 + qx_2$; $y_3 = y_1 + qy_2$.

4. Set $a_1 = a_2$; $a_2 = a_3$; $x_1 = x_2$; $x_2 = x_3$; $y_1 = y_2$; $y_2 = y_3$.

5. If $a_2 > 0$ loop back to step 2.

6. If $ax_1 - by_1 > 0$ return $(d, p, q) = (a_1, x_1, -y_1)$ else return $(d, p, q) = (a_1, -x_1, y_1)$.

Every random natural number $p > 1$ whose only divisors are ± 1 and $\pm\, p$ is called a prime number. If p is not prime number, it is a composite number. The number 1 is neither prime nor composite; it is a unit. Any odd prime of the form $4k + 1$ can be written as a sum of two squares. Some characteristics of prime numbers are.

1. $p/a_1a_2......a_n \Rightarrow p/a_i$ for some i.

2. Every positive integer n can be expressed as a product of nontrivial powers of distinct primes $n = p_1^{e1}.p_2^{e2}.....p_k^{ek}$ and up to a rearrangement of the factors, this prime factorization is unique.

3. The number of primes is infinite.

Two numbers are relatively primes if their greatest common divisor is equal to 1. To determine the number of integers $\leq n$ which are relatively prime to a number n, we look for the Euler phi function named also totient function denoted by $\phi(n)$ where $n \geqslant 1$ and write $\phi(n) = \sum\limits_{\substack{1\leq k\leq n \\ \gcd(k,n)=1}} 1$. The totient function $\phi(n)$ has the following properties:

1. If p is a prime, then $\phi(p) = p - 1$.

2. If $gcd(m, n) = 1$, then $\phi(mn) = \phi(m)\phi(n)$. This implies that the Euler phi function is multiplicative.

3. If an integer n is expressed as a product of prime powers $n = p_1^{e1}.p_2^{e2}....p_k^{e_k}$, the value of $\phi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}).....(1 - \frac{1}{p_k})$ where $p_i \neq p_j$ and $1 \leq i,\ j \leq k$. In particular, if $n = p^k$ then $\phi(n) = \phi(p^k) = p^k(1 - \frac{1}{p}) = p^{k-1}(p - 1)$.

Modular Arithmetic Let n be a positive integer. For $a, b \in \mathbf{Z}$ we say that a is congruent to b modulo n and write $a \equiv b\ (mod\ n) \iff n/(a - b)$. n is called the modulus of the congruence and b is called the remainder or the residue of a modulo n. It is easy to verify that $\equiv$ is an equivalence relation on $\mathbf{Z}$ and hence partitions $\mathbf{Z}$ into a set of n disjoint equivalence classes which are $[0], [1], [2], ...[n - 1]$. The equivalency classes are called residue classes. A complete set of representative for the residue classes is called a complete residue system modulo n. One such system is $\mathbf{Z}_n = \{[0], [1], ..., [n - 1]\}$, denoted by $\{0, 1, 2, ...n - 1\}$ for simplicity. We define addition and multiplication on $\mathbf{Z}_n$ to be the same as ordinary addition and multiplication of integers, except that the results are reduced modulo n. The set $\{a_1, a_2, ..., a_h\}$ is called a reduced residue mod n if for every integer a such that $gcd(a, n) = 1$ there is i, $1 \leqslant i \leqslant h$ such that $a \equiv a_i$ $(mod\ n)$. In other words, any set of $\phi(n)$ integers which are all coprime to n and no two of which are congruent modulo n form a reduced residue system.

Next, we consider the problem of solving the congruence $ax \equiv b(mod\ n)$.

Let $d = gcd(a, n)$. If $b \nmid d$, there is no solution to the congruence. If b/d, there are d incongruent solutions. In particular, if $d = gcd(a, n) = 1$ and $b = 1$ there is one solution. By one solution, we mean that the solution is congruent to every other solution. This unique solution will be the multiplicative inverse of a modulo n denoted by a^{-1}. In turn, this is equivalent to say there exists $a \in \mathbf{Z}_n$ such that $ax + by = 1$ $(mod$ $n)$. This equation can be solved by the extended Euclidean algorithm to obtain the inverse of integer a molulo n. One simple way of calculating the multiplicative inverse of a modulo n is the extended Euclidean algorithm. If $gcd(a, n) > 1$ and $b = 1$, then a has no inverse in $\mathbf{Z}_n$. The computation of the inverse modulo n can also be done via the Euler's theorem which is stated below.

Theorem 4 *(Euler's theorem) Let n be a positive integer and a be an integer relatively prime to n. Then, $a^{\phi(n)} \equiv 1$ $(mod\ n)$.*

In the special case where $n = p$ is prime, every positive integer a less than p is coprime to p, that is $\phi(p) = p - 1$. This lead us to the following theorem.

Theorem 5 *(Fermat's little theorem) Let p be a prime positive integer and a be an integer such that $gcd(a, p) = 1$. Then, $a^{p-1} \equiv 1$ $(mod\ p)$. Therefore, $a^p \equiv a$ $(mod\ p)$.*

We turn now to establish a lemma that will be the base for Wilson's theorem.

Lemma 6 *Let p be a prime. The congruence $x^2 \equiv 1$ $(mod\ p)$ has exactly two solutions, namely, $x \equiv \pm 1 \pmod{p}$.*

Theorem 7 *(Wilson's theorem) Let p be a prime. Then $(p-1)! \equiv -1 \pmod{p}$*

We end up with a result that can be derived from Wilson's theorem.

Theorem 8 *Let p be an odd prime. The congruence $x^2 \equiv -1$ (mod p) is solvable if and only if* $p \equiv 1 \pmod 4$.

The next logical step is to be able to solve simultaneous linear congruencies. A fundamental tool for converting between a single congruence and a system of congruencies with smaller moduli is the Chinese remainder theorem.

Theorem 9 *(Chinese remainder theorem (CRT)) Let $n_1, n_2, \ldots n_k$ denote k positive integers that are relatively primes in pairs, that is to say, $gcd(n_i, n_j) = 1$ for $i \neq j$. Given the integers $a_1, a_2, \ldots, a_k$, there is a solution x_0 to the system such that for all i, $1 \leq i \leq k$ $x_0 \equiv a_i$ (mod n_i). Moreover, all such solutions are congruent to x_0 modulo n where $n = n_1.n_2 \ldots n_k$.*

Recall that an integer a modulo n is invertible modulo n if and only if $(a, n) = 1$. The set of all invertible elements modulo n constitute the multiplicative group $\mathbf{Z}_n^*$. $\mathbf{Z}_n^*$ $= \{a \in \mathbf{Z}_n \ / \ gcd(a, n) = 1\}$ where an element $a \in \mathbf{Z}_n$ is actually a residual class $[a]$ modulo n. If n is prime, then $\mathbf{Z}_n^* = \{a \ / \ 1 \leq a \leq n-1\}$. The order of $\mathbf{Z}_n^*$ defined to be the number of the elements in $\mathbf{Z}_n^*$ and denoted by $|\mathbf{Z}_n^*|$ is equal to the Euler Phi function $\phi(n)$. For $n = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_i^{\alpha_i}$ the Euler Phi function is $\phi(n) = (p_1 - 1)p_1^{\alpha_1 - 1}(p_2 - 1)p_2^{\alpha_2 - 1} \ldots (p_i - 1)p_i^{\alpha_i - 1}$. The order of an element $a \in Z_n^*$ is the least positive integer t such that $a^t \equiv 1 \pmod n$. If the order of $a \in Z_n^*$ is t, and $a^s \equiv 1 \pmod n$, then t divides s. In particular, $t|\phi(n)$. If the order of an element $\alpha \in Z_n^*$ is $\phi(n)$, then α is said to be a generator of Z_n^* or a primitive element of n. If Z_n^* has a generator, then Z_n^* is said to be cyclic.

Next, we list some of the properties of generators of Z_n^*.

1. Z_n^* has a generator if and only if $n = 2, 4, p^t$ or $2p^t$, where p is an odd prime and $t \geq 1$. In particular, if p is a prime, then Z_n^* has a generator.

2. If α is a generator of Z_n^*, then $Z_n^* = \{\alpha^i \pmod n \mid 0 \leq i \leq \phi(n) - 1\}$.

3. Suppose that α is a generator of Z_n^*. Then, $b = \alpha^i \pmod n$ is also a generator of Z_n^* if and only if $\gcd(i, \phi(n)) = 1$. It follows that if Z_n^* is cyclic, then the number of generators is $\phi(\phi(n))$.

4. $\alpha \in Z_n^*$ is a generator of Z_n^* if and only if $\alpha^{\phi(n)/p}$ is not congruent to $1 \pmod n$ for each prime divisor p of $\phi(n)$.

Note that finding a generator of a cyclic group is one of the difficult number theoretic problems. Now, depending on the above properties of a generator, to find a generator α of the cyclic group Z_p^*, we proceed as follows. First we select an element α in Z_p^* with $2 \leq \alpha \leq p - 2$. Then, use the division algorithm to write the order $\phi(p) = p - 1$ as a product of distinct primes p_i. If for every $p_i | p$, $\alpha^{\frac{p-1}{p_i}}$ is not congruent to 1 modulo p, then α is a generator of Z_p^*. Otherwise, α is not a generator. This statement is put in the following algorithm:

Algorithm 10 *(Finding a generator α of Z_p^*)*

```
1. Select an integer $\alpha$ from $[2, p-2]$.
2. Write $\phi(p) = p - 1 = p_1^{e1} p_2^{e2} ... p_k^{ek}$.
3. For $i = 1$ to $k$
    3.1 Compute $\alpha^{\frac{p-1}{p_i}} \pmod p$.
```

3.2 If $\alpha^{\frac{p-1}{p_i}} \pmod p$ *is not congruent to* $1 \pmod p$*, then return to step* 1.

4. Return with α *as generator of* Z_p^*.

2.2 Arithmetics in the Domain of Guassian Integers $\mathbf{Z}[i]$.

This section involves number theoretic properties extended from the domain of natural integers to the domain of Gaussian integers, see [30].We can extend numerous theoretic properties from the domain of natural integers to the domain of Gaussian integers denoted by $\mathbf{Z}[i]$, which is the subring of the field of complex numbers C consisting of all elements of the form $a + bi$ where a and $b \in \mathbf{Z}$.

Gaussian Integer Arithmetics The ring $\mathbf{Z}[i]$ is a commutative ring with unity and no zero divisor (that is, if α and β are two Gaussian integers such that $\alpha \neq 0$, we can't find $\beta \neq 0$ such that $\alpha\beta = 0$). Hence, $\mathbf{Z}[i]$ is an integral domain. An element $\mu \in \mathbf{Z}[i]$ is a unit in $\mathbf{Z}[i]$ if μ posses a multiplicative inverse $\lambda \in \mathbf{Z}[i]$ such that $\mu\lambda = \lambda\mu = 1$. The units in $\mathbf{Z}[i]$ are $1, -1, i, -i$. They divide each element in $\mathbf{Z}[i]$ and form a muliplicative group called group of units and denoted by $U(\mathbf{Z}[i])$. If there is an invertible element $\mu \in \mathbf{Z}[i]$ such that $\beta = \mu\alpha$, we say that β is associate to α denoted by $\beta \sim \alpha$. Hence, the associates of α are α, $-\alpha$, $i\alpha$, $-i\alpha$.

Let α be an element equal $a + bi$. We define the conjugate of α as $\overline{\alpha} = a - bi$ and the norm of α, called also the Gaussian valuation, as the product of the number α by its complex conjugate i.e $\delta(\alpha) = \alpha\,\overline{\alpha} = a^2 + b^2 = |\alpha|^2$. The main properties of the norm of Gaussian integers are defined in the following theorem.

Theorem 11 *(Properties of the norm of Gaussian integers)*

1. *Multiplicativity :* $\delta(\alpha\beta) = \delta(\alpha)\delta(\beta)\ \forall \alpha, \beta \in \mathbf{Z}[i]\backslash\{0\}$.

2. *Positivity :* $\delta(\alpha) \succeq 1 \forall$ *nonzero* $\alpha \in \mathbf{Z}[i]$.

3. *Units :* $\delta(\alpha) = 1 \Longleftrightarrow \alpha \in U(\mathbf{Z}[i]) = \{\pm 1, \pm i\}$.

4. *Divisibility : If* α/β *in* $\mathbf{Z}[i]$ *then* $\delta(\alpha)/\delta(\beta)$ *in* $\mathbf{Z}$.

5. $\delta(\alpha) = 0$ *iff* $\alpha = 0$.

6. *If* $\alpha \sim \beta$ *then* $\delta(\alpha) = \delta(\beta)$.

When a Gaussian integer α is divided by any Gaussian integer β, they are related by an equation that we formulate as the following.

Theorem 12 *(Division Algorithm in* $\mathbf{Z}[i]$*) Given two Gaussian integers* $\alpha, \beta \neq 0$*. Then,* $\exists\ q,\ r \in \mathbf{Z}[i]$ *such that* $\alpha = q\beta + r$ *where* $r = 0$ *or* $\delta(r) < \delta(\beta)$.

Note that the Gaussian integers q and r are not unique.

Example 13 *Take* $\alpha\ = 5 + 4i$ *and* $\beta = 1 - 2i$*. Then,* $5 + 4i = (-1 + 3i)(1 - 2i) - i$ *where* $\delta(r) = 1 < \delta(\beta) = 5$*. Thus, the quotient is* $-1 + 3i$ *and the remainder is* $-i$*. Note that the quotient and the remainder obtained from the division algorithm in* $\mathbf{Z}[i]$ *are not unique. For instance,*$-4+i = (-1+i)(5+3i)+(4-i) = (-1)(5+3i)+(1+4i)$

.

We should mention that we can't compare two Gaussian integers. In formulating the division algorithm in $\mathbf{Z}[i]$, we seem to run into a problem that $\mathbf{Z}[i]$ is not ordered. However, we can simply use the ordering on their norms. For instance, this criteria is used while finding the Euclidean greatest common divisor for Gaussian integers.

Definition 14 *(Greatest common divisor in* $\mathbf{Z}[i]$*) Suppose that* α *and* β *are two nonzero Gaussian integers. There exist a unique Gaussian integer* γ *called the greatest common divisor of* α *and* β*, denoted* $gcd(\alpha, \beta)$ *such that*

1. γ/α *and* γ/β.

2. *If* λ/α *and* λ/β *then* λ/γ $\forall\lambda$.

Example 15 *Let* $\alpha = 10+11i$; $\beta = 7+i$; $gcd(\alpha, \beta) = (10+11i)(4+i) + (7+i)(-5-7i) = 1$.

An efficient way to find the greatest common divisor (α, β) of any two Gaussian integers α and β is to make a repeated application of the division algorithm. This process is called the Euclidean algorithm.

Algorithm 16 *(Euclidean algorithm in* $\mathbf{Z}[i]$*) Given two Gaussian integers* α *and* β*. By a repeated application of the division algorithm, we obtain the following.*

$$\alpha = q_1\beta + r_1 \text{ where } \delta(r_1) < \delta(\beta).$$

$$\beta = q_2 r_1 + r_2 \text{ where } \delta(r_2) < \delta(r_1).$$

$$\cdot$$

$$\cdot$$

$$r_{n-2} = q_n r_{n-1} + r_n \quad \text{where } \delta(r_{n-1}) < \delta(r_{n-2}).$$

$$r_{n-1} = q_{n+1}\, r_n + 0.$$

The last nonzero remainder r_n *is the gcd of* α *and* β*.*

In $\mathbf{Z}[i]$, the Euclidean algorithm can be used to prove the following theorem.

Theorem 17 *(Bezout identity)*

Two Gaussian integers α and β not both equal to zero, have their greatest common divisor of the form $\alpha x + \beta y$ for some x and y.

This is a result of applying the Euclidean algorithm backwards, and write the gcd as a linear combination. Both implementing the Euclidean algorithm in $\mathbf{Z}[i]$ and going backward until the Bezout identity is found is commonly referred to as carrying out the extended Euclidean algorithm. Note that the extended Euclidean algorithm in $\mathbf{Z}[i]$ may also be used to compute γ and μ in the identity $\eta = gcd(\alpha, \beta) = \gamma\alpha + \mu\beta$.

Every nonzero nonunit Guassian integer β whose only divisors are the units and the associates is called a Gaussian prime. If β is not Gaussian prime, it is a Gaussian composite. Note that, if $\delta(\alpha)$ is prime in $\mathbf{Z}$ then α must be a prime in $\mathbf{Z}[i]$. The converse is not true. For example, 7 is prime in $\mathbf{Z}[i]$, but $\delta(7)$ is not prime in $\mathbf{Z}$. Also, not every prime in $\mathbf{Z}$ is prime in $\mathbf{Z}[i]$. For example, 13 is prime in $\mathbf{Z}$, but 13 is not prime in $\mathbf{Z}[i]$. $13 = (3+2i)(3-2i)$. The following theorem characterizes primes in $\mathbf{Z}[i]$.

Theorem 18 *(Types of Gaussian primes)*

1. Gaussian prime and its associate of the form $\alpha = 1 + i$
2. Π and $\bar{\Pi}$ in $\mathbf{Z}[i]$ where $p = \pi\bar{\pi}$ is an odd prime in $\mathbf{Z}$ of the form $4k+1$.
3. Odd primes of the form $4k+3$.

Note that the associate of a prime is also a prime, and the primes π and $\bar{\pi}$ are not associates. Also, note that if a Gaussian integer α is a Gaussian prime, we can also say it is irreducible in $\mathbf{Z}[i]$. In other words, a Gaussian integer α is said to be irreducible in $\mathbf{Z}[i]$ if $\delta(\alpha)$ is prime in $\mathbf{Z}$ that is if α is a Gaussian prime.The Gaussian integer 2 is not prime. The Gaussian integers $1+i$ and $1-i$ are associate primes.

Example 19 *$\delta(1+2i)$ is prime in $\mathbf{Z}$. Therefore, $1+2i$ is irreducible in $\mathbf{Z}[i]$. $\delta(1-2i)$ is prime in Z. Therefore, $1-2i$ is irreducible in $\mathbf{Z}[i]$.*

Gaussian integers obey a unique factorization theorem analog to that of integers.

Theorem 20 *(Unique factorization in $\mathbf{Z}[i]$) In the domain of Guassian integer, each non-zero, non-unit element α can be factorized into a product of Gaussian primes. $\alpha = \beta_1\beta_2....\beta_n$ and this factorization is unique up to ordering and associates.*

Congruencies in $\mathbf{Z}[i]$ Let α, β and γ be Gaussian integers with $\eta \neq 0$. If $\gamma/(\alpha - \beta)$, we say that α and β are congruent modulo γ and we write $\alpha \equiv \beta$ $(mod\ \gamma)$. If $(\alpha - \beta)$ is not divisible by γ, we say that α and β are incongruent modulo γ and we write $\alpha \not\equiv \beta$ $(mod\ \gamma)$. Let u be a unit in $\mathbf{Z}[i]$, then $\alpha \equiv \beta \pmod{\gamma}$ if and only if $\alpha \equiv \beta$ (mod $u\gamma$). Congruencies have many properties. Some properties that follow easily from the definition are listed in the following theorem.

Theorem 21 *(Congruence relation properties) Let α, β, η, μ, λ, v be any Gaussian integers and let $\gamma \neq 0$. Then, the congruence relations has the following properties:*

1. $\alpha \equiv \beta \pmod{\gamma}$ *and* $\eta \equiv \mu \pmod{\gamma} \Longrightarrow \alpha + \eta \equiv \beta + \mu$ $(mod\ \gamma)$.

2. $\alpha \equiv \beta \pmod{\gamma}$ *and* $\eta \equiv \mu \pmod{\gamma} \Longrightarrow \alpha\eta \equiv \beta\mu$ $(mod\ \gamma)$.

3. *If* $\lambda \neq 0$ *then* $\alpha \equiv \beta \pmod{\gamma}$ *if and only if* $\lambda\alpha \equiv \lambda\beta$ $(mod\ \gamma)$.

4. $\alpha \equiv \beta \pmod{\gamma}$ *and* $\eta \equiv \mu \pmod{\gamma} \Longrightarrow \lambda\alpha + \lambda\eta \equiv \lambda\beta + \beta\mu \ (mod\ \gamma)$.

5. *If* $\alpha \equiv \beta \pmod{\gamma}$ *and* μ/γ *then* $\alpha \equiv \beta \ (mod\ \mu)$.

6. *If* $\alpha \equiv \beta \pmod{\gamma}$ *and* $\alpha \equiv \beta \pmod{\nu}$ *then* $\alpha \equiv \beta \ (\mathrm{mod}\,\mathrm{lcm}(\gamma,\ \nu))$.

7. $\lambda\alpha \equiv \lambda\beta \pmod{\gamma}$ *if and only if* $\alpha \equiv \beta \ (\mathrm{mod}\ \gamma/(\gamma, \lambda))$.

From the properties above, we deduce the following.

Corollary 22 *(Congruence relation properties)*

a. If $(\gamma, \lambda) \sim 1$ and $\lambda\alpha \equiv \lambda\beta \pmod{\gamma}$ then $\alpha \equiv \beta \pmod{\gamma}$.

b. If $\alpha \equiv \beta \pmod{\gamma_1}$ and $\alpha \equiv \beta \pmod{\gamma_2}$ then $\alpha \equiv \beta \pmod{\gamma_1\gamma_2}$.

c. If $\alpha \equiv \beta \pmod{\gamma}$ then $(\gamma, \alpha) \sim (\gamma, \beta)$.

The congruencies are networthy specially relating to the multiplicative property.

Theorem 23 *Let* $\alpha,\ \beta$ *be in* $\mathbf{Z}[i]$ *and* $\eta \neq 0$. *Then,*

1. $\alpha\beta \equiv 1 \pmod{\eta}$ iff $(\eta, \alpha) \sim 1$ *and* $(\eta, \beta) \sim 1$.

2. *If* $\alpha\beta \equiv 1 \pmod{\eta}$ *and* $(\eta, \alpha) \sim 1$ *then* $\beta \equiv 0 \pmod{\eta}$.

3. $(\eta, \alpha) \sim 1$ iff $\exists\ \beta$ *such that* $\alpha\beta \equiv 1 \pmod{\eta}$.

4. *If* λ *is prime and* $\alpha\beta \equiv 0 \pmod{\lambda}$ *then* $\alpha \equiv 0$ *or* $\beta \equiv 0 \pmod{\lambda}$.

Given integers α, β and η in $Z[i]$ with $\eta \neq 0$. Consider the linear, or a first degree congruence $\alpha x \equiv \beta \pmod{m}$. We say that two solutions, r_1 and r_2 , are congruent solutions if $r_1 \equiv r_2 (\operatorname{mod} m)$; otherwise the solutions are said to be incongruent. The congruence is said to have a unique solution if a solution exists and is congruent to every other solution. Note that if $r_1 \equiv r_2 (\operatorname{mod} m)$ and r_1 is a solution of the congruence then r_2 is also a solution. Now if $\gcd(\alpha, \eta) \sim 1$ then the congruence $\alpha x \equiv 1 (\operatorname{mod} \eta)$ is solvable and has a unique solution. Moreover, if $(\alpha, \eta) = \gamma$, where γ is a non-unit, then no such solution exists. Also, if $(\alpha, \eta) \sim 1$ then the congruence $\alpha x \equiv \beta (\operatorname{mod} \eta)$ is solvable and has a unique solution.

For example, the congruence $(10 + 11i)x \equiv 1 (\operatorname{mod} 7 + i)$ can be solved applying Bezout's identity to express the *GCD* of $10 + 11i$ and $7 + i$ as the linear combination $1 = (4 + i)(10 + 11i) + (-5 - 7i)(7 + i)$. Hence, the solution is $x_0 = 4 + i$.

Now if α, β and η are Gaussian integers with $\eta \neq 0$ and if $\gamma = (\alpha, \eta)$, then the congruence $\alpha x \equiv \beta (\operatorname{mod} \eta)$ is solvable iff $\gamma | \beta$. The next theorem gives a complete classification of the solutions of the general linear congruence.

Theorem 24 *Let α, β and η be Gaussian integers with $\eta \neq 0$ and let $\gamma = (\alpha, \eta)$. If the congruence $\alpha x \equiv \beta (\operatorname{mod} \eta)$ is solvable, then there are $\delta(\gamma)$ incongruent solutions modulo η. The solutions are given by $x_0 + \rho \frac{\eta}{\gamma}$ where x_0 is the solution of $\frac{\alpha}{\gamma} x \equiv \frac{\beta}{\gamma} (\operatorname{mod} \frac{\eta}{\gamma})$ and ρ ranges over a complete set of residues modulo γ.*

We note that if α, β, γ and η are nonzero Gaussian integers, then the congruence $\gamma \alpha x \equiv \gamma \beta (\operatorname{mod} \gamma \eta)$ is equivalent to $\alpha x \equiv \beta (\operatorname{mod} \eta)$, and if $(\gamma, \eta) = 1$ then $\gamma \alpha x \equiv \gamma \beta (\operatorname{mod} \gamma \eta)$ is equivalent to $\alpha x \equiv \beta (\operatorname{mod} \eta)$.

For example, the congruence $(1+i)x \equiv 4 (\operatorname{mod} 2)$ is solved as follows: Let $\gamma = 1+i$ and $\frac{\alpha}{\gamma}x \equiv \frac{\beta}{\gamma}(\operatorname{mod} \frac{\eta}{\gamma})$ is $x \equiv 2-2i(\operatorname{mod} 1-i)$. Hence, the solutions are

$$x_1 = x_0 + d_1(\frac{\eta}{\gamma}) = 2-2i+0(1-i) = 2-2i$$

and

$$x_2 = x_0 + d_2(\frac{\eta}{\gamma}) = 2-2i+1(1-i) = 3-3i.$$

Let η_1, η_2, ... , η_r be r Gaussian integers that relatively prime in pairs and let α_1, α_2, ... , α_r be any r Gaussian integers. Then the Chinese remainder theorem states that the system of linear congruencies $x \equiv \alpha_1(\operatorname{mod} \eta_1)$, $x \equiv \alpha_2(\operatorname{mod} \eta_2)$, ..., and $x \equiv \alpha_r(\operatorname{mod} \eta_r)$ has a solution x_0. Any two such solution are congruent modulo $\eta = \eta_1.\eta_2...\eta_r$.

For example, the Gaussian integer x with the least value of $\delta(x)$ satisfying the three linear congruencies $x \equiv 3(\operatorname{mod} 1+i)$, $x \equiv 1+i(\operatorname{mod} 2+i)$, and $x \equiv 2(\operatorname{mod} 3)$ is $x = -1$.

Let $\beta \in Z[i]$, and let $\langle\beta\rangle = \beta Z[i] = Z[i]\beta$ be the ideal generated by β. Note that for γ and λ in a ring $Z[i]$, $\gamma \equiv \lambda(\operatorname{mod} \beta)$ iff $\gamma - \lambda \in \langle\beta\rangle$. Define $A(\beta)$ to be any complete set of distinct representatives from the cosets of $\langle\beta\rangle$ in $Z[i]$. We call $A(\beta)$ a complete residue system $(\operatorname{mod} \beta)$. For any nonzero Gaussian β any complete residue system modulo β has a finite number of elements, namely $\delta(\beta)$. For example, the complete residue system modulo $3+i$ is the set $\{0, 1, 2, 3, ..., 9\}$.

For any $\beta \in Z[i]$ we let the symbol $q(\beta)$ to be the order of the quotient ring $Z[i]/\langle\beta\rangle$. El-Kassar [15] showed that for any two non-zero elements γ and β of $Z[i]$, the complete residue system (mod $\gamma\beta$) is the set

$$\{s + r\gamma : s \in A(\gamma), r \in A(\beta)\}.$$

Also,

$$q(\beta\gamma) = q(\beta)q(\gamma).$$

Example 25 *Let $< \alpha >$ and $< \beta >$ be the ideal generated by $\alpha = 1+i$ and $\beta = 2-i$ respectively. Consider the cosets $0+ < \alpha >$, $1+ < \alpha >$, $2+ < \alpha >$. The coset $2+ < \alpha >= (1+i)(1-i)+ < 1+i >=< 1+i >$. Hence, $0+ < \alpha >$ and $1+ < \alpha >$ are distinct and form a complete residue modulo α. Consider the cosets $0+ < \beta >$, $1+ < \beta >$, $2+ < \beta > \ldots 5+ < \beta >$. The coset $5+ < \beta >= (2+i)(2-i)+ < 2-i >=< 2-i >$. Hence $0+ < \beta > \ldots 4+ < \beta >$ are distinct and form a complete residue modulo β . $A(1+i) = \{0, 1\}$ and $A(2-i) = \{0, 1, 2, 3, 4\}$. The complete residue mod $(1+i)(2-i) = A(3+i) = \{0, 1+i, 2+2i, 3+3i, 4+4i, 1, 2+i, 3+2i, 4+3i, 5+4i\}$.Also, 5 is not a prime in $Z[i]$, because $5 = -i\,(1+2i)\,(2+i)$. Hence, the complete residue system modulo $\beta = 5+5i = -i\,(1+2i)\,(2+i)\,(1+i)$ is $\{0,\ 5i,\ 2+i,\ 2+6i,\ 4+2i,\ 4+7i,\ 6+3i,\ 6+8i,\ 8+4i,\ 8+9i,\ 1,\ 1+5i,\ 3+i,\ 3+6i,\ 5+2i,\ 5+7i,\ 7+3i,\ 7+8i,\ 9+4i,\ 9+9i,\ 2,\ 2+5i,\ 4+i,\ 4+6i,\ 6+2i,\ 6+7i,\ 8+3i,\ 8+8i,\ 10+4i,\ 10+9i,\ 3,\ 3+5i,\ 5+i,\ 5+6i,\ 7+2i,\ 7+7i,\ 9+3i,\ 9+8i,\ 11+4i,\ 11+9i,\ 4,\ 4+5i,\ 6+i,\ 6+6i,\ 8+2i,\ 8+7i,\ 10+3i,\ 10+8i,\ 12+4i,\ 12+9i\}$.*

J. T. Cross [2] describes the complete residue systems modulo Gaussian prime powers for $\alpha = 1+i$, $p = 4k+3$, and π as follows.

Theorem 26 *Let $\alpha = 1+i$ and p be a prime satisfying $p \equiv 3(mod\ 4)$. Let Π be* *the prime factor of an integer $q \equiv 1$ (mod 4). We can construct the complete residue systems modulo $\pi^n, p^n, \alpha^{2m}, \alpha^{2m+1}$ as follow.*

1. $A(\alpha^{2m}) = \{a + bi : 0 \leq a \leq 2^m - 1, 0 \leq b \leq 2^m - 1\}$.

2. $A(\alpha^{2m+1}) = \{a + bi : 0 \leq a \leq 2^{m+1} - 1, 0 \leq b \leq 2^m - 1\}$.

3. $A(\pi^n) = \{a : 0 \leq a \leq q^n - 1\}$.

4. $A(p^n) = \{a + bi : 0 \leq a \leq p^n - 1, 0 \leq b \leq p^n - 1\}$.

Example 27 *Let $\beta = (1 + i)^4$. The complete residue system modulo $(1 + i)^4$ is $\{0,\ i,\ 2i,\ 3i,\ 1,\ 1 + i,\ 1 + 2i,\ 1 + 3i,\ 2,\ 2 + i,\ 2 + 2i,\ 2 + 3i,\ 3,\ 3 + i,\ 3 + 2i,\ 3 + 3i\}$. The complete residue systems modulo 7 is $\{0,\ i,\ 2i,\ 3i,\ 4i,\ 5i,\ 6i,\ 1,\ 1 + i,\ 1 + 2i,\ 1 + 3i,\ 1 + 4i,\ 1 + 5i,\ 1 + 6i,\ 2,\ 2 + i,\ 2 + 2i,\ 2 + 3i,\ 2 + 4i,\ 2 + 5i,\ 2 + 6i,\ 3,\ 3 + i,\ 3 + 2i,\ 3 + 3i,\ 3 + 4i,\ 3 + 5i,\ 3 + 6i,\ 4,\ 4 + i,\ 4 + 2i,\ 4 + 3i,\ 4 + 4i,\ 4 + 5i,\ 4 + 6i,\ 5,\ 5 + i,\ 5 + 2i,\ 5 + 3i,\ 5 + 4i,\ 5 + 5i,\ 5 + 6i,\ 6,\ 6 + i,\ 6 + 2i,\ 6 + 3i,\ 6 + 4i,\ 6 + 5i,\ 6 + 6i\}$. The complete residue system modulo $(2 + i)^2$ is $\{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10,\ 11,\ 12,\ 13,\ 14,\ 15,\ 16,\ 17,\ 18,\ 19,\ 20,\ 21,\ 22,\ 23,\ 24\}$.*

Note that using the results of J. T. Cross [2] and the results of El-Kassar [15], any complete residue system modulo Gaussian prime integer can be constructed.

Let $R(\beta)$ be those elements of $A(\beta)$ that are relatively prime to β; in other words an element x of $A(\beta)$ is in $R(\beta)$ if and only if $\gcd(x, \beta) = 1$. The set $R(\beta)$ is called a reduced residue system (mod β). Since for any Gaussian β any complete residue system modulo β, $A(\beta)$, has a finite number of elements, then the number of elements in any reduced residue system modulo β, $R(\beta)$, is constant.

Example 28 *Let $\beta = 5+5i = -i\,(1+2i)\,(2+i)\,(1+i)$. The complete residue system modulo β is* $\{0,\ 5i,\ 2+i,\ 2+6i,\ 4+2i,\ 4+7i,\ 6+3i,\ 6+8i,\ 8+4i,\ 8+9i,\ 1,\ 1+5i,\ 3+i,\ 3+6i,\ 5+2i,\ 5+7i,\ 7+3i,\ 7+8i,\ 9+4i,\ 9+9i,\ 2,\ 2+5i,\ 4+i,\ 4+6i,\ 6+2i,\ 6+7i,\ 8+3i,\ 8+8i,\ 10+4i,\ 10+9i,\ 3,\ 3+5i,\ 5+i,\ 5+6i,\ 7+2i,\ 7+7i,\ 9+3i,\ 9+8i,\ 11+4i,\ 11+9i,\ 4,\ 4+5i,\ 6+i,\ 6+6i,\ 8+2i,\ 8+7i,\ 10+3i,\ 10+8i,\ 12+4i,\ 12+9i\}$. *Hence, the reduced residue modulo β is* $\{1,\ 5+2i,\ 7+3i,\ 9+4i,\ 2,\ 4+i,\ 8+3i,\ 10+4i,\ 3,\ 5+i,\ 7+2i,\ 11+4i,\ 4,\ 6+i,\ 8+2i,\ 10+3i\}$.

Theorem 29 *Let γ and β be any two non-zero elements of $Z[i]$. In [??] El-Kassar showed that if s is in $A(\beta)$ and r in $A(\beta)$, then $\gcd(s+r\gamma,\gamma)=1$ iff $s \in R(\gamma)$. Also, if $\gcd(\beta,\gamma)=1$ and $s \in A(\gamma)$, then the set $R(\beta)=\{s+r\gamma : r \in A(\beta)\}$ consists of distinct residues $(\mathrm{mod}\,\beta)$. Moreover, this collection is a complete residue system $(\mathrm{mod}\,\beta)$. If $s \in R(\beta)$ then there exist $\overline{s} \in R(\beta)$ such that $\overline{s}s \equiv 1(\mathrm{mod}\,\beta)$. Note also that if p is a prime in $Z[i]$, then $x^2 \equiv 1(\mathrm{mod}\,p)$ iff $x \equiv \pm 1(\mathrm{mod}\,p)$.*

J. T. Cross [2] classified the reduced residue systems modulo Gaussian prime powers for $\alpha = 1+i$, $p = 4k+3$, and π as follows:

1. $R(\alpha^n) = \{a+bi \in A(\alpha^n) : a \not\equiv b(\mathrm{mod}\,2)\}$.
2. $R(p^n) = \{a+bi \in A(p^n) : (a,p) \sim 1$ *or* $(b,p) \sim 1\}$.
3. $R(\pi^n) = \{a \in A(\pi^n) : (a,q) \sim 1\}$.

For example, the reduced residue system modulo $(1+i)^4$ is $\{i,\ 3i,\ 1,\ 1+2i,\ 2+i,\ 2+3i,\ 3,\ 3+2i\}$. The reduced residue system modulo 7 is $\{0,\ i,\ 2i,\ 3i,\ 4i,\ 5i,\ 6i,\ 1,\ 1+i,\ 1+2i,\ 1+3i,\ 1+4i,\ 1+5i,\ 1+6i,\ 2,\ 2+i,\ 2+2i,\ 2+3i,\ 2+4i,\ 2+5i,\ 2+6i,\ 3,$

$3+i$, $3+2i$, $3+3i$, $3+4i$, $3+5i$, $3+6i$, 4, $4+i$, $4+2i$, $4+3i$, $4+4i$, $4+5i$, $4+6i$, 5, $5+i$, $5+2i$, $5+3i$, $5+4i$, $5+5i$, $5+6i$, 6, $6+i$, $6+2i$, $6+3i$, $6+4i$, $6+5i$, $6+6i\}$. The reduced residue system modulo $(2+i)^2$ is $\{1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24\}$.

Now, we are going to consider the factor rings of the Domain of Gaussian integers. First, we will describe these rings, classify their equivalence classes and examine their group of units. Then, we will classify the factor rings with primitive roots.

Let β be a Gaussian integer. The factor ring modulo β is the ring $Z[i]/\langle\beta\rangle$. The elements of this ring are the equivalence classes of the form $[a+bi]$ or $(a+bi)+\langle\beta\rangle$. The operations of the factor ring defined by $[\alpha]+[\beta]=[\alpha+\beta]$ and $[\alpha][\beta]=[\alpha\beta]$. Note that the order of a factor ring modulo β is equal to the number of elements in a complete residue system modulo β.

Then from the above results obtained by J. T. Cross [2], the factor ring modulo prime powers can be constructed as follows:

1. $Z[i]/\langle\pi^n\rangle = \{[a] : 0 \le a \le q^n - 1\}$.

2. $Z[i]/\langle p^n\rangle = \{[a+bi] : 0 \le a \le p^n - 1, 0 \le b \le p^n - 1\}$.

3. $Z[i]/\langle\alpha^{2m}\rangle = \{[a+bi] : 0 \le a \le 2^m - 1, 0 \le b \le 2^m - 1\}$.

4. $Z[i]/\langle\alpha^{2m+1}\rangle = \{[a+bi] : 0 \le a \le 2^{m+1} - 1, 0 \le b \le 2^m - 1\}$.

Note that the order of the factor ring $Z[i]/\langle\pi^n\rangle$ is q^n , that of $Z[i]/\langle p^n\rangle$ is p^{2n} , that of $Z[i]/\langle\alpha^{2m}\rangle$ is 2^{2m}, and that of $Z[i]/\langle\alpha^{2m+1}\rangle$ is 2^{2m+1}. In El-Kassar [15], it was shown that for a Gaussian integer β, $Z[i]/\langle\beta\rangle$ is of order $\delta(\beta)$ and for any two elements

β and γ of $Z[i]$, the order of $Z[i]/\langle\beta\gamma\rangle$ is the product of the order of $Z[i]/\langle\beta\rangle$ by the order of $Z[i]/\langle\gamma\rangle$.

We note that $Z[i]/\langle\beta\rangle$ is a finite ring with identity. and $Z[i]/\langle\beta\rangle$ is a field iff β is prime. Moreover, the units of the factor ring $Z[i]/\langle\beta\rangle$ form a group under multiplication, $(Z[i]/\langle\beta\rangle)^*$.

For simplicity, $Z[i]/\langle\beta\rangle$ will be denoted by G_β and its group of units is denoted by G_β^*. In $Z[i]/\langle\beta\rangle$, $[\alpha]=[\beta]$ if and only if $\alpha\equiv\gamma(\mathrm{mod}\,\beta)$, so we will identify G_β by a complete residue system modulo β. For example, when $\beta=1+2i$, then $Z[i]/\langle 1+2i\rangle=\{[0],[1],[2],[3],[4]\}=G_{1+2i}$. This ring is identified by $G_{1+2i}=\{0,1,2,3,4\}$. Hence,

1. $G_\pi=\{a:0\le a\le q-1\}$.
2. $G_p=\{a+bi:0\le a\le p-1, 0\le b\le p-1\}$.
3. $G_{\pi^n}=\{a:0\le a\le q^n-1\}$.
4. $G_{p^n}=\{a+bi:0\le a\le p^n-1, 0\le b\le p^n-1\}$.
5. $G_{\alpha^{2m}}=\{a+bi:0\le a\le 2^m-1, 0\le b\le 2^m-1\}$.
6. $G_{\alpha^{2m+1}}=\{a+bi:0\le a\le 2^{m+1}-1, 0\le b\le 2^m-1\}$.
7. $G_{\gamma\beta}=\{s+r\gamma:s\in G_\gamma, r\in G_\beta\}$, where γ and β are non-zero elements of $Z[i]$.

The group of units of G_β, G_β^* is identified to be the reduced residue system modulo β. When β is a Gaussian prime, G_β is a field and G_β^* is the set of its nonzero elements. Note that to determine the inverse δ of γ in G_β we solve the congruence

$\gamma x \equiv 1 \,(\operatorname{mod} \beta)$. For example, the inverse of $10+11i$ in G_{7+i} is $4+i$. Then group of units modulo prime powers can be constructed as follows:

1. $G^*_{\alpha^n} = \{a+bi \in G_{\alpha^n} : a \neq b(\operatorname{mod} 2)\}$.

2. $G^*_{\pi^n} = \{a \in G_{\pi^n} : (a,q) \sim 1\}$.

3. $G^*_{p^n} = \{a+bi \in G_{p^n} : (a,p) \sim 1 \text{ or } (b,p) \sim 1\}$.

4. $G^*_{\gamma\beta} = \{s + r\gamma : s \in G^*_{\gamma}, r \in G^*_{\beta}\}$, where γ and β are two non-zero elements of $Z[i]$.

Example 30 $G^*_{(1+i)^2} = \{1, i\}$; $G^*_{(1+2i)} = \{0, 1, 2, 3, 4\}$

The order of the group of units modulo a Gaussian prime β is the number of elements in any reduced residue system modulo β, G^*_{β}, which is constant and is denoted by $\phi(\beta)$. Note that $\phi(\beta)$ is the extension of Euler's ϕ-function to the domain of Gaussian integers. The value of $\phi(\beta)$ is obtained by using the fact that $\phi(\beta)$ is a multiplicative function. J. T. Cross [2] and El-Kassar [15] described $\phi(\beta)$ as follows:

1. $\phi(\alpha^n) = 2^n - 2^{n-1}$.

2. $\phi(\pi^n) = q^{n-1}(q-1)$.

3. $\phi(p^n) = p^{2n-2}(p^2-1)$.

4. $\phi(\gamma\beta) = \phi(\gamma)\phi(\beta)$, where γ and β are two non-zero elements of $Z[i]$.

For example, the orders of the groups G^*_{11+2i} is

$$\phi(11+2i) = \phi(-(1+2i)^3) = \phi((1+2i)^3) = 5^{3-1}(5-1) = 100.$$

That of G^*_{4-4i} is

$$\phi(4-4i) = \phi(i\alpha^5) = \phi(\alpha^5) = 2^5 - 2^4 = 16.$$

That of G^*_{27} is

$$\phi(27) = \phi(3^3) = 3^{2(3)-2}(3^2 - 1) = 648.$$

In [2] and [15], J. T. Cross and El-Kassar determined the Structure of the group of units of G^*_β as follows:

1. $G^*_{\alpha^n} \cong Z_{2^{m-1}} \times Z_{2^{m-2}} \times Z_4$ if $n = 2m$.

2. $G^*_{\alpha^n} \cong Z_{2^{m-1}} \times Z_{2^{m-1}} \times Z_4$ if $n = 2m+1$.

3. $G^*_{\pi^n} \cong Z_{q^n - q^{n-1}}$.

4. $G^*_{p^n} \cong Z_{p^{n-1}} \times Z_{p^{n-1}} \times Z_{p^2-1}$.

5. If $(\beta_1, \beta_2) \sim 1$, then $G^*_{\beta_1\beta_2} \cong G^*_{\beta_1} \times G^*_{\beta 2}$.

For example, the structure of the group of units $G^*_{2340+5580i}$ is isomorphic to $G^*_{(1+i)^5} \times G^*_{(1+2i)^2} \times G^*_{2+i} \times G^*_{3^3} \times G_{7+8i} \cong Z_2 \times Z_2 \times Z_4 \times Z_{20} \times Z_4 \times Z_3 \times Z_3 \times Z_8 \times Z_{112}$.

A Gaussian integer β is said to have a primitive root iff G^*_β is cyclic. If γ is a generator of G^*_β then we say that γ is a primitive root for β. J. T. Cross [2] showed that any Gaussian integer β has a primitive root iff β is α^2, α^3, π^n, p, $\alpha\pi^n$, or αp.

Example 31 $G^*_{\alpha^2} = \{1,\ i\}$ *is cyclic with* i *being a generator. Also,* $G^*_{\alpha^3} = G^*_{2\alpha} = \{1,\ 3,\ i,\ 2+i\}$ *is cyclic with* i *being a generator. However,* $G^*_4 = G^*_{\alpha^4} = \{1,\ 3,\ i,\ 3i,\ 1+2i,\ 2+i,\ 3+2i,\ 2+3i\}$ *is not cyclic. To see this, one can verify that* G^*_4 *is the direct* *product of the two subgroups* $\{1,\ 1+2i\}$ *and* $\{1,\ 3,\ i,\ 3i\}$*. Finally, the group* $G_3^* = \{1,\ 2,\ i,\ 2i,\ 1+i,\ 1+2i,\ 2+i,\ 2+2i\}$ *has* $1+i$ *as a generator.*

In a similar manner to that in the domain of natural integers, we could find generators of a cyclic group G_β^* in the domain of Gaussian integers $Z[i]$. That is, with some modifications to $Z[i]$, algorithm (1) is used. That is, to find a generator θ of the cyclic group G_β^*, where β is either α^2, α^3, π^n, p, $\alpha\pi^n$, or αp, we proceed as follows. First we select an element θ in G_β^* with $\theta \neq 1$ and $\phi(\beta)$. Then, we use the division algorithm to write the order $\phi(\beta)$ as a product of distinct primes p_i. If for every $p_i|\phi(\beta)$, the element $\theta^{\frac{\phi(\beta)}{p_i}}$ is not congruent to 1 modulo β, then θ is a generator of G_β^*. Otherwise, θ is not a generator. This statement is put in the following algorithm:

Algorithm 32 *(Finding a generator* θ *of* G_β^**)*

```
1. Select an integer $\theta \neq 1$ and $\phi(\beta)$ from $G_\beta^*$.
2. Write $\phi(\beta) = p_1^{e_1} p_2^{e_2} ... p_k^{e_k}$.
3. For $i = 1$ to $k$
    3.1 Compute $\theta^{\frac{\phi(\beta)}{p_i}} \pmod{\beta}$.
    3.2 If $\theta^{\frac{\phi(\beta)}{p_i}} \pmod{\beta}$ is not congruent to $1 \pmod{\beta}$, then return to step 1.
4. Return with $\theta$ as generator of $G_p^*$.
```

Theorem 33 *(Euler's theorem in* $\mathbf{Z}[i]$*) Let* β *be a Gaussian integer. For any Gaussian integer* α *relatively prime to* β*, we have* $\alpha^{\phi(\beta)} \equiv 1$ *(mod* β*) where* $\phi(\beta)$ *is the Euler's phi function of the Gaussian integer* β*.*

A special case of Euler's theorem for Gaussian integers is Fermat's theorem for Gaussian integers.

Theorem 34 *(Fermat's theorem in* $\mathbf{Z}[i]$*)*

Let γ be a Gaussian prime. For every Gaussian integer α such that $(\alpha, \gamma) = 1$, we have $\alpha^{\phi(\gamma)} \equiv 1$ (mod γ).

Example 35 *Let $\gamma = 1 + i$ and $\alpha = 2 + i$. Hence, $\alpha^{\phi(\gamma)} = (2 + i)^1 \equiv 1 \pmod{(1 + i)}$.*

Theorem 36 *(Wilson's theorem in* $\mathbf{Z}[i]$*) Let β be a Gaussian prime and let $R(\beta)$ be any reduced system modulo β. Then, $\prod_{\alpha \in R(\beta)} \alpha \equiv -1$ (mod β).*

Example 37 *Take $\beta = 1 + 3i$. Then, $R(1 + 3i) = \{1, 1 + i, 1 + 2i, i, \ 2i, 3i\}$ is the reduced system modulo $1 + 3i$. The product $(1)(1 + i)(1 + 2i)(i)(2i)(3i) \equiv -1$ (mod $(1 + 3i)$).*

2.3 Arithmetics in the Domain of Polynomial Rings

Let F be a field with q elements. By the ring of polynomials in variable x over F which is always written $F[x]$, we mean the set of all formal expressions $p(x) = a_0 + a_1x + a_2x^2 + ...a_{n-1}x^{n-1} + a_nx^n$ where the coefficients a_i are in F. a_n is called the leading coefficient of $p(x)$. If $a_n = 1$, the polynomial $p(x)$ is called a monic polynomial. If $a_n \neq 0$, the degree of $p(x)$ denoted by $deg p(x)$ is n. So, the degree of a polynomial $p(x)$ is the highest power of x that occurs in the expression $p(x)$ with a nonzero coefficient. If $p(x) = a_0$ then $p(x)$ is of degree 0 and so it is called a constant polynomial in F. Note that every constant polynomial is a unit (invertible polynomial). We will state some properties of the degree function.

Theorem 38 *(Properties of the degree function).Let $p(x)$, $q(x)$ be polynomials in $F[x]$.*

1. *$deg(p(x)q(x)) = degp(x) + degq(x)$*

2. *If $p(x) + q(x) \neq 0$ then*

   $deg(p(x) + q(x)) \leq max(degp(x), degq(x))$

Using the usual methods of algebra, we can perform arithmetic operations on polynomials in $F[x]$.

Definition 39 *(Polynomials divisibility) Let $f(x)$, $g(x) \in F[x]$ with $g(x) \neq 0$. We say that $g(x)$ divides $f(x)$ in $F[x]$, written as $g(x)/f(x)$ iff $\exists$ a polynomial $a(x) \in F[x]$ such that $f(x) = g(x).a(x)$. In this case, we call $g(x)$ a factor or a divisor of $f(x)$.*

Note that, if $f(x)/g(x)$ and $g(x)/f(x)$then $f(x)$and $g(x)$ are called associates. If $f(x)/g(x)$ and $g(x) \nmid f(x)$ then the degree of $f(x)$ is less than the degree of $g(x)$.

Theorem 40 *(Division algorithm for polynomials) Let $g(x)$, $h(x)$ be polynomials in $F[x]$ with $h(x) \neq 0$. The ordinary polynomial long division of $g(x)$ by $h(x)$ yields unique polynomials $q(x)$ and $r(x) \in F[x]$ called the quotient and the remainder such that*

$$g(x) = q(x)h(x) + r(x) \qquad r(x) = 0 \text{ or } degr(x) < degh(x).$$

Immediate applications of the Division algorithm for polynomials theorem allow us to determine the following theorems.

Theorem 41 *(Remainder theorem) Given a polynomial $f(x) \in F[x]$ and $c \in F$. There exist a polynomial $q(x) \in F[x]$ such that $f(x) = q(x)(x - c) + f(c)$.*

Theorem 42 *(Factor theorem) Given a polynomial $f(x) \in F[x]$ and $c \in F$. Then, $x - c$ divides $f(x)$ iff $f(c) = 0$, that is to say, c is a root of $f(x)$. More generally, $x - c$ divides $f(x)$ with remainder $f(c)$.*

Theorem 43 *(Basis theorem) Let $g(x) \in F[x]$. $I = \{f(x)g(x)/f(x) \in F[x]\}$ is an ideal of $F[x]$, that is, I consists of all multiples of the fixed polynomial $g(x)$ by the elements of $F[x]$.*

The polynomial ring $F[x]$ has many properties in common with the integers. More precisely, $F[x]$ and $\mathbf{Z}$ are both Euclidean domains. Hence we can talk about the greatest common divisor for polynomials.

Definition 44 *(Greatest common divisors for polynomials) Given two nonzero polynomials $f(x)$ and $g(x)$ in $F[x]$. There exist a unique monic polynomial $d(x)$ called the greatest common divisor of $f(x)$ and $g(x)$ and written $gcd(f(x), g(x))$ such that*

1. *$d(x)/f(x)$ and $d(x)/g(x)$.*

2. *If $c(x)/f(x)$and $c(x)/g(x)$ for some $c(x) \in F[x]$, then $c(x)/d(x)$.*

Every pair of nonzero polynomials has a greatest common divisor. A simple way to determine the greatest common divisor for polynomials is to make an analogy with the Euclidean algorithm for integers and yielding the polynomial Euclidean algorithm.

Algorithm 45 *(Polynomial Euclidean algorithm) Assume $f(x)$ and $g(x)$ are two nonzero polynomials. We let $r(x)$ be the remainder after division of $f(x)$ by $g(x)$. We denote $r(x)$ by $r(x) = g(x) - q_1(x).f(x)$.*

By a repeated application of the division algorithm, we obtain the following.

$$g(x) = q_1(x).f(x) + r_1(x) \qquad \textit{where } \deg(r_1(x)) < \deg(f(x)).$$

$$r_1(x) = q_2(x)r_1(x) + r_2(x) \qquad \textit{where } \deg(r_2(x)) < \deg(r_1(x)).$$

$$\cdot$$

$$\cdot$$

$$r_{n-2}(x) = q_n(x)r_{n-1}(x) + r_n(x) \qquad \textit{where } \deg(r_{n-1}(x)) < \deg(r_n(x)).$$

$$r_{n-1}(x) = q_{n+1}(x)r_n\ (x) + 0.$$

The last positive integer $r_n(x)$ in the sequence $r_1(x), r_2(x), ...$ is the greatest common divisor of $f(x)$ and $g(x)$.

Since $F[x]$ is a Euclidean domain, Bezout identity holds and is stated as follow.

Theorem 46 *(Bezout identity)*

Two polynomials $f(x)$ and $g(x)$ in $F[x]$, not both equal to zero, have gcd of the form $a(x)f(x) + b(x)g(x)$ for some $a(x), b(x) \in F[x]$.

This is a result of applying the Euclidean algorithm backwards and writing the gcd as a linear combination. Both, implementing the polynomial Euclidean algorithm and going backward until the Bezout identity is found, is commonly referred to as carrying out the extended Euclidean algorithm. Note that, the extended polynomial Euclidean algorithm may also be used to compute the coefficients $a(x)$ and $b(x)$ in the identity $d(x) = gcd(f(x), g(x)) = a(x)f(x) + b(x)g(x)$. Moreover, If $d(x) = 1$, then the two polynomials $f(x)$ and $g(x)$ are said to be relatively prime or coprime.

Every polynomial $g(x) \in F[x]$ of degree $m \geqslant 1$ is said to be irreducible if it cannot be written as a product of two polynomials in $F[x]$ each having a positive degree less than m. Note that every linear polynomial is irreducible over F. The only divisors of the irreducible polynomial $g(x)$ are polynomials of degree 0 (constant polynomials) and associates of $g(x)$. Thus, given any polynomial $f(x)$ in $F[x]$, either $g(x)/f(x)$ or $(g((x), f(x)) = 1$. Irreducible polynomials are rather like the prime numbers. This similarity allow us to determine the following theorem.

Theorem 47 *Let $p(x)$ be an irreducible polynomial in $F[x]$. If $p(x)$ divides $r(x).s(x)$ for $r(x)$ and $s(x) \in F[x]$, then either $p(x)$ divides $r(x)$ or $p(x)$ divides $s(x)$.*

For any polynomial $f(x)$ there is a unique (up to order of factors) representation of $f(x)$ as a constant multiple of a product of monic irreducibles. This is called the factorization of $f(x)$. That is, given a polynomial $f(x)$, $\exists$ a scalar c and monic irreducible polynomials $p_1(x), ..., p_r(x)$ such that $f(x) = ap_1(x)^{m_1}p_2(x)^{m_2}....p_k(x)^{m_k}$ where a is the leading coefficient of $f(x)$ and $p_1(x), ...p_k(x)$ are monic and irreducible in $F[x]$, $m_1 > 0, ...m_k > 0$. This factorization is unique up to the order of the $p_i(x)$.

Quotient Rings and Modular Arithmetic over Polynomials Given a field F. Consider the ring of polynomials $F[x]$ with coefficients in F and a polynomial $f(x) \in F[x]$ of degree n. Let $< f(x) >= \{f(x).h(x)/h(x) \in F[x]\}$ be the principal ideal generated by $f(x)$. We note that $< f(x) >=< g(x) >$ where $g(x)$ is a monic polynomial associate to $f(x)$. Hence, we restrict our attention to the case where $f(x)$ is a monic polynomial. The set $F[x]/ < f(x) >= \{g(x)+ < f(x) > / \ g(x) \in F[x]\}$ of all distinct cosets $< f(x) >$ is a ring in which addition and multiplication are defined and $F[x]/ < f(x) >$ is a commutative ring with identity. If $f(x)$ is irreducible in $F[x]$, $F[x]/ < f(x) >$ is a field. For simplity, we denote any coset $h(x)+ < f(x) >$ by $[h(x)]$ and $[g(x)] = [h(x)]$

iff $g(x) - h(x) = q(x).f(x)$ for some $q(x) \in F[x]$. Also $g(x) \in < f(x) >$ iff $[g(x)] = [0]$.

Given a field F. Let $p(x)$ be a fixed polynomial over F and $a(x)$ and $b(x)$ $\in F[x]$. We say that $a(x)$ is congruent to $b(x)$ modulo $p(x)$ and we write $a(x) \equiv b(x)\ (mod\ p(x)) \Longleftrightarrow p(x)/\ (a(x) - b(x))$. If $a(x)$, $p(x) \in F[x]$ then $p(x)/a(x)$ iff $a(x) \equiv 0\ (mod\ p(x))$. The relation of congruence modulo a fixed polynomial $p(x)$ is an equivalence relation. The set of equivalence classes of polynomials in $F[x]$ modulo $p(x)$ will be denoted by $F[x]/ < p(x) >$. So, $a(x) \equiv b(x)\ (\mathrm{mod}\, p(x))$ is equivalent to $[a(x)] = [b(x)]$ in $F[x]/ < p(x) >$. Note that, if $p(x)$ is a polynomial in $F[x]$ such that deg $p(x) > 0$ then $\forall\ a(x) \in F(x)$, the congruence class $[a(x)]$ modulo $p(x)$ contains by the division algorithm a unique representative $r(x)$ with degree $r(x) < deg\ p(x)$ or $r(x) = 0$. Moreover, for any polynomial $a(x)$ in $F[x]$ we have that $[a(x)] = [r(x)]$ in $F[x]/ < p(x) >$. It can be easily shown that all the properties of congruencies of integers hold also for polynomials.

Each congruence class of any polynomial in $F[x]$ have a set of distinct representative elements named complete residue system defined as the following.

Definition 48 *(Complete residue system of polynomials) Let $f(x)$ be a fixed polynomial in $F[x]$ of degree n. The set of the polynomials in $F[x]$ that are distinct with degree less than n constitute the complete residue system of $A(f(x))$. This system can be defined to be the set $A(f(x)) = \{a_0+a_1x+a_2x^2+.......a_{n-1}x^{n-1} \quad / \ a_0, a_1, ..., a_{n-1} \in F\}$.*

Note that, all polynomials in the complete residue system modulo $f(x)$ are incongruent and each polynomial in $F[x]$ must be congruent to a polynomial in $A(f(x))$. Also, we can define a set containing all polynomials of the complete residue system of a fixed polynomial $f(x)$ that are relatively primes to $f(x)$. From the complete residue system modulo $A(f(x))$, we can obtain another system called the reduced residue system modulo $f(x)$ which can be defined as follow.

Definition 49 *(Reduced residue system of polynomials) The reduced residue system of a polynomial $f(x)$ of degree n is the set of polynomials belonging to the complete residue system $A(f(x))$ and are relatively prime to .$f(x)$. This set can be defined as $R((p(x)) = \{g(x) \in A((p(x)) : (f(x), g(x)) = 1\}$.*

The order of the reduced residue system is $\phi(p(x)) = p^n - 1$ where n is the degree of $p(x)$. Let $p(x)$ and $q(x)$ be fixed polynomials in $F[x]$. We give some properties of congruencies in polynomials

Theorem 50 *Let $a(x)$, $b(x)$, $c(x)$ and $d(x) \in F[x]$ where $p(x)$ and $q(x)$ are fixed polynomials in $F[x]$.*

1. *If $a(x) \equiv b(x) \pmod{p(x)}$ and $q(x)/p(x)$, then $a(x) \equiv b(x) \quad (mod\ q(x).)$*
2. *If $a(x) \equiv b(x) \pmod{p(x)}$ and $a(x) \equiv b(x) \quad (mod\ q(x))$, then $a(x) \equiv b(x)$ mod $[p(x), q(x)]$.*
3. *If $c(x).a(x) \equiv c(x).b(x) \quad (mod\ p(x))$ then $a(x) \equiv b(x) \quad mod(\dfrac{p(x)}{(p(x), c(x))})$.*
4. *$c(x).a(x) \equiv c(x).b(x)$ mod $(c(x).p(x))$ iff $a(x) \equiv b(x)$ $(mod\ p(x).)$*

To find a multiplicative inverse, we have two problems. The first is to decide whether a multiplicative inverse exists. The second is to develop a technique for finding it systematically. To address the first problem, we check if $gcd(a(x), p(x)) = 1$. If so, then the multiplicative inverse exists.

Definition 51 *(Multiplicative inverse of a polynomial) Let $a(x)$ be a polynomial in $F[x]$. Define the multiplicative inverse of $a(x)$ modulo $p(x)$ to be a polynomial $b(x) \in F[x]$ such that $a(x).b(x) \equiv 1 \ (mod\ p(x))$. That is, the multiplicative inverse of $[a(x)]$ in $F[x]/ < p(x) >$ is a congruence class $[b(x)]$ such that $[a(x)].[b(x)] = 1$.*

Hence, every nonzero polynomial whose degree is less than the degree of $p(x)$ has a unique multiplicative inverse modulo $p(x)$. A simple way to calculate the multiplicative inverse of a polynomial is the Euclidean algorithm for polynomials. We can say therefore that if $(a(x), p(x)) = 1$, the equation $a(x).b(x) \equiv 1 \ (mod\ p(x))$ has a unique solution which is the multiplicative inverse of $a(x)$ modulo $p(x)$. The congruence $a(x).f(x) \equiv b(x) \ (\mathrm{mod}\, p(x))$ where $(a(x), p(x)) = 1$ has also a unique solution.

Next, we consider the general problem of solving the congruence $a(x).f(x) \equiv b(x)$ $(\mathrm{mod}\, p(x))$ where $p(x)$ is a fixed polynomial. If $(a(x), p(x)) = d(x)$ then the congruence $a(x).f(x) \equiv b(x) \quad (\mathrm{mod}\ p(x))$ is solvable iff $d(x)/b(x)$. To know the solutions for a congruent relation use the following theorem.

Theorem 52 *Let $d(x) = (a(x), p(x))$. The congruence $a(x).f(x) \equiv b(x)(\mathrm{mod}\, p(x))$ is solvable iff $d(x)/b(x)$. If the congruence is solvable then the incongruent solutions are of the form $f(x) = f_0(x) + t(x).\dfrac{p(x)}{d(x)}$ where $t(x) \in A(d(x))$ and $f_0(x)$ is a solution of $\dfrac{a(x)}{d(x)}.f(x) \equiv \dfrac{b(x)}{d(x)} \ \mathrm{mod}\ \dfrac{p(x)}{d(x)}$.*

To avoid working with large polynomials, we can often choose a set of moduli $\{m_1(x), m_2(x), ..., m_i(x)\}$ and perform the computation modulo each of the moduli separately. Provided the moduli are pairwise relatively prime, $\gcd(m_i(x), m_j(x)) = 1$ for $1 \leq i < j \leq k$, and the product $m(x) = m_1(x).m_2(x)....m_i(x)$ is sufficiently large.

We may be able to construct the answer from the separate results. The following theorem, known as "Chinese remainder theorem", is helpful.

Theorem 53 *(Chinese remainder theorem for polynomials over finite fields) Let* $m_1(x), m_2(x)$. *be pairwise relatively prime polynomials over a field* F *and let* $a_1(x), a_2(x), ..., a_i(x) \in F[x]$. *The system of congruencies* $f(x) \equiv a_j(x)$ $(mod\ m_j(x))$, $1 \leq j \leq i$, *has a common solution which is unique modulo the product* $m(x) = m_1(x).m_2(x)....m_i(x)$.

Example 54 *Let us solve the system in* $\mathbf{Z}_5[x]$

$$f(x) \equiv 3x + 1 \qquad (mod\ (x^2 + x + 4))$$

$$f(x) \equiv x^2 + 3 \qquad (mod\ (x^3 + 2x^2 + 3x + 1))$$

Since $(x^2 + x + 4,\ x^3 + 2x^2 + 3x + 1) = 1$, *let* $m_1(x) = x^2 + x + 4$ *and* $m_2(x) = x^3+2x^2+3x+1$. *Thus,* $m(x) = (x^2+x+4)(x^3+2x^2+3x+1) = x^5+3x^4+4x^3+2x^2+3x+4$. *Hence,* $\dfrac{m(x)}{m_1(x)} = x^3+2x^2+3x+1$; $\dfrac{m(x)}{m_2(x)} = x^2+x+4$. *Solving* $\dfrac{m(x)}{m_1(x)}.b_1(x) \equiv 1$ $(mod\ (x^2+x+4))$, *we get* $b_1(x) = 3x+1$. *Solving* $\dfrac{m(x)}{m_2(x)}.b_2(x) \equiv 1$ $(mod\ (x^3+2x^2+3x+1))$, *we get* $b_2(x) = 2x^2 + x$.

$$\begin{aligned} \textit{Hence, } f_0(x) &= (x^3+2x^2+3x+1).(3x+1).(3x+1) + (2x^2+x)(x^2+x+4)(x^2+3) \\ &= 4x^5 + 4x^4 + 4x^2 + 4x + 1\ (mod\ m(x)) \\ &= 2x^4 + 3x^3 + x^2 + 2x. \end{aligned}$$

Example 55 *Let* $f(x) = 2x^2 + 2x + 1$ *in* $\mathbf{Z}_3[x]$. $f(x)$ *is irreducible in* $\mathbf{Z}_3[x]$. *Applying the above theorem , we have* $\dfrac{3^2 - 1}{2} = 4$ *quadratic residues and* 4 *quadratic nonresidues of the polynomial* $f(x)$ *in* $\mathbf{Z}_3[x]$.

Let $h(x)$ be an irreducible polynomial in $\mathbf{Z}_p[x]$ where p is an odd prime. To determine whether any polynomial $g(x)$ in the reduced residue system modulo $h(x)$

is a square or a quadratic residue of $h(x)$ in $\mathbf{Z}_p[x]$, we are in need of very important theorems which are Euler's theorem, Fermat theorem and Wilson theorem in the domain of polynomial rings.

Theorem 56 *(Euler's theorem for polynomials)*

Let F be a field with finite elements, and let $p(x)$ be an irreducible polynomial of degree $n > 1$. For every polynomial $a(x)$ relatively prime to $p(x)$, we have $a(x)^{\phi(p(x))} \equiv 1 (mod\ p(x))$, where $\phi(p(x))$ is the number of polynomials in $R(p(x))$.

A special case of the above theorem is the Fermat's theorem for polynomials.

Theorem 57 *(Fermat's theorem for polynomials)*

Let F be a finite field of order p. Let $p(x)$ be an irreducible polynomial of degree n in $F[x]$. For every polynomial $a(x)$ not divisible by $p(x)$, $a(x)^{p^n-1} \equiv 1\ (mod\ p(x))$.

Example 58 *Let $p(x) = x^2+1$ in $\mathbf{Z}_3[x]$. Let $a(x) = x+2$. Hence, $a(x)^{p^n-1} \equiv (x+2)^8 \equiv 1\ (mod\ (x^2+1))$.*

Theorem 59 *(Wilson's theorem over polynomials)*

Let p be an odd integer and let $h(x)$ be an irreducible in $\mathbf{Z}_p[x]$. $R(h(x))$ is the reduced residue system modulo $h(x)$. Then, $\prod_{f(x)\in R(h(x))} f(x) \equiv -1\ (mod\ h(x))$.

Finite Field. A finite field (also called a Galois field) is a finite set of elements together with the description of two operations (addition and multiplication). The order of a finite field is the number of field elements it contains. This number is always a prime or a power of prime. It turns out that there is a finite field containing q field elements if and only if q is a power of a prime number. The finite field containing q elements is denoted by F_q. The finite field of order p is the prime field F_p. It is the field of congruent classes modulo p where the p elements are the set of integers $\{0, 1, ...p-1\}$. For every prime p and every positive integer n, there is a finite field of order p^n. In fact, in a technical sense, there is essentially only one field with p^n elements since all fields of the same order are isomorphic. By isomorphic we mean evidently, they are structurally the same although the representation of their field elements may be different. Every field of order p is isomorphic to $\mathbf{Z}_p$. F_p will hence forth be identified with $\mathbf{Z}_p$. In every finite field, there is a positive integer n such that the sum of one n times is equal to zero. The minimal such n is called the characteristic of F denoted by $Char(F)$. For any finite field F, $char(F)$ is always prime number. The characteristic of F_q is p where $q = p^n$. Consequently, F_q contains a copy of $\mathbf{Z}_p$ .We say $\mathbf{Z}_p$ is a subfield of F_q or F_q is an extension field of $\mathbf{Z}_p$.Every subfield of F_q has order p^m for some m that is a positive divisor of n. Conversely If m is a positive divisor of n then, there is exactly one subfield of F of order p^m An element $a \in F_q$ is in the subfield F_{p^m} if and only if $a^{p^m} = a$.

The most concrete way to represent the field F_q where $q = p^n$ is to find an irreducible polynomial $g(x) \in F_p$ of degree n and this polynomial exists because it is well known that for each $n \geqslant 1$, $\exists$ a monic irreducible polynomial of degree n over $\mathbf{Z}_p$. In that case, it is plain to see by uniqueness that we must have $Fq \approx F_p[x]/ < g(x) >$. For example, $x^3 + x + 1$ is irreducible over F_2, hence $F_8 \approx F_2[x]/ < x^3 + x + 1 >$. $F_p[x]/ < g(x) >$ is a field for different reasons. Recall that all primes are maximal in a Euclidean ring and that $F_p[x]$ is Euclidean and also $g(x)$ is irreducible in $F_p[x]$. Thus, the elements of a finite field F_q of order $q = p^n$ where p is a prime are represented by polynomials in $\mathbf{Z}_p[x]$ (the set of all polynomials over $\mathbf{Z}_p$) of degree less than n. The finite field is exactly the set of all polynomials of degree 0 to $n-1$ with the two field operations being addition and multiplication of polynomials modulo $g(x)$ and with modulo p integer arithmetic on the polynomial coefficients. More precisely, the addition of two polynomials $f(x)$ and $h(x)$ is the usual addition of polynomials in $\mathbf{Z}_p[x]$ whereas the product $f(x).h(x)$ can be formed by first multiplying $f(x)$ and $h(x)$ as polynomials by ordinary method, and then taking the remainder after polynomial division by $g(x)$. The multiplicative inverses in the finite can be computed by using the extended Euclidean algorithm for the polynomial ring $\mathbf{Z}_p[x]$.We should note that since F_q is the splitting field of $x^q - x$, we can get an irreducible monic polynomial of degree n with coefficients in $\mathbf{Z}_p$ by factoring the polynomial $x^q - x$ There are other efficient algorithms for finding irreducible polynomials over finite fields.

Remark 60 *(Conversion from finite field elements to integers and to binary numbers) Let F_q be a finite field where $q = p^m$, p is a prime number and $m > 1$. Recall that F_q is isomorphic to $Z_p[x]/ < f(x) >$ where $f(x)$ is an irreducible polynomial of degree m over Z_p. Let $p(x) \in \mathbf{Z}_p[x]/ < f(x) >$, then $p(x)$ can be written as $p(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ...a_0$. Each coefficient $a_i \in Z_p$, and all arithmetic operations are performed modulo $f(x)$. Representing $p(x)$ as an integer, it will be equivalent to $a_{m-1}p^{m-1} + a_{m-2}p^{m-2} + ...a_0$. Representing $p(x)$ as binary string, replace each a_i in the string $(a_{m-1}.....a_1a_0)$ by $\lceil \log_2 p \rceil$. The whole new bit string will represent the polynomial $p(x)$.*

If F_q is a finite field of order $q = p^n$, the non-zero elements of F_q form a group under multiplication called the multiplicative group of F_q and denoted by F_q^*. The group of units F_q^* consists of the nonzero elements of F and forms a cyclic group under multiplication of order $q-1$. It follows that $a^q = a$ for all $a \in F^*$. Note that $F^* \cong \mathbf{Z}_{p^n-1}$. Then, there exists at least one element α of F^* called a generator (or a primitive element) of F^* which has the same order of F^*.Thus, $F^* =< \alpha >= \{e, \alpha, \alpha^2, ..., \alpha^{q-1}\}$. If a and b belong to a finite field of characteristic p, then $(a+b)^{p^t} = a^{p^t} + b^{p^t}$ for all $t \geq 0$. Also, if $G =< \alpha >$ is a cyclic group of order n then $G = < \alpha^k >$ iff $gcd(k, n) = 1$. As a result, one can find all generators of the cyclic group F^* given a generator α.The elements α^k of F^* with $(k, p^n - 1) = 1$ are the other generators. To find a generator α of the cyclic group F_q^* of order $q-1$, we select an element α in F_q^* with $\alpha \neq 1$. We use the division algorithm to write the order $q-1$ as a product of distinct primes p_i. α is a generator of F_q^* only if $\alpha^{\frac{q-1}{p_i}} \not\equiv 1 \ (mod \ (q-1))$ for every p_i divisor of $q-1$,. This statement is put in the following algorithm.

Algorithm 61 *(Finding a generator α of F_q^*)*

1. *Choose an integer $\alpha \neq 1$ and choose $q-1$ from F_q^*.*

2. *Write $q - 1 = p_1^{e_1} p_2^{e_2} ....p_k^{e_k}$.*

3. *For $i = 1$ to k*
   Compute $\alpha^{\frac{q-1}{p_i}}$ $(mod \ q)$
   If $\alpha^{\frac{q-1}{p_i}} \not\equiv 1 \ (mod \ q)$ then return to step 1.

4. *Return with α as a generator of F_q^*.*

In a similar manner, the above algorithm is applied to the ring of polynomials over finite fields to find generators $\alpha(x)$ of the cyclic groups $\mathbf{Z}_p^*[x] \cong \mathbf{Z}_p[x]/ < h(x) >$. Note that, the order of $\mathbf{Z}_p^*[x]$ is $q = p^n - 1$ where n is the degree of the monic irreducible polynomial $h(x)$. Hence the algorithm would be as follow.

Algorithm 62 *(Finding a generator $\alpha(x)$ of $\mathbf{Z}_p^*[x]$)*

```
1. Select a polynomial $\alpha(x) \neq 1$ from $\mathbf{Z}_p^*[x]$

2. Write $p^n - 1 = p_1^{e_1} p_2^{e_2} ...... p_k^{e_k}$.

3. For $i = 1$ to $k$
   Compute $\alpha(x)^{\frac{p^n - 1}{p_i}} \pmod{h(x)}$
   If $\alpha(x)^{\frac{p^n - 1}{p_i}} \not\equiv 1(mod\ h(x))$ in $\mathbf{Z}_p[x]$, then return to step 1.

4. Return with $\alpha(x)$ as a generator of $\mathbf{Z}_p^*[x]$.
```

To find the structure of the group of units $U(F[x]/ < h(x >)$ where $h(x)$ is a reducible polynomial in $F[x]$, the next theorem shows that it is enough to find the structure of the group of units $U(F[x]/ < g(x)^m >$ where $g(x)$ is irreducible over F.

Lemma 63 *If $h(x) = h_1(x)^{m_1}.h_2(x)^{m_2}....h_1(x)^{m_r}$ where all $h_i(x)$ are distinct irreducible polynomials in $F[x]$, then $F[x]/ < h(x) > \cong \dfrac{F[x]}{< h_1(x)^{m_1} >} X \dfrac{F[x]}{< h_2(x)^{m_2} >} ..... \dfrac{F[x]}{< h_r(x)^{m_r} >}$. Since the group of units of a direct sum of rings is the direct product of the group of units of the factors, then $U(F[x]/ < h(x) >) \cong U\left(\dfrac{F[x]}{< h_1(x)^{m_1} >}\right) X..... U\left(\dfrac{F[x]}{< h_r(x)^{m_r} >}\right)$*

From the above lemma, we deduce the following.

Lemma 64 *Let F be a finite field with order p^n, where p is an odd prime. Let $h(x)$ be a reducible polynomial. If $U(F[x]/ < h(x) >)$ is cyclic then $h(x)$ is a power of an irreducible polynomial $g(x)$ in $F[x]$.*

The structure of the irreducible polynomial $g(x)$ such that $g(x) = h(x)^m$, where $h(x)$ is a reducible polynomial, is illustrated in the following theorem.

Theorem 65 *If F is a finite field of order p^n, where p is an odd prime, and $g(x)$ an irreducible polynomial over $F[x]$. In case $U(F[x]/ < g(x)^m)$ is cyclic and $m > 1$ then the degree of $g(x)$ is equal to 1 and the field F is isomorphic to $\mathbf{Z}_p$.*

We can show that, if the degree of $g(x)$ is equal to 1 and $m > 1$ then $h(x)$ will be the square of the irreducible polynomial $g(x)$. Moreover, we can show that, if a is a root of the irreducible polynomial $g(x)$ such that $k = F(a)$ then $\dfrac{F[x]}{< g(x)^m >} \cong \dfrac{k[x]}{< x^m >}$.

We conclude this chapter by a theorem that state the conditions for the group of units $U(F[x]/ < h(x >)$ to be cyclic

Theorem 66 *Let F be a finite field with order p^n, where p is an odd prime.*

$U(F[x]/ < h(x) >)$ is cyclic iff

1. $h(x)$ is irreducible and $U\left(\dfrac{F[x]}{< h(x)}\right) \cong \mathbf{Z}_{p^n-1}$.
2. $h(x) = g(x)^2$ where $g(x)$ is linear and $U\left(\dfrac{F[x]}{< h(x) >}\right) \cong \mathbf{Z}_{p^n-1} X \ \mathbf{Z}_p$.

CHAPTER 3

A COMPARATIVE STUDY OF ELGAMAL BASED DIGITAL SIGNATURE ALGORITHMS

In this chapter, we compare and evaluate the classical and modified ElGamal algorithms by implementing and running them on a computer. We investigate the issues of complexity, efficiency and reliability by running the programs with different sets of data. Moreover, comparisons will be done among these different algorithms given the same data as input. In addition, implementation of an attack algorithm will be presented. The attack algorithm consists of subroutines used to verify the signed messages. This is done by applying certain mathematical concepts to find the private key of the signed message. After finding the key, it will be easy to sign the message. A study will be done using the results of running the attack algorithm to compare the security of the different classical and modified signature scheme algorithms.

3.1 Classical And Modified ElGamal Signature

Classical ElGamal Signature Scheme The classical ElGamal signature scheme is one of the most popular and widely used signature scheme. It is described in the setting of the multiplicative group Z_p^* of the field Z_p, the field of integers modulo a prime integer p. Let p be a large odd prime integer and let $Z_p = \{0, 1, 2, 3, ..., p-1\}$. Then Z_p is a ring under addition and multiplication modulo p. Since p is prime, Z_p is actually a field under these operations. Moreover, the multiplicative group of the ring of integers modulo p, $Z_p^* = \{1, 2, 3, ..., p-1\}$, is a cyclic group generated by some generator $\theta \neq 1$ whose order is equal to $p-1$. That is, every element of Z_p^* is a power of θ. Note that

Z_p is a complete residue system modulo p and Z_p^* is a reduced residue system modulo p.

The key is generated as follows.First generate a large random prime p, and a generator θ of Z_p^*. Then, choose randomly an integer a, $1 \leq a \leq p-2$, and compute $y = \theta^a (\bmod p)$. The public-key is (p, θ, θ^a) and the private-key is a. The signature is generated as follows. First, hash the message then compute the hash value $f = h(m)$ where h is a hash function.Generate a random secret integer k, such that $1 \leq k \leq p-2$ with $\gcd(k, p-1) = 1$. Compute $r = \theta^k$ $(\bmod p)$ and compute k^{-1} $(\bmod (p-1))$. Then,compute $s = k^{-1}\{h(m) - a.r\}$ $(\bmod (p-1))$. Entity A then sends m and the signature (r, s).

The signature is validated as follows.Obtain A's authentic public key (p, θ, θ^a) and verify that $1 \leq r \leq p-1$.Hash the message m and obtain the hash value $f = h(m)$. Then, compute $v_1 = y^r r^s (\bmod p)$ and $v_2 = \theta^{h(m)} (\bmod p)$.Accept the signature only if $v_1 = v_2$.

The following algorithms show the functionality of the ElGamal signature :

Algorithm 67 *(Key generation for ElGamal classical signature).*

Entity A should do the following:

1- Generate a large random prime p and generator θ of $\mathbf{Z}_p^*$.

2- Select a random integer a, $1 \leq a \leq p-2$.

3- Compute $y = \theta^a (\bmod p)$

4- A's public key is (p, θ, θ^a); A's private key is a.

The following algorithm shows how entity A signs a message m for B.

Algorithm 68 *(ElGamal classical signature).*

Entity A should do the following:

1- Select a random secret integer k, $1 \leq k \leq p-2$ such that $\gcd(k, p-1) = 1$.

2- Represent the message as an integer m in the range $\{0, 1, ..., p-1\}$.

3- Compute $r = \theta^k \pmod p$.

4- Compute $k^{-1} \ (\mathrm{mod}(p-1))$.

5-Compute $s = k^{-1}\{h(m) - a.r\} \ (\mathrm{mod}(p-1))$.

6- A's signature for m is the pair (r, s).

The following algorithm shows how entity B verify that the message m is from A.

Algorithm 69 *(ElGamal classical signature verification).*

B should do the following:

1-Obtain A's authentic public key (p, θ, θ^a).

2- Make sure that $1 \leq r \leq p-1$.

3- Compute $v_1 = y^r r^s \pmod p$.

4- Compute $h(m)$ and $v_2 = \theta^{h(m)} \pmod p$.

5-Accept the signature only if $v_1 = v_2$.

Example 70 *In order to generate the public-key, entity A selects the prime* $p = 367$ *and finds a generator* $\theta = 272$ *of* Z_{367}^*. *Then, A chooses the private-key* $a = 141$ *and computes* $\theta^a = 272^{141} \equiv 295 (\mathrm{mod}\, 367)$. *Therefore, A's public-key is* $(p = 367, \theta = 272, \theta^a = 295)$ *and A's private-key is* $a = 141$.

To sign the message $m = 214$ *chosen from* Z_{359}, *for simplicity , we take* $h(m) = m$ *so that* h *is the identity function.Then,* $f = h(214) = 214$. *Selects a random integer* $k = 43$ *and computes*

$$r = 272^{43} \equiv 330 (\bmod 367)$$

Also, k^{-1} *is obtained from*

$$43.k^{-1} \equiv 1 (\bmod 366).$$

Hence,

$$k^{-1} = 349.$$

Then,

$$s = 349\{214 - (141).(330)\} \equiv 106 (\bmod 366)$$

Thus, A's signature for 214 *is then* $(r = 330,\ s = 106)$

To validate the signature , B make sure that $1 \leq r \leq 366$. *then, obtain the hashed message* $h(m) = m$ =214. *Finally,B computes*

$$v_1 = 295^{330}.330^{106} \equiv 182 (\bmod 367)$$

and

$$v_2 \equiv 272^{214} \equiv 182\ (\bmod 367).$$

Since $v_1 = v_2$, *the signature is accepted.*

Computer Program: (* We create a computer program with mathematica to illustrate the classical ElGamal signature*)

```
(*Key generation*)

FindGenerator[p_] := (ϕ = p - 1;
j = FactorInteger[ϕ]; z = {};
Do[z = Append[ z, j[[i]][[1]]], {i, 1, Length[j]}];
flag = 0; While[flag == 0,θ= Random[Integer, {2, p - 1}];
flag = 1;
Do[If[ PowerMod[θ, ϕ/z[[i]], p] == 1, flag = 0; Break[]], {
i, 1, Length[z]}]];
Print["θ= ",θ];
a = Random[Integer, { 2, ϕ - 1}];
θa = PowerMod[θ, a, p];)
(*Signature generation*)
Signature[p_] := ( k = 2;
While[GCD[k,ϕ] ≠ 1, k = Random[Integer, {2, ϕ - 1}]];
r = PowerMod[θ, k, p]; hm = m;
k1 = PowerMod[k, -1, ϕ];
s = Mod[k1*(hm - r*a),ϕ];
Print[ "r=", r, "s=", s])
(*Signature Verification*)
Verification[p_] := (v1 = Mod[PowerMod[θa, r, p]*PowerMod[r, s, p], p];
hm = m; v2 = PowerMod[θ, hm, p];

Print["v1=", v1, " v2= ", v2];

If[ v1 == v2, Print["accc"], Print["not acc"]])
```

The classical ElGamal signature scheme, described in the setting of the multiplicative group Z_p^*, can be easily generalized to work in any finite cyclic group G. The security of the generalized ElGamal encryption scheme is based on the intractability of the discrete logarithm problem in the group G. The group G should be carefully chosen so that the group operations in G should be relatively easy to apply for efficiency and the discrete logarithm problem in G should be computationally infeasible for the security of the protocol that uses the ElGamal signature scheme.

Menezes [22] mentioned that some of the groups that appear to meet the above criteria, of which the first three have received the most attention, are the multiplicative group Z_p^* of the integers modulo a prime p, the multiplicative group $F_{2^m}^*$ of the finite field F_{2^m} of characteristic two, the group of points on an elliptic curve over a finite field, the multiplicative group F_q^* of the finite field F_q, where $q = p^m$, p is a prime, the group of units Z_n^*, where n is a composite integer, the Jacobian of a hyperelliptic curve defined over a finite field, and the class group of an imaginary number field.

For any of the above cases used to generalize ElGamal signature scheme, the following procedures are followed: To generate the public-key, entity A should select an appropriate cyclic group G of order n, with generator θ. Assuming that G is written multiplicatively, a random integer a, $1 \leq a \leq n-1$, is selected and the group element θ^a is computed. $A's$ public-key is $(\theta, y = \theta^a)$, together with a description of how to multiply elements in G. $A's$ private-key is a. To sign a message m in the cyclic group

G, entity A selects a random integer k, $1 \leq k \leq n-1$,with $\gcd(k, \phi(n)) = 1$ and compute $r = \theta^k \pmod{n}$ and $k^{-1} \pmod{\phi(n)}$. Then,compute $s = k^{-1}\{h(m) - a.r\}$ $\pmod{\phi(n)}$.Finally, A sends m and the signature (r, s) to entity B. To validate the signature, B make sure that $r \in U(Z_n)$ .Then, obtain the hash value $f = h(m)$.Finally, B computes $v_1 = y^r r^s \pmod{n}$ and $v_2 = \theta^{h(m)} \pmod{n}$.Accept the signature only if $v_1 = v_2$.

Next, we describe the modifications of ElGamal signature scheme in Z_n, where n is a composite integer, the domain of Gaussian integers $Z[i]$, and the domain of the rings of polynomials over finite fields $F[x]$.

ElGamal Signature in Z_n. In this scheme, the group G selected is Z_n^*, where n is not necessarily a prime. Since ElGamal signature scheme depends on the fact that the group selected is cyclic, it is important to know the positive integers n for which the multiplicative group Z_n^* of integer modulo n is cyclic. Although the ring Z_n modulo a composite integer is not a field, it is well known, that the multiplicative group Z_n^* is cyclic if and only if n is either 2, 4, p^t, or $2p^t$, where p is an odd prime and $t \geq 1$, see [20]. The order of Z_n^* is $\phi(n) = (p-1)p^{t-1}$, where $\phi(n)$ is Euler phi function. We note that if θ is a generator of Z_n^*, then $Z_n^* = \{\theta^i \mid 0 \leq i \leq \phi(n) - 1\}$ and $b = \theta^k \pmod{n}$ is also a generator of Z_n^* if and only if $(k, \phi(n)) = 1$.

To describe the generalized scheme used in the above operation, three algorithms describing the modified ElGamal method restricted to the multiplicative group Z_n^* are needed.

To generate the public and private-keys, entity A should use the following algorithm:

Algorithm 71 *(Key generation for ElGamal signature in Z_n.)*

1. *Generate a large random odd prime p and a positive integer t.*
2. *Compute the composite integer n ($n = p^t$ or $n = 2p^t$) and the integer $\phi(n) = (p-1)p^{t-1}$.*
3. *Find a generator θ of the cyclic multiplicative group of integers modulo n, $Z_n^* = \{\theta^i \mid 0 \le i \le \phi(n) - 1\}$.*
4. *Select a random integer a, $2 \le a \le \phi(n) - 1$.*
5. *Compute $y = \theta^a (\bmod n)$.*
6. *A's public-key is $(n,\ \theta,\ \theta^a)$.*
7. *A's private-key is a.*

Note that the integer a in step 4 is chosen to be between 2 and $\phi(n) - 1$ since a is a power of the generator θ of Z_n^*.

To sign a message m in Z_n, the complete residue system modulo n, Entity A should use the following algorithm:

Algorithm 72 *(ElGamal signature in Z_n.)*

1. *Represent a message as an integer m in the range $\{1, ..., n-1\}$.*
2. *Select a random integer k, $2 \le k \le \phi(n) - 1$ such that $\gcd(k, \phi(n)) = 1$.*

To sign the message $m = 7$ *chosen from* Z_{18818}, *for simplicity , we take* $h(m) = m$ *so that* h *is the identity function. Then,* $f = h(7) = 7$. *Selects a random integer* $k = 5$ *and computes*

$$r = 13^5 \equiv 13751 (\bmod 18818)$$

Also, k^{-1} *is obtained from*

$$5.k^{-1} \equiv 1(\bmod(\phi(18818) = 9312)).$$

Hence,

$$k^{-1} = 3725$$

Then,

$$s = 3725\{7 - (4246).(13751)\} \equiv 3625 (\bmod 9312)$$

Thus, A's signature for 7 *is then* $(r = 13751,\ s = 3625)$

To validate the signature , B make sure that $r \in U(Z^*_{18818})$. *then, obtain the hashed message* $h(m) = m$ =7. *Finally,B computes*

$$v_1 = 15135^{13751}.13751^{9312} \equiv 9305 (\bmod 18818)$$

and

$$v_2 \equiv 13^7 \equiv 9305 \ (\bmod 18818).$$

Since $v_1 = v_2$, *the signature is accepted.*

Note that there are 18817 *values for* m *you can choose from the complete residue system modulo* 18818*,* Z_{18818}.

Computer Program: (* We create a computer program with mathematica to illustrate the modified ElGamal signature in Z_n *)

```
(*Key generation*)

FindGeneratorPower[p_] := ( φ = p^2 - p;
j = FactorInteger[φ]; z = {};
Do[z = Append[z, j[[ i]][[1]]], {i, 1, Length[j]}];
flag = 0;
While[flag == 0, θ = 2;
While[Mod[θ, 2] == 0 || Mod[θ, p] == 0,
θ = Random[Integer, { 2, 2 p^2 - 1}]];
flag = 1;
Do[If[PowerMod[θ, φ/z[[i]], 2p^2] == 1, flag = 0; Break[]], {i, 1, Length[z]}]];
Print["θ= ",θ];
a = Random[Integer, {2, φ - 1}];
θa = PowerMod[θ, a, 2 p^2])
(*Signature generation*)
SignaturePower[p_] := (k = 5; While[GCD[k, φ] ≠ 1,
k = Random[Integer, {2, φ - 1}]];
r = PowerMod[θ, k, 2 p^2];
hm = m; k1 = PowerMod[k, -1, φ];
s = Mod[k1*(hm - r*a), φ]; Print["r=", r, " s=", s])

(*Signature Verification*)

VerificationPower[p_] := (

v1 = Mod[ PowerMod[θa, r, 2 p^2]*PowerMod[r, s, 2 p^2], 2p^2];

hm = m; v2 = PowerMod[θ, hm, 2p^2];

Print["v1= ", v1, " v2= ", v2]; If[v1 == v2, Print["accc"], Print["not acc"]])
```

ElGamal Signature in the Domain of Gaussian Integers El-kassar et al. [5] extended ElGamal signature from the domain of integers to the domain of Gaussian integers as follows: First, a Gaussian prime β is chosen. Let G_β be a complete residue system modulo β. Then, G_β is a field under addition and multiplication modulo the Gaussian integer β. If $\beta = \pi$, where $q = \pi\bar{\pi}$ is prime integer of the form $4k+1$, then $G_\beta = \{a : 0 \leq a \leq q-1\} = Z_q$, see [2]. This choice will be excluded since the calculations will be identical to those of the classical case. Hence, β is chosen to be a large prime integer p of the form $4k+3$ so that $G_\beta = \{a+bi : 0 \leq a \leq p-1, 0 \leq b \leq p-1\}$, see [2]. The number of elements in G_β is p^2 and in G_β^*, its multiplicative group of units, is $\phi(\beta) = p^2 - 1$. Hence, the cyclic group used in the extended ElGamal signature has an order larger than the square of that used in the classical ElGamal signature with no additional efforts required for finding the prime p. Now, a generator of θ of G_β^* is chosen. Note that there are $\phi(p^2-1)$ generators in G_β^*. A random positive integer a is then chosen so that the public key is (p, θ, θ^a). Since a is a power of θ, then a must be less than the order of the group power G_β^* which is p^2-1. This power, a, is the private key. To sign a message, first hash the message to obtain the hash value $f = h(m)$ .Then, select a random secret integer k, such that $1 \leq k \leq p^2 - 2$ with $\gcd(k, p^2 - 1) = 1$ and compute $r = \theta^k \pmod p$ .Note that r is in $G_\beta = \{a + bi : 0 \leq a \leq p - 1, 0 \leq b \leq p - 1\}$. Next, compute k^{-1} $(\mathrm{mod}(p^2 - 1))$ and $s = k^{-1}\{h(m) - a.k\}$ $(\mathrm{mod}(p^2 - 1))$. Also, compute $\delta = r^a (\mathrm{mod}\, \beta)$. Send the message m and the signature (r, δ, s).

To validate the signature ,obtain the authentic public key (p, θ, θ^a) and verify that $r \in G^*_\beta$.Hash the message m and obtain the hash value $f = h(m)$. Compute $v_1 = \delta r^s (\mathrm{mod}\, \beta)$ and compute $v_2 = \theta^{h(m)} (\mathrm{mod}\, \beta)$.Accept the signature only if $v_1 = v_2$.

We note that the reduction modulo a Gaussian integer requires computational procedures that are more involved than those used in the reduction modulo an integer. However, since β was chosen to be a prime integer $p = 4k + 3$, then the reduction modulo β do not require computational procedures that are different from those used for the integers. In fact, to reduce $a + bi$ modulo β, we find c, d with $0 \leq c, d \leq p - 1$ such that $c \equiv a (\mathrm{mod}\, p)$ and $d \equiv b (\mathrm{mod}\, p)$. Then $c + di \in G_\beta$ and $c + di \equiv a + bi (\mathrm{mod}\, \beta)$. Hence, the reduction modulo β in $Z[i]$ is done using integer reductions.

Algorithm 75 *(Key generation for ElGamal Gaussian signature).*

Entity A should do the following:

1- Generate a large random Gaussian prime integer β of the form $p = 4.k + 3$,where p is a prime integer.

2-Find a generator θ of the multiplicative group G^*_β.

3- Select a random integer a, $1 \leq a \leq p^2 - 2$.

4- Compute $y = \theta^a (\mathrm{mod}\, \beta)$

5- A's public key is $(\beta, \theta, \theta^a)$; A's private key is a.

The following algorithm shows how entity A signs a message m for B.

Algorithm 76 *(ElGamal Gaussian signature).*

Entity A should do the following:

1- Select a random secret integer k, $1 \leq k \leq p^2 - 2$ such that $\gcd(k, p^2 - 1) = 1$.

2- Represent the message as an integer m and compute $h(m)$.

3- Compute $r = \theta^k \pmod p$.

4- Compute $k^{-1} \pmod{(p^2 - 1)}$.

5-Compute $s = k^{-1}\{h(m) - a.k\} \pmod{(p^2 - 1)}$.

6-Compute $\delta = r^a \pmod p$.

7- A's signature for m is the triplet (r, s, δ).

The following algorithm shows how entity B verify that the message m is from A.

Algorithm 77 *(ElGamal Gaussian signature verification).*

B should do the following:

1-Obtain A's authentic public key (p, θ, θ^a).

2- Make sure that $r \in G^*_\beta$, otherwise reject the signature.

3- Compute $v_1 = \delta r^s \pmod \beta$.

4- Compute $h(m)$ and $v_2 = \theta^{h(m)} \pmod \beta$.

5-Accept the signature only if $v_1 = v_2$.

Example 78 *In order to generate the public-key, entity A selects the Gaussian prime $p = 479$ and finds a generator $\theta = 398 + 327i$ of G^*_{479}. Then, A chooses the private-key*

$a = 21506$ *and computes* $\theta^a = (398 + 327i)^{21506} \equiv 461 + 372i (\bmod 479)$. *Therefore, A's public-key is* $(p = 479, \theta = 398 + 327i, \theta^a = 461 + 372i)$ *and A's private-key is* $a = 21506$.

To sign the message $m = 214$, *calculate* $f = h(214) = 214$. *Selects a random integer* $k = 13$ *and computes*

$$r = (398 + 327i)^{13} \equiv 416 + 447i (\bmod 479)$$

Also, k^{-1} *is obtained from*

$$13.k^{-1} \equiv 1(\bmod (479^2 - 1)).$$

Hence,

$$k^{-1} = 70597.$$

Then,

$$s = 70597\{214 - (13).(21506)\} \equiv 172652(\bmod (479^2 - 1))$$

Thus, A's signature for 214 *is then* $(r = 416 + 447i,\ s = 172652)$

To validate the signature , B make sure that $r \in G^*_{479}$ *then, obtain the hashed message* $h(m) = m$ =214. *Finally,B computes*

$$v_1 = (416 + 447i)^{172652}.(461 + 372i)^{13} \equiv 296 + 335i (\bmod 479)$$

and

$$v_2 \equiv (398 + 327i)^{214} \equiv 296 + 335i\ (\bmod 479).$$

Since $v_1 = v_2$, the signature is accepted.

Computer Program: (* We create a computer program with mathematica to illustrate the modified ElGamal signature in $Z[i]$ *)

```
    (*Key generation*)
FindGeneratorGaussian[p_] := (ϕ = p^2 - 1;
j = FactorInteger[ϕ]; z = {};
Do[z = Append[z, j[[i]][[1]]], {i, 1, Length[j]}];
flag = 0;
While[flag == 0,
θ = Random[Integer, {2, p - 1}] +I*Random[Integer, {2, p - 1}];
flag = 1;
Do[If[PowerMod[θ, ϕ/z[[i]], p] == 1, flag = 0; Break[]], {i, 1, Length[z]}]];
Print["θ= ", θ];
a = Random[Integer, {2,ϕ -1}];
θa = PowerMod[θ, a, p];)
(*Signature generation*)
SignatureGaussian[p_] := (k =13; While[GCD[k, ϕ] ≠ 1; k = Random[Integer,
{2,ϕ- 1}]];
Print["k= ", k]; r = PowerMod[θ, k, p];
k1 = PowerMod[k, -1,ϕ];
Print["k1= ", k1]; hm = m;
s =Mod[k1*(hm - k*a),ϕ];
δ = PowerMod[r, a, p];

Print["signature ", r, "     ", s, "     ", δ];)

(*SignatureVerification*)

VerificationGaussian[ p_] := (v1 = Mod[■*PowerMod[r, s, p], p];

hm = m; v2 =PowerMod[θ, hm, p];

Print["v1= ",v1, "v2= ", v2];

If[v1 ==v2, Print["accc"], Print["not acc"]])
```

The following are some of the advantages of ElGamal signature scheme in $\mathbb{Z}[i]$. Using arithmetics in the domain of Gaussian integers $\mathbf{Z}[i]$, ElGamal signature scheme was extended to $\mathbf{Z}[i]$. The computational procedures in the new setting were described and the advantages of the new scheme were pointed out. The following are some of these advantages. First, generating a prime p in both the classical and the modified method requires the same amount of effort. However, the cyclic group $\mathbf{Z}_p^*$ has $p-1$ elements; while the cyclic group G_β^* has $p^2\ -1$. Second, the choice for the power a in the classical method is from 1 to $p-1$, while the choice of this power in the modified method is from 1 to $p^2\ -1$. Hence, the modified method provide more security since the security of ElGamal scheme depends on the discrete logarithm. Also, the number of elements that can be chosen to represent the hash of the message is more than the square of that used in the classical case. The computations involved in the modified method do not require computational procedures that are different from those used in the classical method.

ElGamal Signature over Finite Fields The generalized ElGamal signature in the setting of a finite field F_q, where $q = p^n$ for an odd prime integer p and a positive integer n, is based on working with the quotient ring $Z_p[x]/\langle h(x)\rangle$, where $h(x)$ is an irreducible polynomial over $Z_p[x]$. We extend the ElGamal signature to the setting of a finite field. It is well known that $Z_p[x]/\langle h(x)\rangle$ is a field whose elements are the congruence classes modulo $h(x)$ of polynomials in $Z_p[x]$ with degree less than n. We identify this field by the complete residue system $A(h(x)) = \{a_0 + a_1x + ... + a_{n-1}x^{n-1} \mid a_0, a_1, ..., a_{n-1} \in Z_p[x]\}$. Note that $Z_p[x]/\langle h(x)\rangle$ is of order p^n and its nonzero elements form a cyclic group denoted by $U(Z_p[x]/\langle h(x)\rangle)$. The order of $U(Z_p[x]/\langle h(x)\rangle)$ is $\phi(h(x)) = p^n - 1$. Let $\theta(x)$ be a generator of the cyclic group $U(Z_p[x]/\langle h(x)\rangle)$. The elements in $U(Z_p[x]/\langle h(x)\rangle)$ can be written as a power of the generator $\alpha(x)$. Hence, $U(Z_p[x]/\langle h(x)\rangle) = \{e, \theta(x), \theta(x)^2, ..., \theta(x)^{p^n-1}\}$.

Algorithm 79 *(Key generation for ElGamal signature over finite fields).*

Entity A should do the following:

1- Selecting an odd prime p and generating a monic irreducible polynomial $h(x)$ of degree n in $Z_p[x]$.

2-Finding a generator $\theta(x)$ of the multiplicative group $U(Z_p[x]/\langle h(x)\rangle)$.

3- Selecting a random integer a, $1 \leq a \leq \phi(h(x)) - 1$.

4- Computing $y(x) = \theta(x)^a (\bmod h(x))$

5- Publishing $(p, h(x), \theta(x))$ as public key and keeping a as private key.

The following algorithm shows how entity A signs a message m for B.

Algorithm 80 *(Signature generation of ElGamal signature over finite fields).*

Entity A should do the following:

1- Represent the message as a polynomial $m(x)$.

2- Select a random secret integer k, $1 \leq k \leq \phi(h(x))-1$ *such that* $\gcd(k, \phi(h(x))=1$.

3- Compute $h(m(x))$.

4- Compute $r(x) = \theta(x)^k \pmod{h(x)}$.

5- Compute $k^{-1} \pmod{\phi(h(x))}$.

6-Compute $s = k^{-1}\{h(m(x)) - a.k\} \pmod{\phi(h(x))}$.

7-Compute $\delta(x) = r(x)^a \pmod{h(x)}$.

8- Send $(r(x), s, \delta(x))$ *to* B.

The following algorithm shows how entity B verify that the message m is from A.

Algorithm 81 *(Signature verification of ElGamal signature over finite fields).*

B should do the following:

1-Obtain A's authentic public key .

2- Make sure that $r(x) \in U(Z_p[x]/\langle h(x)\rangle)$, *otherwise reject the signature.*

3- Compute $v_1(x) = \delta(x)r(x)^{s(x)} \pmod{h(x)}$.

4- Compute $h(m(x))$ *and* $v_2(x) = \theta(x)^{h(m(x))} \pmod{h(x)}$.

5-Accept the signature only if $v_1(x) = v_2(x)$.

Example 82 *(with small parameters). Consider the polynomial* $f(x) = x^3 + 3x + 2$ *which is irreducible over* $\mathbf{Z}_5$. *Hence the quotient ring* $Z_5[x]/ < x^3 + 3x + 2 >$ *is a finite field of order* $5^3 = 125$, *and where multiplication is performed modulo the irreducible polynomial* $f(x)$. *So* $\mathbf{Z}_5[x]/ < x^3 + 3x + 2 > -\{0\}$ *is of order* 124 *and one of the* *generators is the polynomial* $\alpha(x) = 3x^2 + 3x + 2$. *In order to generate the public key, entity A selects the private key* $a = 46$ *and computes* $y(x) = (3x^2 + 3x + 2)^{46} = 4x^2+3x+3$ *(mod* $f(x)$*) in* $\mathbf{Z}_5[x]$. *Therefore, A's public key is* $(5, x^3+3x+2, 3x^2+3x+2, 4x^2+3x+3)$. *To sign a message* $m(x) = 4$, *suppose the hash algorithm is the identity (for simplicity). Afterwards, A selects* $k = 11$ *with gcd*$(5, 124) = 1$ *and computes* $r(x) = (3x^2+3x+2)^{11} = x^2+x+3 \pmod{f(x)}$. *Also* k^{-1} *is obtained from* $11k^{-1} \equiv 1$ *(mod* 124*). Hence,* $k^{-1} = 79$. *Then, A computes* $s = 79\{4-(46).(11)\} \pmod{124} = 22$ *and* $\delta(x) = (x^2+x+3)^{46} = 2x$ *.Finally, A sends the signature to B. To validate the signature, B first makes sure that* $r(x) \in U(Z_5[x]/ < f(x) >)$. *Then, B calculate* $h(4) = 4$ *(supposing h identity function). Finally, B computes* $v_1 = (2x)(3x^2+3x+2)^{22} = x^2 + 3x + 4$.*and* $v_2 = (3x^2+3x+2)^4 = x^2+3x+4$. *Since* $v_1 = v_2$, *the signature is accepted.*

Computer Program: (* We create a computer program with mathematica to illustrate the modified ElGamal signature over finite fields*)

```
(*Key generation*)

FindGeneratorPolyirr[p_, n_] := (h = IrreduciblePolynomial[x, p, n];

Print["h=", h];ϕ = p^n - 1; Print["ϕ=",ϕ];

j = FactorInteger[ϕ];

z = {};

Do[z = Append[z, j[[ i]][[1]]], {i, 1, Length[j]}];

flag = 0;
While[flag == 0, θ = 0; Do[θ = θ + Random[Integer, { 1, p - 1}]*x^i, {i, 0, n -
1}];
While[PolynomialGCD[θ, h] ≠ 1, θ = 0;
Do[θ= θ+ Random[Integer, {0, 1}]*x^i, {i, 0, n - 1}]];
flag = 1;
Do[If[ PolynomialPowerMod[θ, ϕ/z[[ i]], {h, p}] == 1, flag = 0; Break[]], {i, 1,
Length[ z]}]];
Print[ "generator is =", θ]; a = Random[Integer, {2, ϕ - 1}];
θa = PolynomialPowerMod[θ, a, {h, p}]);
(* Signature generation*)
SignaturePolyirr[p_, n_] := (k = 2;
While[GCD[k,ϕ] ≠ 1, k = Random[ Integer, {2, ϕ - 1}]];
hm = m; r = PolynomialPowerMod[θ, k, {h, p}];
k1= PowerMod[k, -1, ϕ];
s = Mod[k1*(hm - k*a), ϕ];
δ = PolynomialPowerMod[r, a, {h, p}];
Print["signature is ", r, " ", s, "", δ ])
(* Signature Verification*)
VerificationPolyirr[p_, n_] := (If[PolynomialGCD[r, h] == 1,
v1 = PolynomialMod[PolynomialPowerMod[r, s, {h, p}]*δ, {h, p}]];
hm = m; v2 = PolynomialPowerMod[θ, hm, {h, p}];
Print[v1, "", v2]; If[v1 == v2, Print["accc"], Print["not acc"]])
```

Using the arithmetics in the domain of polynomial rings, ElGamal signature scheme was extended from the domain of natural integers $\mathbf{Z}$ to the ring of polynomials over finite fields, $\mathbf{Z}_p[x]/ < f(x) >$ and $f(x)$ is an irreducible polynomial over $\mathbf{Z}_p[x]$ of degree n, using the required arithmetics for this extension. The computational procedures in the new setting were described and the advantages of the new scheme were pointed out. The following are some of the advantages. First, The classic group $\mathbf{Z}_p^*$ has p elements, and the cyclic group $(\mathbf{Z}_p[x]/ < f(x) >)^*$ has $p^n -1$. Second, the choice for the power a in the classical method is from 1 to $p - 1$, while the choice of this power in the extended method is from 1 to $p^n - 1$. Hence, the extended method provides more security since the security of ElGamal signature scheme depends on the discrete logarithm. Also, the number of elements that can hash the message m is more than the n^{th} power of that used in the classical case. Moreover, note that finding an irreducible polynomial in $\mathbf{Z}_p[x]$ is not an easy problem. From this point of view, in the next section , ElGamal signature scheme in the ring of polynomials will be modified over quotient rings where $p(x)$ is not provided to be irreducible.

ElGamal Signature over Quotient Rings of Polynomials The ElGamal public-key signature is also extended in the setting of the cyclic group of the finite quotient ring $Z_p[x]/\langle f(x)\rangle$, where p is an odd prime, and $f(x)$ is a reducible polynomial of degree n over $Z_p[x]$, see [7]. In this case the ring $Z_p[x]/\langle f(x)\rangle$ is not a field. But according to ElGamal public-key signature scheme, we are only interested in the cyclic groups of units of such rings. Hence, throughout this section we are dealing with any quotient ring $Z_p[x]/\langle f(x)\rangle$ of order p^n, where p is an odd prime and n is the degree of the reducible polynomial $f(x)$. Using the characterization of the structure of the group unit of $Z_p[x]/\langle f(x)\rangle$ given in [29], El-Kassar et Haraty [11] obtained a characterization of all primes p and polynomials $f(x)$ for which $Z_p[x]/\langle f(x)\rangle$ is cyclic. Two particular cases of interest are as follows. For any finite field F of order $q = p^n$, where p is an odd prime integer, the group of units $U(F[x]/ < f(x) >)$ is cyclic and isomorphic to Z_{q-1} if and only if $f(x)$ is linear. Also, $U(F[x]/ < f(x) >)$ is cyclic and isomorphic to $Z_{p-1} \times Z_p$ if and only if $f(x) = h(x)^2$, where $h(x)$ is linear. Hence, we conclude that in order for the group of units $U(Z_p[x]/\langle h(x)\rangle)$ to be cyclic, $h(x)$ must be irreducible or a square power of only one linear irreducible polynomial. That is, $h(x) = h_1(x)^2$, where $h_1(x) = ax + b$. This means that $U(Z_p[x]/\langle (ax+b)^2\rangle)$ is cyclic. Moreover, we have that $Z_p[x]/\langle (ax+b)^2\rangle \cong Z_p[x]/\langle x^2\rangle$. Hence, we can say that the extension of the ElGamal scheme in this case applies to the group of units of the ring $Z_p[x]/\langle x^2\rangle$, of order $\phi(x^2) = p^2 - p$. We note that a polynomial $f(x)$ in $Z_p[x]$ belongs to the cyclic group $U(Z_p[x]/\langle x^2\rangle)$ if and only if $(f(x), x) = 1$. This is equivalent to saying that x does not divide $f(x)$, where $f(x)$ is a linear polynomial. Hence, $U(Z_p[x]/\langle x^2\rangle) = \{c + dx \mid 1 \leq c \leq p-1, 0 \leq d \leq p-1\}$.

Algorithm 83 *(Key generation for ElGamal signature over polynomials).*

Entity A should do the following:

1- Selecting an odd prime p and a reducible polynomial $f(x)$ of degree n in $Z_p[x]$ as a square of a linear polynomial.

2- Computing $\phi(x^2) = p^2 - p$.

3-Finding a generator $\theta(x)$ of the multiplicative group $U(Z_p[x]/\langle x^2\rangle)$. Therefore,

$U(Z_p[x]/\langle x^2\rangle) = \{e, \theta(x), \theta(x)^2, ..., \theta(x)^{p(p-1)-1}\}$.

4- Selecting a random integer a, $1 \leq a \leq \phi(x^2) - 1$.

5- Computing $y(x) = \theta(x)^a (\bmod x^2)$

6- Publishing $(p, h(x), \theta(x))$ as public key and keeping a as private key.

To generate a signature of a message, entity A should do the following.

Algorithm 84 *(Signature generation of ElGamal signature over polynomials).*

Entity A should do the following:

1- Represent the message as a polynomial $m(x)$.

2- Select a random secret integer k, $1 \leq k \leq p(p-1) - 1$ such that $\gcd(k, p(p-1)) = 1$.

3- Compute $h(m(x))$.

4- Compute $r(x) = \theta(x)^k \ (\bmod x^2)$.

5- Compute $k^{-1} \ (\bmod(p^2 - p))$.

6-Compute $s = k^{-1}\{h(m(x)) - a.k\} \ (\bmod(p^2 - p))$.

7-Compute $\delta(x) = r(x)^a (\bmod x^2)$.

8- Send $(r(x), s, \delta(x))$ to B.

The following algorithm shows how entity B verify that the message m is from A.

Algorithm 85 *(Signature verification of ElGamal signature over polynomials).*

B should do the following:

1-Obtain A's authentic public key $(p, f(x), \theta(x), y(x))$

2- Make sure that $r(x) \in U(Z_p[x]/\langle x^2\rangle)$, otherwise reject the signature.

3- Compute $v_1(x) = \delta(x)r(x)^s (\bmod x^2)$ *.*

4- Compute $h(m(x))$ *and* $v_2(x) = \theta(x)^{h(m(x))} (\bmod x^2)$*.*

5-Accept the signature only if $v_1(x) = v_2(x)$*.*

Example 86 *To generate public and private keys, entity A selects the prime* $p = 3$ *and the polynomial* $f(x) = 1 + x^5$*. Since* $f(x)$ *is reducible over* $Z_3[x]$*, then* $Z_3[x]/ < f(x) > \cong Z_3[x]/ < x^2 >$*,* $U(\mathbf{Z}_3[x]/ < x^2 >= \{1, 2, 1 + x, 2 + x, 1 + 2x, 2 + 2x\}$ *and* $\phi(x^2) = 3(3 - 1) = 6$*. Entity A chooses the generator* $\alpha(x) = 2 + 2x$ *of* $U(\mathbf{Z}_3[x]/ < x^2 >$ *.and chooses* $a = 5$*. Then, A computes* $y(x) = [\alpha(x)]^a = (2x+2)^5 = 1+x$*. Hence, A's public key is* (3, x^2, $2x + 2$) *and A's private key is* $a = 5$*. To sign the message* $m(x) = 2 \in U(\mathbf{Z}_3[x]/ < x^2 >)$*, assume that* $h(m(x)) = m(x)$ *(for simplicity h is the identity function). Thus* $h(2) = 2$*. Afterwards, A selects a random integer* $k = 5$ *with* $gcd(5 , p^2 - p) = 1$ *and computes* $r(x) = (2x+2)^5$ $(mod\ x^2) \equiv x+2$*. Hence, computing the value of* $\delta(x)$*, A obtains* $\delta(x) \equiv (x+2)^5$ $(mod\ x^2) = 2x+2$ *over* $Z_3[x]$*. Also,* k^{-1} *is obtained from* $5k^{-1} \equiv 1$ $(mod\ (p^2 - p))$*. Hence,* $k^{-1} = 5$*. Then, A computes* $s = 5\{2 - (5).(5)\}$ $(\bmod 6) = 5$ *. Thus, A's signature for* $m(x)$ *is then* $(r(x) = x+2\ , s = 5, \delta(x) = 2x+2)$*. To validate the signature, for sure* $2 \in U(\mathbf{Z}_3[x]/ < x^2 >)$*. Then, B computes the hashed message* $h(2) = 2$ *(for simplicity h is the identity function). Finally, B computes* $v_1(x) = (2x + 2)(x + 2)^5 (\bmod x^2) = 1 + 2x$ *and* $v_2(x) = (2x + 2)^2$ $(\bmod x^2) \equiv 1 + 2x$*. Since* $v_1(x) = v_2(x)$*, the signature is accepted.*

Computer Program: (* We create a computer program with mathematica to illustrate the modified ElGamal signature over quotient rings of polynomials over finite field *)

```
(*Key generation*)
FindGeneratorPoly[p_] := (ϕ = p(p - 1); j = FactorInteger[ϕ]; z = {};
Do[z = Append[z, j[[ i]][[1]]], {i, 1, Length[j]}];
flag = 0;
While[flag == 0,
θ = Random[Integer, {1, p - 1}] + x* Random[Integer, {1, p - 1}];
flag = 1;
Do[If[PolynomialPowerMod[θ,ϕ/z[[i]], { x^2, p}] == 1, flag = 0; Break[]], {i, 1,
Length[z]}]];
Print["θ=", θ];
a = Random[Integer, {2, ϕ- 1}];
θa = PolynomialPowerMod[θ, a, { x^2, p}]);
(* Signature generation*)
SignaturePoly[p_] := (k = 3; While[GCD[k, ϕ] ≠ 1, k = Random[Integer, {2,
ϕ- 1}]];
hm = m;r= PolynomialPowerMod[θ, k, {x^2, p}];
k1 = PowerMod[k, -1,ϕ];
s = Mod[k1*(hm - k*a), ϕ];
δ = PolynomialPowerMod[r, a, {x^2, p}]; Print["signature is ", r, "", s, "", δ])
(* Signature Verification*)
VerificationPoly[p_] := (
v1 = PolynomialMod[PolynomialPowerMod[r, s, {x^2, p}]*δ, {x^2, p}];
hm = m;
v2 = PolynomialPowerMod[θ, hm, {x^2, p}];

Print[v1, "", v2];

If[v1 == v2, Print["acc"], Print["not acc"]]);
```

Conclusion 87 *The extended ElGamal signature over quotient rings of polynomials over finite fields is described using the same arithmetics in polynomial rings over finite fields $\mathbf{Z}_p[x]/ < g(x) >$ where p is an odd prime and n is the degree of $g(x)$. This extension is prefered to the previous one because the polynomial $g(x)$ need not to be irreducible over $\mathbf{Z}_p[x]$. In this case, $g(x)$ is a square power of a linear irreducible polynomial. That is $g(x) = h(x)^2$, where $h(x) = ax + b$. Hence, we deduce that $U(\mathbf{Z}_p[x]/ < (ax+b)^2 >)$ is cyclic and $\mathbf{Z}_p[x]/ < (ax+b)^2 > \cong \mathbf{Z}_p[x]/ < x^2 >$. The computational procedures in the new setting were described and the advantages of the new scheme were pointed out. The following are some of these advantages. First, generating a prime p in both the classical and the extended methods requires the same amount of effort. However, if p of the form $4k + 3$, we use the extended ElGamal signature scheme in the domain of Gaussian integers modulo Gaussian primes $\beta = p$. Otherwise, that is to say, if p is of the form $4k + 1$, we use the extended ElGamal signature scheme in the rings of polynomials over finite fields. But, finding irreducible polynomials in finite fields is not an easy problem. Hence, if one cannot find an irreducible polynomial $h(x)$, he could use the extended ElGamal signature scheme over quotient rings of polynomials over finite fields, $\mathbf{Z}_p[x]/ < x^2 >$ instead of using the extended ElGamal signature scheme over polynomial rings over finite fields. However, the cyclic group $\mathbf{Z}_p^*$ has $p - 1$ elements, while the cyclic group $(\mathbf{Z}_p^*[x]/ < x^2 >)$ which is the same as the group $U(\mathbf{Z}_p[x]/ < x^2 >)$ has*

$p^2 - p$ elements. Second, the choice for the power a in the classical method is from 2 to $p-1$. While the choice of this power in the extended method over polynomial is from 2 to $p^2 - p - 1$. Hence, the extended method provides more security since the security of ElGamal scheme depends on the discrete logarithm. Also, the number of elements that can be chosen to represent the message m is more than the square power of that used in the classical case. Third, the computations involved in the extended method in $\mathbf{Z}_p[x]/ < x^2 >$ do not require computational procedures that are different from those used in the extended method $\mathbf{Z}_p[x]/ < f(x) >$. Hence, the efforts for finding a generator for $\mathbf{Z}_p[x]$ and $\mathbf{Z}_p[x]/ < x^2 >$ are the same.

3.2 ElGamal Public-Key Scheme Attack

In order to attack any protocol that uses ElGamal signature scheme we have to solve the discrete logarithm problem. There are many algorithms for solving the discrete logarithm problem, see [22]. The security of ElGamal signature scheme depends on the intractability of the discrete logarithm problem. In the following, we describe some of the algorithms for solving the discrete logarithm problem in the general setting of a finite cyclic group G of order n generated by α. The discrete logarithm of an element β in G to the base α, denoted $log_\alpha \beta$, is the unique integer x, such that $\beta = \alpha^x$, where $0 \leq x \leq n-1$.

The groups considered in this chapter are:

1. The multiplicative group Z_p^*;of the integers modulo a prime p.

2. The multiplicative group G_β^*; of the Gaussian integers modulo a prime β.

3. The group of units Z_n, where n is a composite integer of the form $2p^t$.

4. The multiplicative group $F^*$of a finite field F.

5. The group of units of the quotient ring $Z_p[x]/ < x^2 >$.

Given a generator α of a finite cyclic group G of order n and an element β in G, the discrete logarithm problem is the problem finding the unique integer $x, 0 \leq x \leq n-1$, such that given $\beta = \alpha^x$. The known algorithms for solving the discrete logarithm problem are: exhaustive search, the baby-step giant-step algorithm, Pollard's rho algorithm, Pohlig-Hellman algorithm, and the index-calculus algorithms. The first three algorithms work in arbitrary groups. The baby-step giant-step algorithm is a time-memory trade-off of exhaustive search. Pollard's rho algorithm is a randomized algorithm. It requires a negligible amount of storage and has the same expected running time as the baby-step giant-step algorithm. The Pohlig-Hellman algorithm works in arbitrary groups but is especially efficient if the order of the group has only small prime factors. The index-calculus algorithms are efficient only in certain groups. Detailed descriptions and references of these algorithms can be found in [22].

In the following we describe the exhaustive search and the baby-step giant-step algorithms. These algorithms will be used in section 4 to evaluate the various versions of ElGamal digital signatures.

Exhaustive Search An obvious algorithm for the discrete logarithm problem is to successively compute α^0,α^1, α^2, ... until β is obtained. If n is the order of α, the exhaustive search takes $O(n)$ multiplications, and is therefore inefficient if n is large (i.e., in cases of cryptographic interest). However, it can be used to compare the various ElGamal based digital signatures having small parameters. The algorithm is as follows:

Algorithm 88 ***Exhaustive Search***

INPUT: a generator α of a cyclic group G of order n, and an element $\beta \in G$.

OUTPUT: the discrete logarithm $x = \log_\alpha \beta$.

1. *Set $k = 0$.*

2. *Set $\beta = \alpha^k$. If $\beta = x^a$, then return k.*

3. *Set $k = k + 1$, then return with new k; $0 \leq k \leq n - 1$, until $\beta = x^a$ is reached.*

Example 89 *Let $p = 11$. $\theta = 2$ is a generator of Z_{11} of order $p - 1 = 10$.Consider $\beta = 10 = 2^5$. Applying the* ***Exhaustive Search*** *algorithm,*$\log_2 10 (\bmod 11)$ *is computed as follows.*

1. *Set $k = 0$.*

2. *Compute $\beta = \theta^k = 2^0 = 1 \neq 10$*

3. *Set $k = 1$ and compute $\beta = \theta^k = 2^1 = 2 \neq 10$.Keep on computing θ^k for $k = 0, 1, 2....$ until 10 is reached.This yields the following table:*

k	0	1	2	3	4	5
$\theta^k = 2^k$	1	2	4	8	5	10

4. *When $k = 5, \theta^k = 10$.Therefore, $\beta = \theta^5$, i.e., the discrete logarithm $\log_2 10 = 5$.*

Example 90 *Let $p = 3$. $\theta = 2 + 2i$ is a generator of G_3^* of order $p^2 - 1 =$ 8. Consider $\beta = 2 = (2{+}2i)^4$. Applying the **Exhaustive Search** algorithm,*$\log_{2+2i} 2($mod *is computed as follows.*

1. Set $k = 0$.
2. Compute $\beta = \theta^k = (2 + 2i)^0 = 1 \neq 2$.
3. Set $k = 1$ and compute $\beta = \theta^k = (2 + 2i)^1 = 2 + 2i \neq 2$.Keep on computing θ^k for $k = 0, 1, 2....$ until 2 is reached.This yields the following table:

k	0	1	2	3	4
$\theta^k = (2 + 2i)^k$	1	$2 + 2i$	$2i$	$2 + i$	2

Example 91 *1. 4. When $k = 4, \theta^k = 2$.Therefore, $\beta = \theta^4$, i.e., the discrete logarithm* $\log_{2+2i} 2 = 4$.

Baby-step Giant-step Algorithm Let n be the order of a generator α and let $m = \lceil \sqrt{n} \rceil$, the smallest integer greater than or equal to $\sqrt{n}$. The baby-step giant-step algorithm is based on the following observation. Suppose that $\beta = \alpha^x$. Writing $x = km + j$, where $0 \leq k,\ j \leq m$, we have that $\alpha^x = \alpha^{km}\alpha^j$ and hence $\beta(\alpha^{-m})^k = \alpha^j$. This suggests the following algorithm for determining $x = \log_\alpha \beta$.

Algorithm 92 *$Baby - step\ giant - step\ algorithm\ for\ finding\ discrete\ logarithms$*

INPUT: α, a generator of a cyclic group G of order n, and an element $\beta \in G$.

OUTPUT: the discrete logarithm $x = \log_\alpha \beta$, where $\beta = \alpha^x$.

1. Set $m = \lceil \sqrt{n} \rceil$.

2. For $0 \leq j \leq m$, find α^j and construct a table with entries $(j,\ \alpha^j)$.

3. Sort this table by second component.

4. Find α^{-m} and set $\gamma = \beta$.

5. For k from 0 to $m-1$ do the following:

5.1 If $\gamma = \alpha^j$ then return $(x = km + j)$. Else, set $\gamma = \gamma.\alpha^{-m}$.

This algorithm requires storage for $O(\sqrt{n})$ group elements. To construct the table, $O(\sqrt{n})$ multiplications and $O(\sqrt{n}\lg n)$ comparisons to sort are required. Step 5 takes $O(\sqrt{n})$ multiplications and $O(\sqrt{n})$ table look-ups. Assuming that a group multiplication requires more time than $\log n$ comparisons, we have that the running time of Baby-step giant-step algorithm is $O(\sqrt{n})$ group multiplications.

Note that the performance can be improved by restricted exponents a special form. Usually, exponents having small Hamming weight are used. The Hamming weight of an integer is the number of ones in its binary representation. The following is an example of a modified Baby-step giant-step attack for Gaussian integers.

Example 93 $(Baby-step\ giant-step\ algorithm\ for\ logarithms\ in\ Z_{13})$

Let $p = 13$. The element $\alpha = 2$ is a generator of Z_{13}, of order $n = p - 1 = 12$. Consider $\beta = 9 = 2^8$. Applying the Baby-step giant-step algorithm, $\log_2 9 \pmod{13}$ is computed as follows.

1. Set $m = \lceil \sqrt{12} \rceil = 4$.

2. Construct a table whose entries are $(j,\ \alpha^j (\operatorname{mod} p))$ for $0 \leq j < 4$:

3.

j	0	1	2	3
2^j	1	2	4	8

4. *Compute* $\alpha^{-1} = 2^{-1} (\mathrm{mod}\, 13) = 7$ *and then compute* $\alpha^{-m} = 7^4 (\mathrm{mod}\, 13) = 9$.

5. *Next,* $\gamma = \beta\alpha^{-mk} (\mathrm{mod}\, 13)$ *for* $k = 0, 1, 2, \ldots$ *is computed until a value in the second row of the above table is obtained. This yields the following table:*

k	0	1	2
γ	9	3	1

When $k = 2$, $\gamma = 1$ *is the first value of* γ *appearing in the second row of the table in step 4 corresponding to* $j = 0$. *Finally,* $x = km + j = 2 * 4 + 0 = 8$. *Therefore,* $\beta = \alpha^8$, *i.e., the discrete logarithm* $\log_2 9 = 8$.

Example 94 $(Baby - step\ giant - step\ algorithm\ for\ logarithms\ in\ G_7^*)$

Let $p = 7$. *The element* $\alpha = 2 + 6i$ *is a generator of* G_7^*, *of order* $n = p^2 - 1 = 48$. *Consider* $\beta = 1 + 6i = (2 + 6i)^{22}$. *Applying the Baby-step giant-step algorithm,* $\log_{2+6i} 1 + 6i\ (\mathrm{mod}\, 7)$ *is computed as follows.*

1. *Set* $m = \lceil \sqrt{48} \rceil = 7$.

2. *Construct a table whose entries are* $(j,\ \alpha^j (\mathrm{mod}\, p))$ *for* $0 \leq j < 7$*:*

3.

j	0	1	2	3	4	5	6	7
$(2+6i)^j$	1	$2+6i$	$3+3i$	$2+3i$	$4i$	$4+i$	$2+5i$	$2+i$

4. *This table can be sorted by second component using the linear order* $\preceq$ *defined by* $a + bi \preceq c + di$ *iff* $(a < c)$ *or* $(a = c$ *and* $b < d)$. *The table becomes:*

j	4	0	7	3	6	1	2	5
$(2+6i)^j$	$4i$	1	$2+i$	$2+3i$	$2+5i$	$2+6i$	$3+3i$	$4+i$

5. *Compute* $\alpha^{-1} = (2+6i)^{-1}(\bmod 7) = 6+3i$ *and then compute* $\alpha^{-m} = (6+3i)^7(\bmod 7) = 6+4i$.

6. *Next,* $\gamma = \beta\alpha^{-mk}(\bmod 11)$ *for* $k = 0,1,2,...$ *is computed until a value in the second row of the above table is obtained. This yields the following table:*

k	0	1	2	3
γ	$1+6i$	$3+5i$	5	$2+6i$

When $k=3$, $\gamma = 2+6i$ *is the first value of* γ *appearing in the second row of the table in step 4 corresponding to* $j=1$. *Finally,* $x = km+j = 3\times 7+1 = 22$. *Therefore,* $\beta = \alpha^{22}$, *i.e., the discrete logarithm* $\log_{2+6i} 1+6i = 22$.

Computer Program: (* We create a computer program with mathematica to illustrate the *Baby − step giant − step algorithm* *)

```
Timing[n = p\2 - 1;
m = Ceiling[Sqrt[n]];
t1 = Array[ t11, {2, m}];
MatrixForm[t1];
Do[t1[[1, j]] = j - 1; t1[[2, j]] = PowerMod[α, j - 1, p];
(*Print[tt1[[1, j]], "", t1[[2, j]]]*) , {j, 1, m}];
MatrixForm[t1]; ai = PowerMod[α, -1, p];
am = PowerMod[ai, m, p];
h = 1; i = 0; e = 1; l = 0;
While[e≠0 && l ≤ m,
ami = PowerMod[am, i, p];
γ = Mod[β*ami, p]; (*Print[i, "", γ]*) ;
Do[If[γ == t1[[2, k]],
e = 0; j= t1[[1, k]]; (*Print["j = ", j]; Print["γ = ", γ]*) ;
h = 2; Break[]],
{k, 1, m}]; i = i + 1;] Print["i = ", i - 1]; a = (i - 1)*m + j])
```

3.3 Testing and Evaluation

In this section, we compare and evaluate the different classical and modified digital signatures schemes by showing the implementation of the algorithms with their running results. Also, we test the security of the algorithms by implementing different attack algorithms to forge the signature. All this is done using Mathematica 5.0 as a programming language and a centrino acer computer with 1.5 GHZ CPU and 256 MB DDRAM.

ElGamal based Algorithms Using Mathematica 5.0 functions and an additional abstract algebra library, we have written programs for the following algorithms:

1. Classical ElGamal.

2. Classical ElGamal with n of the form $2p^t$.

3. ElGamal with Gaussian numbers.

4. ElGamal with irreducible polynomials.

5. ElGamal with reducible polynomials.

The various procedures were compared as follows.

a-A total of 25 runs of the various algorithms were conducted. In each run, a 20-digit random prime integer p of the form $4k+3$ was generated.

b-The same prime p was used for all algorithms.

c-For each method a public key was generated by finding a generator θ, a random integer a, and computing θ^a.

d-Using the public key (θ, θ^a), the same message $m = 12345678$ was signed by all algorithms to obtain the signature (r, s, δ).

e-The verification algorithms were then used to verify the signature.

f-All algorithms used the built-in Mathematica functions for modular arithmatic and for finding powers modulo an integer, Gaussian integer or a polynomial over Z_p.

g-The running times of the algorithm (Key generation, signature, verification) for each method were recorded.

Note that the cyclic groups used, and their corresponding orders, are:

1. Classical ElGamal: Z_p^* of order p.

2. Classical ElGamal with n of the form $2p^2$: Z_n^* of order $p^2 - p$.

3. ElGamal with Gaussian integers: G_p^* of order $p^2 - 1$.

4. ElGamal with reducible polynomials: $U(Z_p[x]/ < x^2 >)$ of order $p^2 - p$.

5. ElGamal with irreducible polynomials: $U(Z_p[x]/ < x^2 + ax + b >)$ of order $p^2 - 1$.

Except for the classical ElGamal in the setting of the cyclic group Z_p^*, all cyclic groups used have comparable sizes. Hence, we expect the algorithms in the first case to be much faster. A different prime p having 40 digits could have been used for that case; but this would have been equivalent to case 2.

After running the programs, it was clear that these programs have applied the ElGamal signature scheme in the correct way. All the programs have generated a public and private key with different mathematical concepts. Then a message is signed using the signature scheme and is sent to a verification procedure which verify the signature. Figure 3.1 shows the average execution time of 25 runs of the various algorithms.

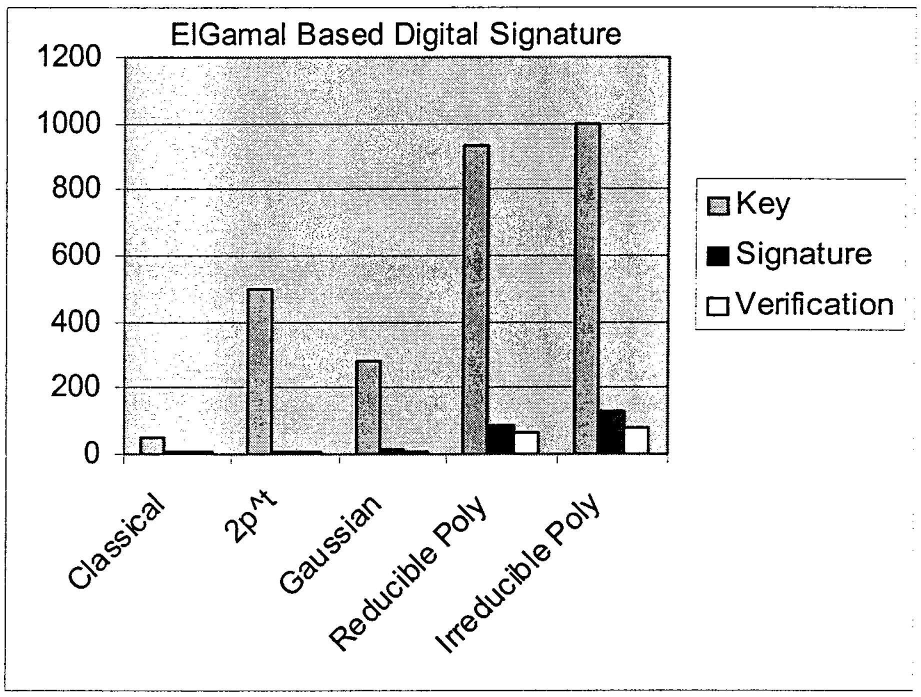

Figure 3.1: Average execution time for ElGamal Digital Signature

Comparing these algorithms, we observe the following:

a- All programs are reliable; they can sign and verify any signature.

b- The complexity for each of the algorithms is $O(n^2)$.

c- The reducible polynomial signature scheme is reliable but took more time to generate a key and to sign a message. This does not mean that it is inefficient because it is more secure than the other algorithms. This will be shown later in the attack section. Also, these algorithms can be made more efficient by modifying the built-in Mathematica functions to take advantage of the arithmetic modulo x^2.

d- The irreducible polynomial program in the setting $Z_p[x]/ < x^2 + ax + b >$ worked well but requires more time. The signature and verification execution time for the irreducible polynomial scheme is slightly more than that of the reducible case. This is due to the additional computations needed for the arithmetic modulo the polynomial $x^2 + ax + b$.

e-The key generation time for the irreducible polynomial scheme is considerably more than that of the other methods. This is due to the fact that an irreducible polynomial must be generated before a generator θ is found. On average, it took about 0.4 sec to generate a quadratic irreducible polynomial for a 20-digit prime number.

Note that in Figure 3.1, the time of generating the irreducible polynomial is not included. Figure 3.2, shows the average execution time if the time of finding an irreducible polynomial is included in the key generating algorithm.

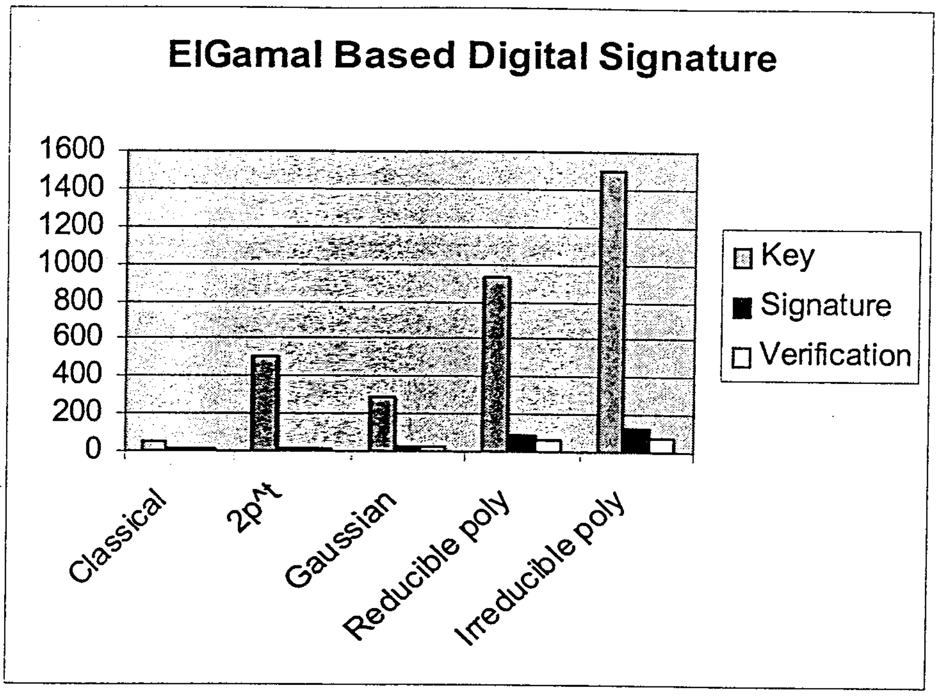

Figure 3.2: Average execution time for ElGamal Digital Signature

From these results, we can conclude that:

a- The time for generating the key depends on finding the generator θ and not the prime p. The time for generating a prime number is negligible. The average time needed for generating a 100-digit (recommended size) random prime is approximately 0.1 sec.

b- It took more time to find the key in the case of polynomials. This will not be a problem if common system-wide parameters are used. In such a case, all entities may elect to use the same cyclic group G and generator ϕ. Also, once a generator ϕ for a given prime p is found, all other generators can be easily obtained.

c- The time needed to sign and to verify the signature for the classical, modified $2p^t$ and Gaussian is better than the time needed for the polynomials. However, the time is very reasonable even for larger primes.

d- Overall, the Gaussian integers methods performed very well. The algorithms can be made more efficient by modifying Mathematica built-in functions to take advantage of the arithmetic modulo the Gaussian prime p of the form $4k + 3$.

e- The key generation time in the case of Gaussian integers is less than that of the modified $2p^t$ method. This is due to the fact that the number of generators in $Z^*_{2p^2}$, which is $\phi(p(p-1))$, is almost always more than the number of generators in G^*_p, which is $\phi(p^2 - 1)$. In fact, among the first $200,000$ primes, there are only 7 primes p of the form $4k + 3$ for which $\phi(p^2 - 1) > \phi(p(p - 1))$.

f-The reducible polynomial method is little slower but provide more security. The irreducible polynomial method is not recommended since it is as secure as the reducible case but requires more time especially in finding the key.

Attack Algorithm In order to attack any protocol that uses ElGamal signature scheme we have to solve the discrete logarithm problem. We enhanced the *Exhaustive search* and *Baby − step giant − step* algorithms to work with the modified algorithms.

To test the security of the algorithms, we implemented and applied the attack schemes to the classical and modified signature algorithms. The ElGamal algorithm using irreducible polynomials was not tested since the attack time would be equivalent to that of the reducible polynomial case. For the exhaustive search algorithm, a random 3-digit prime p of the form $4k+3$ was generated and a public key was obtained for each the four methods using the same prime. The average time it took to break the key for a total of 25 runs are shown in figure 3.3.

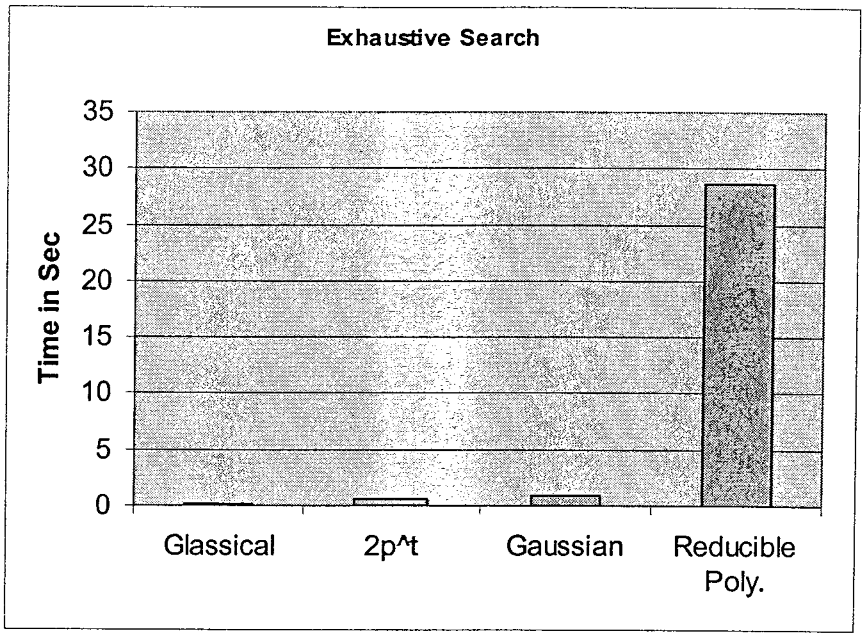

Figure 3.3: Attack Time: Exhaustive Search Algorithm

Attacking the ElGamal schemes using Baby-step giant-step algorithm, a random 4-digit prime p of the form $4k+3$ was generated and a public key was obtained for each the four methods using the same prime. The average time it took to break the key for a total of 25 runs are shown in figure 4.

After running these attack algorithms, we observed the following:

a- All the attack programs are reliable so that they can forge any message by finding the private key.

b- The $2p^t$ algorithm is stronger than the classical algorithm because we have an unknown power t.

c- The Gaussian algorithm is stronger than the classical algorithm. The attack algorithm for Gaussian integers required more time than that of the $2p^2$ algorithm.

d- The most difficult algorithm to attack is in the polynomial domain. This is due to the fact that mathematically it is complex and needs considerable computing time to perform arithmetic modulo a given polynomial.

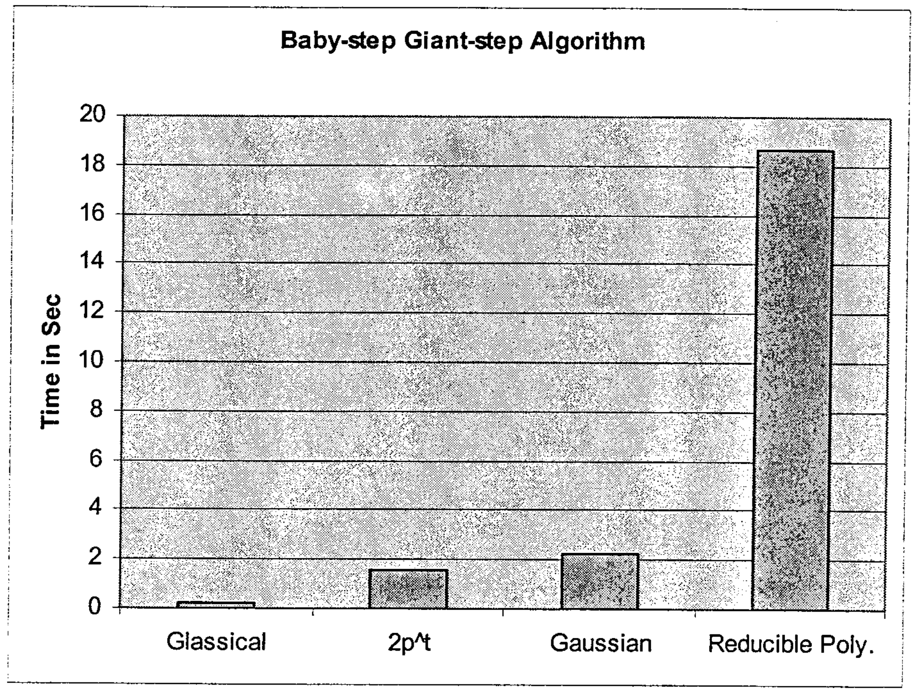

Figure 3.4: Attack Time: Baby-Step Giant-Step Algorithm

3.4 Conclusion

In this work, we presented the classic ElGamal signature scheme and four modifications to it, namely, the ElGamal signature scheme in Z_n, in the domain of Gaussian integers, $Z[i]$, over finite fields, and over quotient rings of polynomials over finite fields. We implemented these algorithms and tested their efficiency, reliability, and security. The results obtained showed that all the algorithms applied the ElGamal signature scheme correctly and generated public and private key using different mathematical concepts. Messages were then signed using the signature scheme and were sent to a verification procedure which verify the signature.

We also built attack scenarios directly aimed at solving the discrete logarithm problem that these algorithms utilize. We modified the Baby-step Giant-step algorithm to handle the modified algorithms. We observed that the classical ElGamal scheme is the weakest to attack and one of the modified methods should be used. The ElGamal scheme in the multiplicative group Z_n^*, where $n = 2p^2$, is the easiest to apply and the weakest among the modified method. The ElGamal scheme in the domain of Gaussian is superior to that of Z_n^* since it requires less time to generate a key, about the same time to sign and verify a signature, and is more secure. The ElGamal scheme in the setting of a finite field has a disadvantage in finding an irreducible polynomial. The reducible polynomial scheme was the most challenging to attack due to the mathematical complexity of arithmetic modulo a polynomial.

CHAPTER 4

A COMPARATIVE STUDY OF RSA BASED DIGITAL SIGNATURE ALGORITHMS

In this chapter, we compare and evaluate the classical and modified RSA algorithms. We investigate the issues of complexity, efficiency and reliability by running the programs with different sets of data. Moreover, these different algorithms will be compared. In addition, implementation of an attack algorithm will be presented.This is done by applying certain mathematical concepts to find the private key . After finding the key, it will be easy to sign the message. A study will be done using the results of running the attack algorithm to compare the security of the classical and modified signature scheme algorithms.

4.1 Classical and Modified RSA Signature Scheme

The classical and modified RSA signature schemes are described in this section. Algorithms and examples are given. These algorithms will be implemented to evaluate and compare the various methods in section 5.

Classical RSA Signature Scheme In RSA signature scheme, entity A generates the public-key by first generating two large random odd primes p and q, each roughly of the same size, and computing the modulus $n = pq$ and Euler phi-function $\phi(n) = (p-1)(q-1)$, see [22]. Entity A then selects the exponent e to be any random integer in the interval $(1, \phi(n))$ such that $\gcd(e, \phi(n)) = 1$. Using the extended Euclidean algorithm for integers, entity A finds the exponent d which is the unique integer $(1, \phi(n))$

relatively prime to $\phi(n)$ such that $ed \equiv 1(\bmod\, \phi(n))$. Hence, the public-key is the pair $(n,\ e)$, and $A's$ private-key is the triplet $(p,\ q,\ d)$.

The signature is generated by A as follows. First, entity A computes the redundancy function of the message m which is $\widetilde{m}\ = R(m)$ such that $R(m) \in \mathbf{Z}_n$ and also computes

$$s \equiv \widetilde{m}^d (\bmod\, n)$$

Finally, A sends the signature s to entity B.

The signature is validated by B as follows. B obtains A's authentic public key (n, e), computes

$$\widetilde{m} \equiv s^e (\bmod\, n)$$

and rejects the signature if $\widetilde{m} \notin M_R$ (image of R). Finally, B recovers m by computing $R^{-1}(\widetilde{m})$.

Algorithm 95 *RSA Signature Scheme:*

1. Find two large primes p and q and compute their product $n = pq$.

2. Find an integer d that is relatively prime to $\phi(n) = (p-1)(q-1)$.

3. Compute e from $ed \equiv 1(\bmod\ \phi(n))$.

4. Broadcast the public key (n, e).

5. Compute the redundancy function of the message m which is $\widetilde{m}\ = R(m)$ such that $R(m) \in \mathbf{Z}_n$

6. Sign the message m using the private-key by applying the rule $s \equiv \widetilde{m}^d (\bmod n)$.

7. The receiver validate the signature using the rule $\widetilde{m} \equiv s^e (\bmod n)$.

Example 96 *In order to generate the public-key, entity A selects the primes* $p = 852225047$ *and* $q = 603309029$ *and then computes the modulus* $n = pq = 514155065595049363$ *and the Euler phi-function* $\phi(n) = (p-1)(q-1) = 514155064139515288$. *Next, A selects the exponent* $e = 2318142620792164 29$ *and uses the extended Euclidean algorithm for integers to find the exponent* $d = 387883402970610381$ *so that* $ed \equiv 1(\bmod \phi(n))$. *Now, the public-key is the pair*

$$(n = 514155065595049363, e = 2318142620792164 29)$$

and A'*s private-key is the triplet*

$$(p = 852225047, q = 603309029, d = 387883402970610381).$$

To sign the message $m = 1101100100111$ *,for simplicity, take* $R(m) = m$ *so that* R *is the identity function. Then,* $\widetilde{m} = R(1101100100111) = 1101100100111$. *A computes* $s = \widetilde{m}^d (mod\ n) = 1101100100111^{387883402970610381} (mod\ 514155065595049363) = 502534570854711493$ *and sends the signature* 502534570854711493 *to B. B obtains A's authentic public key* $(514155065595049363, 2318142620792164 29)$, *computes* $\widetilde{m} = s^e\ (mod\, n) = 502534570854711493^{2318142620792164 29} (mod\, 514155065595049363) = 110110010011$ *and computes* $m = R^{-1}(\widetilde{m}) = \widetilde{m} = 1101100100111$.

Computer Program: (* We create a computer program with mathematica to illustrate the classical RSA signature*)

```
(*Key generation*)
Generatek[l_] := (p = 1; q = 1;
While[Mod[p, 4] == 1 || Mod[q, 4] == 1,
p = lDigitPrime[l]; q = lDigitPrime[l]];
n = p*q; φ= (p - 1)(q - 1);
e = φ; While[GCD[e, φ] ≠ 1,
e = Random[ Integer, {1, φ - 1}]];
d = PowerMod[e, -1, φ]; Print["the public key is ( n=", n, ", e=", e, ")"])
(*Signature generation*)
GenerateS[l_] := (m1 = m; s = PowerMod[m1, d, n];
Print[" the signature is ( s=", s, ")"])
(*Signature verification*)
VeriS[l_] := (u = {};
Do[u = Append[u, i], {i, n - 1}];
ms = PowerMod[s, e, n];
If[MemberQ[u, ms], Print["acc"]]; Print["ms=", ms])
```

4.2 RSA Signature Scheme in the Domain of Gaussian Integers, $\mathbb{Z}[i]$

In this section, we proceed to the extension of RSA signature scheme from the domain of natural integers to the domain of Gaussian integers using the arithmetic in $\mathbf{Z}[i]$. In this process, we have used the mathematical material concerning Gaussian integers presented in chapter 2. Algorithms, examples, and computer programs illustrating this extended method are given. Moreover, as the classical RSA signature, the

RSA signature algorithm requires a redundancy function.

Choice of Gaussian Primes The Gaussian primes choosen in the Gaussian signatures could not be choosen randomely. There are two choices for the Gaussian prime β to select. $\beta = \pi$ where $\pi\overline{\pi}$ is a prime integer of the form $4k+1$ and $\beta = p$ where p is a prime integer of the form $4k+3$. Therefore, we have three cases for the composite η. where $\eta = \beta_1\beta_2$

1st case : $\beta_1 = \pi_1$,and $\beta_2 = \pi_2$ where $q_1 = \pi_1\overline{\pi}_1 = 4k_1 + 1$and $q_1 = \pi_2\overline{\pi}_2 = 4k_2 + 1$ are prime integers. Then, due to El-Kassar [15], the complete residue system modulo $\eta = \beta_1\beta_2$ is of the form $G_\eta = \{a + b\beta_1 : a\epsilon\beta_1 : a \in G_{\beta_1}\text{and } b \in G_{\beta_2}\}$ where $G_{\beta_1} = \{a : 1 \leq a \leq q_1 - 1\}$ and $G_{\beta_2} = \{b : 1 \leq b \leq q_2 - 1\}$. But the reduced residue system modulo η has the order q_1q_2 -1 which will be similar to working in the classical case (case of integers). So, it will be neglected.

2nd case : $\beta_1 = \pi_1$and $\beta_2 = p$, where $q_1 = \pi_1\pi_1 = 4k_1 + 1$ and $p = 4k_2 + 3$ are prime integers. Then, the factorization problem of the composite Gaussian integer $\mu = \beta_1\beta_2$ which has the form $a + bi$ could be easily solved by finding $gcd(a, b)$ which will be equal to p. Therefore, this case will be also neglected.

3rd Case : $\beta_1 = p_1$and $\beta_2 = p_2$ where $p_1 = 4k_1 + 3$ and $p_2 = 4k_2 + 3$ are prime integers. Then, due to El-Kassar [15], the complete residue system modulo $\eta = \beta_1\beta_2$ is of the form $G_\eta = \{r+s\beta_1 : r \,\epsilon\, G_{\beta_1}\text{and } S \in G_{\beta_2}\}$ where $G_{\beta_1} = \{a+bi : 1 \leq a, b \leq p_1 - 1\}$ and $G_{\beta_2} = \{c + di : 1 \leq c, d \leq p_2 - 1\}$.

The following theorem determine the the number of elements in G_η where η is a composite integer.

Theorem 97 *Let $\eta = \beta_1\beta_2$ be a composite Gaussian integer where $\beta_1 = 4k_1 + 3$ and $\beta_2 = 4k_2 + 3$. Then, the complete residue system modulo η is the set of Gaussian integers $G_\eta = \{\lambda + \gamma i : 0 \leq \lambda \leq p_1p_2 - 1, 0 \leq \gamma \leq p_1p_2 - 1\}$.*

For the proof of the theorem, see [15].

Thus, the number of elements in G_η is $q(\eta) = (p_1p_2)^2$ and the number of the invertible elements in the reduced residue system modulo η, G_η^* is $\phi(\eta) = (p_1^2p_2^2 - 1)(p_1^2p_2^2 - 1)$.

<u>Description of the Gaussian signature scheme</u> In RSA signature scheme, entity A generates the public-key by first generating two large random Gaussian primes β, γ and computes $\eta = \beta\gamma$. Next, entity A computes $\phi(\eta) = \phi(\beta)\phi(\gamma) = (\beta^2 - 1)(\gamma^2 - 1)$, where $\phi(\eta)$ is Euler phi-function in $\mathbb{Z}[i]$, see [2]. It selects a random integer e such that $1 < e < \phi(\eta)$ and e is relatively prime to $\phi(\eta)$. Then, entity A finds the unique integer d such that $1 < d < \phi(\eta)$ and d is relatively prime to $\phi(\eta)$ such that $ed \equiv 1 (\mathrm{mod}\, \phi(\eta))$. A's public-key is

$$(\eta,\ e)$$

and A's private-key is

$$(\beta,\ \gamma,\ d).$$

Represent the message as a number μ chosen from the complete residue system modulo η, $G_\eta = \{a + bi | 0 \leq a \leq \beta\gamma - 1, 0 \leq b \leq \beta\gamma - 1\}$.After computing the redundancy function of the message μ which is $\widetilde{\mu} = R(\mu)$,A computes the signature $s = \widetilde{\mu}^d (mod\ \eta)$ and sends it to B. To verify the signature sent by A, B gets A's public key (η, e), computes the message representative $\widetilde{\mu}$ as $\widetilde{\mu} = s^e (mod\ \eta)$ and finally applies verification process to $\widetilde{\mu}$ to recover μ.

We note that the message space is enlarged so that its order is the square of that of the classical case. Also, the range for the public exponent e is enlarged by more than the square of that of the classical case.

In the following we provide three algorithms describing the RSA signature scheme over the domain of Gaussian integers. First, entity A generates the public and private keys by doing the following.

Algorithm 98 *(Key generation for the RSA Gaussian signature)*

1. *Generating two distinct large random Gaussian primes α and β, each roughly the same size.*
2. *Computing $\eta = \alpha\beta$ and $\phi(\eta) = (\alpha^2 - 1)(\beta^2 - 1)$.*
3. *Selecting a random integer e, $1 < e < \phi(\eta)$ such that $gcd(e, \phi(\eta)) = 1$.*
4. *Computing the multiplicative inverse d of e such that $ed \equiv 1(\mathrm{mod}\,\phi)$ using the extended Euclidean algorithm over the domain of Gaussian integers.*
5. *Publishing the pair (η, e) as the public key, and keeping d as the private key.*

To generate a signature of a message,.entity A should do the following.

Algorithm 99 *(Signature generation of RSA Gaussian signature)*

1. *Represent the message as μ chosen from the complete residue system modulo η, G_η.*

2. *Compute* $\widetilde{\mu} = R(\mu)$ *where* $\widetilde{\mu} \in G_{\eta}$.

3. *Compute* $s = \widetilde{\mu}^{d} \ (mod \eta)$.

4. *Output* s *as the signature to* B.

To verify the signature of the message μ, entity B should do the following.

Algorithm 100 *(Signature verification of RSA Gaussian signature)*

1. *Obtain* A*'s authentic public key* (η, e) .

2. *Recover* $\widetilde{\mu} \equiv s^{e}(mod\ \eta)$.

3. *Verify that* $\widetilde{\mu} \in M_R$ *, otherwise reject the signature.*

4. *Recover* $\mu = R^{-1}(\ \widetilde{\mu})$.

Example 101 *(RSA Gaussian Signature Scheme with Small Parameters)*

Public-Key Generation: Let $\beta = 91939$ *and* $\gamma = 69383$ *be two Gaussian primes of the form* $4k+3$. *Compute the product* $\eta = \beta\gamma = 6379003637$ *and* $\phi(\eta) = (91939^2 - 1)(69383^2 - 1) = 40691687387592447360$. *Entity A chooses* $e = 25600002082007742863$ *such that* $\gcd(e, \phi(\eta)) = 1$ *and* $1 < e < \phi(\eta)$. *Using the extended Euclidean algorithm for integers, A finds* $d = 33899823343652452847$ *such that* $ed \equiv 1(\text{mod}\,\phi(\eta))$. *Hence,* A'*s public-key is the pair*

$$(\eta = 6379003637, e = 25600002082007742863)$$

and A'*s private-key is the triplet*

$$(\beta = 91939, \gamma = 69383, d = 33899823343652452847).$$

Signature Generation: To sign the message $\mu = 320177 + 147i$, *for simplicity, take* $R(\mu) = \mu$ *so that* R *is the identity function so* $\widetilde{\mu} = R(320177{+}147i) = 320177{+}147i$ *. Afterwards, A computes*

$$\begin{aligned} s &= \widetilde{\mu}^d = (320177 + 147i)^{33899823343652452847} \\ &\equiv 3059266386 + 5412724259i (\bmod 6379003637) \end{aligned}$$

Finally, A sends the signature to B.

Signature Verification :To validate the signature, B obtains first A's authentic public key $(\eta = 6379003637, e = 25600002082007742863)$. *Then, B computes*

$$\begin{aligned} \widetilde{\mu} \quad &\equiv \quad s^e \; (\bmod \eta) \quad = \quad 3059266386 + 5412724259i^{25600002082007742863} \quad (mod 6379003637) \\ &= \quad 320177 + 147i \end{aligned}$$

Finally, B computes $\mu = R^{-1}(\widetilde{\mu}) = 320177 + 147i$ *.*

Computer Program: (* We create a computer program with mathematica to illustrate the modified RSA signature in $Z[i]$ *)

```
(*Key generation*)
GeneratekGaussian[l_] := (α = 1; β = 1;
While[Mod[α, 4] == 1 || Mod[ β, 4] == 1,
α = lDigitPrime[ l]; β = lDigitPrime[l]];
η = α*β; φ = (α^2 - 1) ( β^2 - 1);
```

```
e = $\phi$; While[GCD[e,$\phi$] $\neq$ 1,

e = Random[Integer, {1,$\phi$ - 1}]];

d = PowerMod[e, -1, $\phi$]; Print["the

public key is ( $\eta$=", $\eta$, ", e=", e,])

(*Signature generation*)

GenerateSGaussian[l_] := (m1 = m;

s = PowerMod[ m1, d, $\eta$]; Print["the signature is (

s=", s, ")"])

(*Signature verification*)

VeriSGaussian[l_] := (ms = PowerMod[s, e, $\eta$]; Print["ms=", ms])
```

Conclusion 102 *Using the well known and arithmetics in the domain of Gaussian integers* $\mathbf{Z}[i]$*, the RSA signature scheme in the domain of natural integers was extended to* $\mathbf{Z}[i]$*. The computational properties in the new setting were described and the advantages of the new scheme were pointed out. The following are some of the advantages. First, generating the odd primes p and q of RSA signature in the domain of natural integers and in the domain of Gaussian integers requires the same amount of effort. However, the complete residue system* $\mathbf{Z}_n$ *has pq elements, while the complete residue system* G_η *has* $\delta(pq)$ *elements. If p and q are of the form* $4k+3$, $\delta(pq) = p^2q^2$*. If p is of the form* $4k+3$ *and q is of the form* $4k+1$, $\delta(pq) = p^2q$*. If p and q are of the form* $4k+1$*, which is similar to the classical method,* $\delta(pq) = pq$*. Hence, one of the two primes p and q should be of the form* $4k+3$*. Therefore, we deduce that the extended RSA over the domain* $\mathbf{Z}[i]$ *provides an extension to the range of chosen messages, which make trials* *more complicated. Second, note that the Euler phi function in the domain of natural integers is $\phi(n) = (p-1)(q-1)$, while the extended Euler phi function in the domain of Gaussian integers is $\phi(p\pi) = (p^2-1)(q-1)$ or $\phi(p_1p_2) = (p_1^2-1)(p_2^2-1)$ where π is a Gaussian integer that divides a prime integer $q = 4k+1$ such that $\pi\overline{\pi} = q$, and the prime integer $p = 4k+3$ are the Gaussian primes. Hence RSA signature scheme in the domain of Gaussian integers is much better than that in the domain of natural integers if we choose at least one of the Gaussian primes p and q of the form $p = 4k+3$. In this case, the value of $\phi(\eta)$ is larger than that in the domain of natural integers by a multiple of $(p+1)(q+1)$ times. Thus, the trials for finding the private key d will be too complicated. Fourth, the value of the public key e could be large enough to make solving the RSA problem more complicated. Finally, we note that the computations involved in the modified method do not require computational procedures that are different from those used in the classical method.*

4.3 RSA Signature Scheme over Quotient Rings of Polynomials over Finite Fields

Let p be a prime number and let $h(x)$ and $g(x)$ be two distinct irreducible polynomials in $\mathbb{Z}_p[x]$, the domain of polynomials over the finite field $\mathbb{Z}_p$, where $h(x)$ is of degree s and $g(x)$ is of degree r. Let $f(x) = h(x)g(x)$. The polynomials $h(x)$ and $g(x)$ should be selected so that factoring $f(x) = h(x)g(x)$ is computationally infeasible. The quotient ring $\mathbb{Z}_p[x]/\langle f(x)\rangle$ is finite of order p^n, where $n = r+s$ is the degree of $f(x)$. It is well known that the quotient ring $\mathbb{Z}_p[x]/\langle f(x)\rangle$ is the direct sum of $\mathbb{Z}_p[x]/\langle g(x)\rangle$ and $\mathbb{Z}_p[x]/\langle h(x)\rangle$, that is

$$\mathbb{Z}_p[x]/\langle f(x)\rangle \cong \frac{\mathbb{Z}_p[x]}{\langle g(x)\rangle} \oplus \frac{\mathbb{Z}_p[x]}{\langle h(x)\rangle}.$$

Its group of units $U(\mathbb{Z}_p[x]/\langle f(x)\rangle)$ is the direct product of groups of units $U(\mathbb{Z}_p[x]/\langle g(x)\rangle)$ and $U(\mathbb{Z}_p[x]/\langle h(x)\rangle)$, that is

$$U(\mathbb{Z}_p[x]/\langle f(x)\rangle \cong U\left(\frac{\mathbb{Z}_p[x]}{\langle g(x)\rangle}\right) \times U\left(\frac{\mathbb{Z}_p[x]}{\langle h(x)\rangle}\right).$$

Since $h(x)$ and $g(x)$ are irreducible, the quotient rings $\frac{\mathbb{Z}_p[x]}{\langle h(x)\rangle}$ and $\frac{\mathbb{Z}_p[x]}{\langle g(x)\rangle}$ are finite fields of order p^s and p^r, respectively. Hence, the groups $U(\mathbb{Z}_p[x]/\langle g(x)\rangle)$ and $U(\mathbb{Z}_p[x]/\langle h(x)\rangle)$ are cyclic with orders $\phi(h(x)) = p^s - 1$ and $\phi(g(x)) = p^r - 1$, respectively, so that $\phi(f(x)) = (p^s - 1)(p^r - 1)$. We provide the algorithms of the extended RSA signature over polynomials. First, entity A generates the public and private keys by doing the following.

Algorithm 103 *(Key generation for RSA signature over polynomials)*

1. Generating an odd prime p two distinct monic irreducible polynomials $f(x)$ and $g(x)$ over $\mathbf{Z}_p$.
2. Computing $h(x) = f(x).g(x)$
3. Computing the order of $U(\mathbf{Z}_p[x]/ < h(x) >)$ which is $\phi(h(x)) = (p^r - 1)(p^s - 1)$
4. Selecting a random integer e where $1 < e < \phi(h(x))$ such that $\gcd(e, \phi(h(x))) = 1$
5. Using the Euclidean algorithm for integers to find the unique multiplicative inverse d of e with respect to $\phi(h(x))$ such that $1 < d < \phi(h(x))$ and $e.d \equiv 1(\text{mod } \phi(h(x)))$
6. Publishing the key $(p, h(x), e)$ and keeping d as private key.

To generate a signature on a message, entity A should do the following.

Algorithm 104 *(Signature generation of RSA signature over polynomials)*

1. *Represent the message as a polynomial $m(x)$ in the complete residue system modulo $f(x)$ in $\mathbf{Z}_p[x]$.*

2. *Compute $\widetilde{m}(x) = R(m(x))$, as a polynomial in the complete residue system modulo $h(x)$ in $\mathbf{Z}_p[x]$.*

3. *Use the private key d to compute $s(x) = (\widetilde{m}(x))^d (\bmod h(x))$.*

4. *Output $s(x)$ as signature of $m(x)$.*

To verify the signature $s(x)$ and recover the real message $m(x)$, entity B should do the following.

Algorithm 105 *(Signature verification of RSA over polynomials)*

1. *Obtain A's public key $(p, h(x), e)$.*

2. *Compute $\widetilde{m}(x) = s^e$ $(\bmod h(x))$.*

3. *Verify that $\widetilde{m}(x) \in M_R$, otherwise reject the signature.*

4. *Recover $m(x) = R^{-1}(m(x))$ where R^{-1} is the inverse of the Redundancy function.*

Example 106 *(RSA Signature Scheme over Polynomials with small parameters)*

Public-Key Generation: Let $p = 389$. Entity A chooses the two irreducible polynomials $h(x) = x^2 + 376x + 43$ and $g(x) = x^3 + 384x^2 + 3x + 10$ in $\mathbb{Z}_{389}[x]$.

Reducing the polynomial $f(x) = h(x).g(x)$ *in* $\mathbb{Z}_{389}[x]$ *and computing* $\phi(f(x))$, *A gets* $f(x) = x^5 + 371x^4 + 111x^3 + 145x^2 + 388x + 41$ *and* $\phi(f(x)) = (389^3 - 1)(389^2 - 1) = 8907280505760$.*Entity A then chooses the integer* $e = 95561135039$ *such that* $\gcd(e, \phi(f(x)) = 1$ *and* $1 < e < \phi(f(x))$. *Using the extended Euclidean algorithm for integers, A finds* $d = 5878808345759$ *satisfying* $ed \equiv 1 (\bmod\, \phi(f(x)))$. *Hence, A's public-key is*

$$(p = 389, f(x) = x^5 + 371x^4 + 111x^3 + 145x^2 + 388x + 41, e = 95561135039).$$

and A's private-key is

$$(d = 5878808345759, g(x) = x^3 + 384x^2 + 3x + 10, h(x) = x^2 + 376x + 43).$$

Signature Generation: Choose $m(x) = 1 + 3x + x^2$ *and assume that the redundancy function is the identity function (for simplicity). Thus,* $\widetilde{m}(x) = 1 + 3x + x^2$. *Afterwards, A computes*

$$\begin{aligned} s(x) &= m(x)^d = (1 + 3x + x^2)^{5878808345759} \\ &\equiv 172x^4 + 86x^3 + 265x^2 + 59x + 177 (\bmod\, f(x)) \end{aligned}$$

and sends $s(x)$ *to B.*

Signature Verification: To validate the signature, B computes

$$\begin{aligned} \widetilde{m}(x) &= s(x)^e = (172x^4 + 86x^3 + 265x^2 + 59x + 177)^{95561135039} \\ &\equiv 1 + 3x + x^2 (\bmod\, f(x)). \end{aligned}$$

So, $\widetilde{m}(x) = 1 + 3x + x^2 \in M_R$. *Hence,* $m(x) = R^{-1}(1 + 3x + x^2) = 1 + 3x + x^2$.

Computer Program: (* We create a computer program with mathematica to illustrate the RSA signature over quotient rings of polynomials over finite field *)

```
(*Key generation*)
GeneratekPoly[l_] := (p = 101;
r1 = Random[ Integer, {21, 30}];
f = IrreduciblePolynomial[x, p, r1];
r2 = Random[Integer, {21, 30}];
g = IrreduciblePolynomial[x, p, r2];
Print["f=", f, "g=", g];
h = PolynomialMod[Expand[f*g], p];
Print["p=", p, " h=", h];
φ= (p^r1 - 1)(p^r2 - 1); Print["φ=", φ];
e =φ;
While[GCD[ e, φ] ≠1, e = Random[Integer, {1, φ - 1}]];
d = PowerMod[e, -1,φ];
Print["the public key is ( p=", p, ",h=", h,
", e=", e, ")"])
(*Signature generation*)
GenerateSPoly[l_] := (m1 = m;
s = PolynomialPowerMod[m1, d, {h, p}];
Print["the signature is ( s=", s, ")"])
(*Signature verification*)
```

```
VeriSPoly[ l_] := (ms = PolynomialPowerMod[s, e, {h, p}];

Print["ms=", ms])
```

4.4 RSA Signature Scheme Attack

The security of the RSA signature scheme is based on the intractability of both the integer factorization problem and the RSA problem. Various attack schemes have been studied in the literature as well as appropriate measures to counteract these threats. Given the public-key, to forge the signature, a passive adversary must solve the RSA problem. There is no known efficient algorithm for this problem. One possible approach which an adversary could employ is to find the private key. In order to attack any protocol that uses the RSA signature scheme by finding its private key, the factorization problem must be solved first. After factorization, the RSA problem could be solved by computing the value of Euler phi-function, and then finding the private exponent d using the extended Euclidean algorithm for integers. Once d is found, the signature can be forged.

On the other hand, if the classical method is used and an adversary could somehow compute d, then n can efficiently be factored as follows, see [22]. Since $ed \equiv 1 (\operatorname{mod} \phi(n))$, there is an integer k such that $ed - 1 = k\phi(n)$. Hence, by Euler theorem, $a^{ed-1} \equiv 1 (\operatorname{mod} n)$ for all a such that $\gcd(a, n) = 1$. Write $ed - 1 = 2^s t$, where t is an odd integer. It can be shown that $a^{2^{s-1}t}$ is not congruent to either ± 1 modulo n for at least half of all integers a with $\gcd(a, n) = 1$. If a is such an integer, then a non-trivial factor of n is $\gcd(a^{2^{s-1}t} - 1, n)$. This shows that in the classical case, the RSA problem and the integer factorization problem are computationally equivalent. It is not known if this remains true for the modified schemes.

In the next section we evaluate the various RSA signature schemes by recovering the private key using the software package Mathematica. We illustrate the attack schemes in the following example.

Example 107 *(Attacking the RSA signature scheme). Assume that the public key is:* $(n = 221806263006661919, e = 39786855994835377)$. *To find the private key, we use the built-in Mathematica functions FactorInteger and PowerMod. The prime factors* p *and* q *are obtained from the output of* $FactorInteger[221806263006661919]$ *which is* $\{\{315841909, 1\}, \{702269891, 1\}\}$. *Hence,* $p = 315841909$ *and* $q = 702269891$. *Next, we calculate* $\phi(n) = (p-1)(q-1) = (315841909-1)(702269891-1) = 404098131692231616$. *The exponent* $d = 279550294187496277$ *is the output of* $PowerMod[39786855994835377, -1, 40$
The private key is $(p = 315841909, q = 702269891, d = 279550294187496277)$.

4.5 Testing and Evaluation

In this section, we compare and evaluate the different classical and modified signature schemes by showing the implementation of the signature schemes'a lgorithms with their running results. Also, we test the security of the algorithms by implementing different attack algorithms. All this is done using Mathematica 5.0 as a programming language and an acer computer with Intel Pentium M715 processor, 1.5 GHZ CPU and 256 MB DDRAM.

RSA based Algorithms Using Mathematica 5.0 functions and an additional abstract algebra library, we have written programs for the following algorithms:

1. Classical RSA.
2. RSA with Gaussian integers.
3. RSA with polynomials over a finite field.

After running the programs, it was clear that these programs have applied the RSA signature scheme in the correct way. All the programs have generated a public and private key with different mathematical concepts. Then a message is signed using the signature scheme and is sent to a verification procedure which returned the original message.

The classical and Gaussian schemes were tested using the same public-key. The average running time of several runs using 50, 100, 200, 250 and 300-digit primes are given in tables 4.1 and 4.2. The public-key was generated by randomly selecting odd integers having a given number of digits and of the form $4k+3$. The odd integers were tested for primality using the built-in Mathematica function *PrimeQ* until a prime is found.

Table 4.1. Running time in seconds: Classical RSA

Size of primes	Classical RSA		
	Public-Key	Signature	Verification
50 – *digit*	0.1341	0.002	0.006
100 – *digit*	1.3801	0.011	0.0151
200 – *digit*	4.2913	0.0471	0.0851
250 – *digit*	5.7312	0.0923	0.1374
300 – *digit*	7.3706	0.144	0.2074

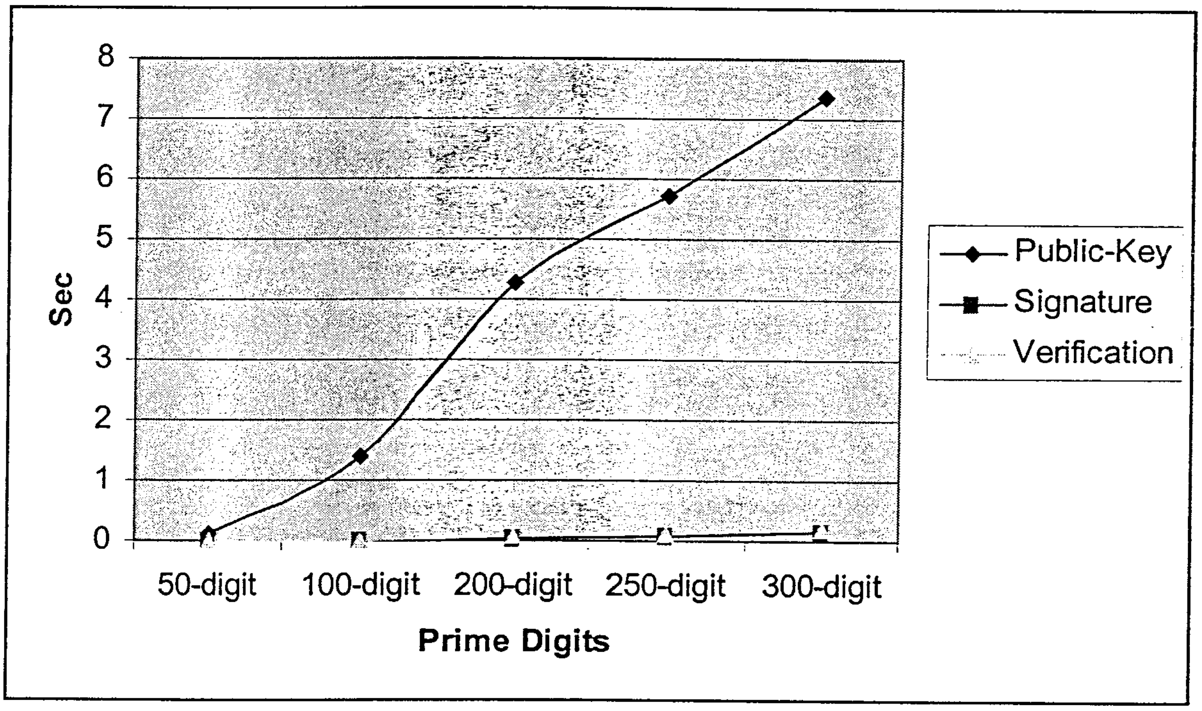

Figure 4.1: Running time in seconds: Classical RSA

Table 4.2. Running time in seconds: Gaussian integers.

Size of primes	RSA with Gaussian integers		
	Public-Key	Signature	Verification
50 – *digit*	0.1341	0.0912	0.097
100 – *digit*	1.3801	0.025	0.032
200 – *digit*	4.2913	0.1101	0.1513
250 – *digit*	5.7312	0.1883	0.2595
300 – *digit*	7.3706	0.3035	0.4238

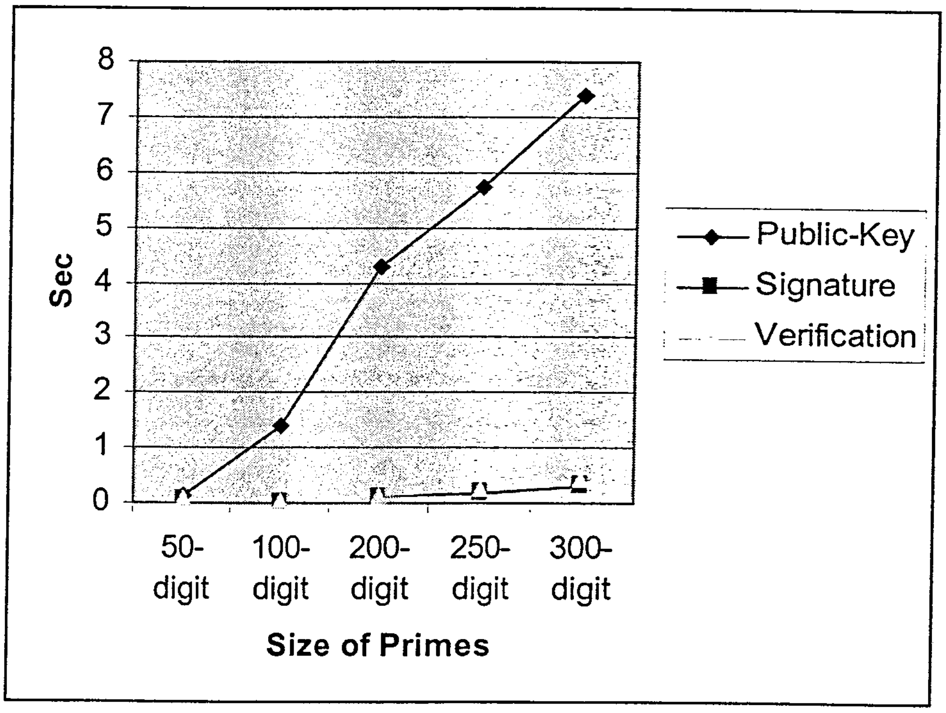

Figure 4.2: Running time in seconds: Gaussian integers

To evaluate RSA algorithms using polynomials, we ran programs for various values of the prime p and degree of the irreducible polynomials. The average running time of several runs are listed in table 4.3. The public-key was generated using the built-in Mathematica function IrreduciblePolynomial[x,p,d].

Table 4.3. Running time in seconds: RSA using polynomials.

	RSA Using polynomials			
Prime p	Degree d	Public-Key	Signature	Verification
$p = 2$	$2 \leq d \leq 10$	0.0331	0.0161	0.0331
	$21 \leq d \leq 30$	1.7188	0.9222	1.6863
	$50 \leq d \leq 60$	15.6215	8.8147	17.211
$p = 101$	$2 \leq d \leq 10$	1.429	0.3365	0.4305
	$11 \leq d \leq 20$	8.992	3.1823	5.53
	$21 \leq d \leq 30$	45.1559	13.792	14.21275

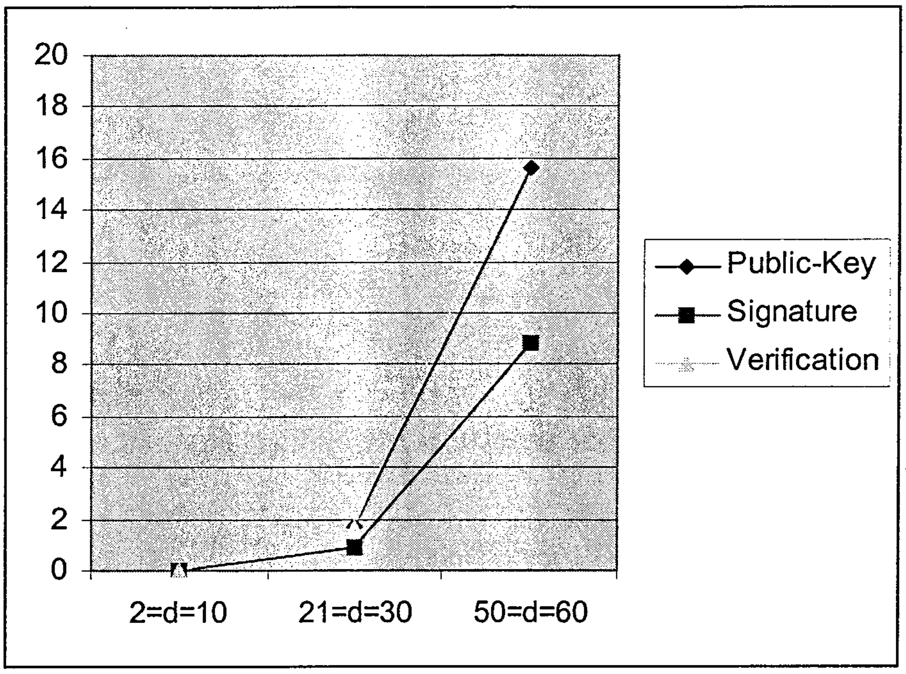

Figure 4.3:RSA using polynomials with prime p=2 digits

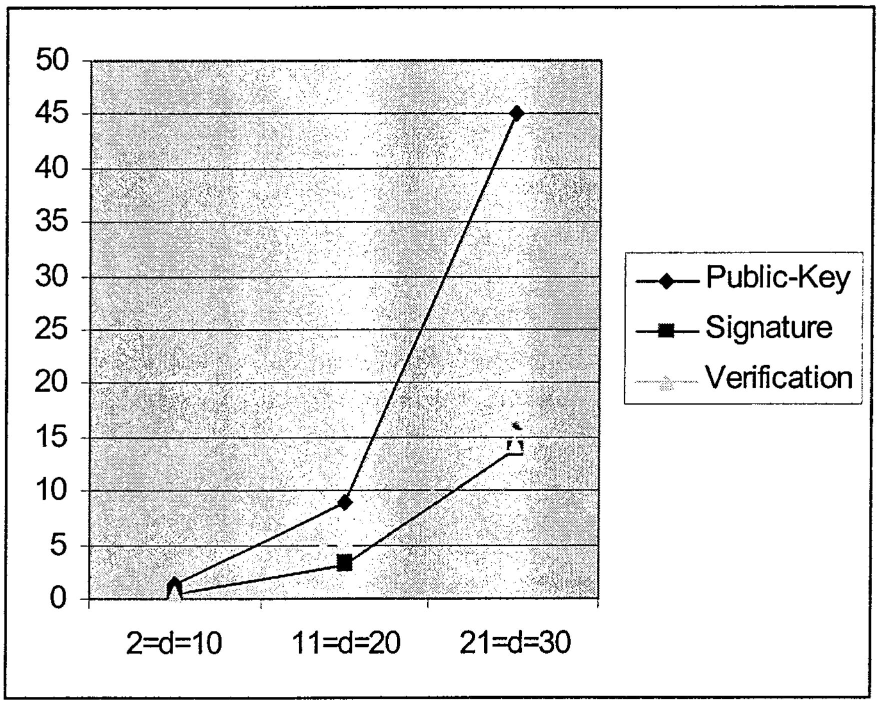

Figure 4.4: RSA using polynomials with prime p=101 digits

Comparing these algorithms, we conclude the following:

1. All programs are reliable; they can sign ,verify and return any message.

2. The running time for the signature/verification algorithms is negligible in the classical and Gaussian cases. In the polynomial case the time for the signature/verification algorithms becomes significant for large primes and irreducible polynomials with large degree.

3. The complexity for the three programs depends on the complexity of generating the public-key. Thus, the classical and Gaussian algorithms are equivalent since their public-key generation algorithms are identical when restricting the choice of primes to those of the form $4k + 3$. The Gaussian method is therefore recommended since the modified method provides an extension to the message space and the public exponent range.

4. The public-key generation algorithm using polynomials requires the search for irreducible polynomials. The Mathematica built-in algorithm for generating irreducible polynomials appears to be inefficient as p becomes very large and the degree of the polynomial increases.

Attack Algorithm In order to attack any protocol that uses the RSA public key signature scheme by finding its private key, the factorization problem must be solved first. To test the security of the algorithms, we implemented attack schemes applied to the classical and modified signature scheme algorithms. For the classical and Gaussian algorithms, we generated a public key using primes of various sizes. The attack was conducted using the Mathematica built-in function *FactorInteger* to recover the prime factors. The Euler phi-function was then computed. Finally, the private exponent was obtained. The average running time of several runs are listed in table 4.

Table 4.4. Attack time in seconds: Classical RSA.

	Classical RSA				
Digits of p & q	20	22	24	26	30
Time	1.406	4.3983	26.3238	65.0656	94.245

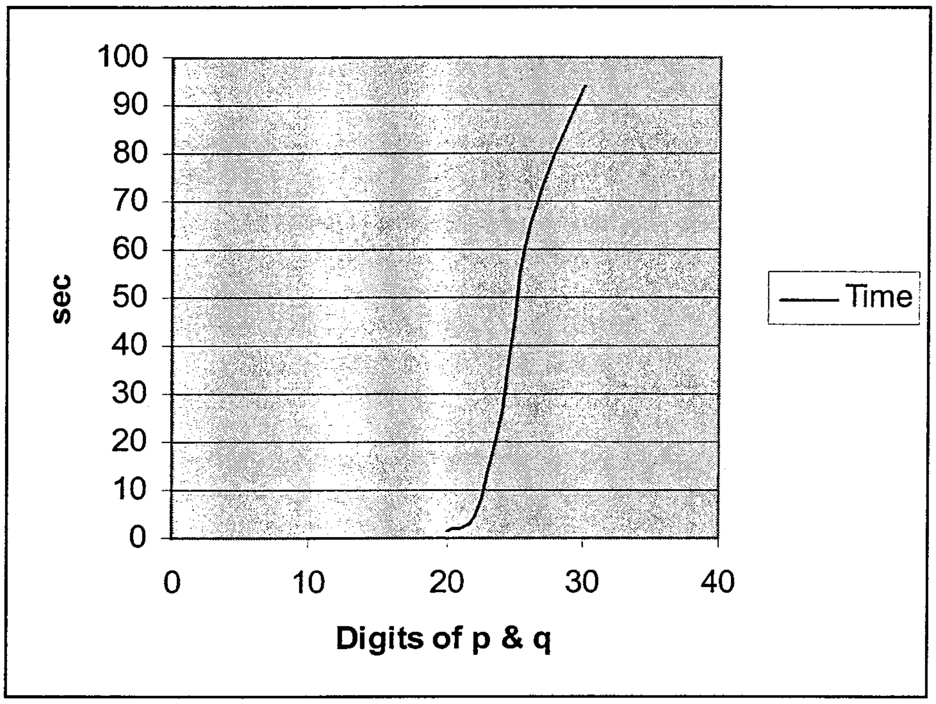

Figure 4.5: Attack time in seconds: Classical RSA

For the RSA algorithms using polynomials, we generated a public-key using a prime p of various sizes and irreducible polynomials $f(x)$ and $g(x)$ of different degrees d. The attack was conducted by factoring $f(x)$ using the built-in function

$Factor[f, \mathrm{mod}\,ulus- > p]$ to recover the irreducible factors. The Euler phi-function was then computed. Finally, the private exponent was obtained. The average running time of several runs are listed in table 4.5.

Table 4.5. Attack time in seconds: RSA algorithms using polynomials.

	RSA algorithms using polynomials			
Digits of p	5	5	10	22
Degree d	$10 \leq d \leq 11$	$12 \leq d \leq 13$	$5 \leq d \leq 6$	$2 \leq d \leq 3$
Time	2.373	2.954	0.651	0.231

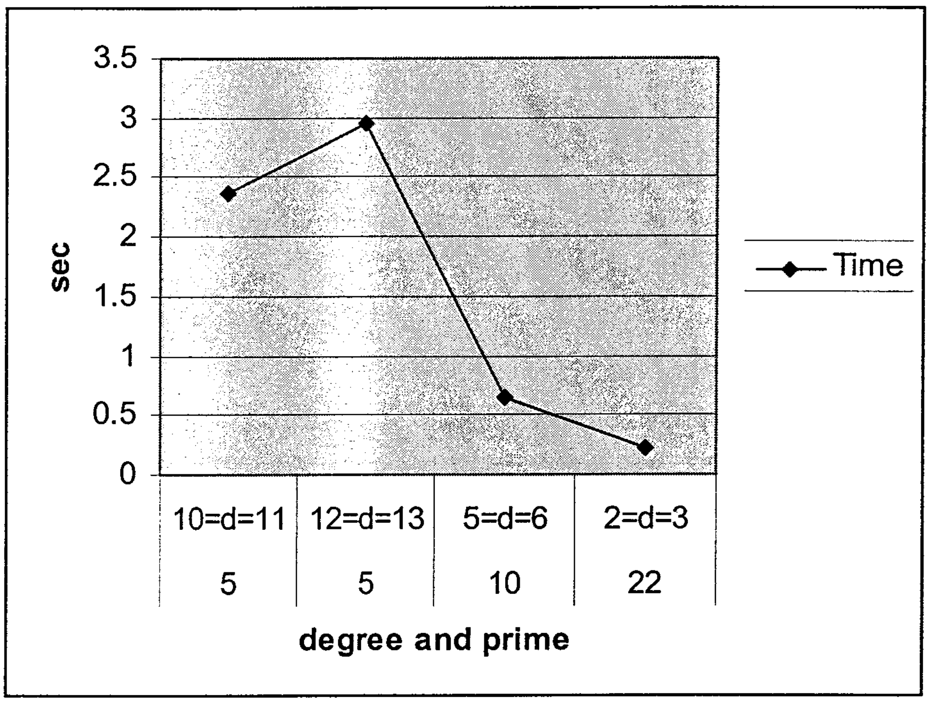

Figure 4.6: Attack Time in seconds: RSA algorithms using polynomials

After running these attack algorithms, we observed the following:

1. All the attack programs are reliable so that they can forge any message by

finding the private key.

2. Attacking the classical and Gaussian RSA algorithms is easy if we are dealing with small prime numbers. However, when it comes to 100-digit prime numbers or higher, it needs about many computers working in parallel processing to compute the prime factorization of the multiplication of two 100-digit prime numbers.

3. Attacking the RSA polynomial algorithm becomes more difficult as the size of p or the degree of the irreducible polynomials become larger.

4.6 Conclusion

In this work, we presented the classic RSA signature scheme and two of its modifications, namely, the RSA signature scheme in the domain of Gaussian integers, $Z[i]$, and over quotient rings of polynomials over finite fields. We implemented these algorithms and tested their efficiency, reliability, and security. The results obtained showed that all the algorithms applied the RSA signature scheme correctly and generated public and private key using different mathematical concepts. Messages were then signed using the signature scheme and were sent in encrypted form to a verification procedure which validated the signature and returned the original messages.

We also built attack scenarios directly aimed at solving the factorization problem. We modified the RSA attack algorithm to handle the modified algorithms.We observed that the Gaussian method is preferred since it is as secure as the classical one but provides an extension to the message space and to the signature exponent range.

CHAPTER 5

CONCLUSION

In this thesis we attempt to describe the concept of digital signatures ,the covenient mathematical techniques that securize and fasten the algorithms (redundancy functions and hash functions) and the mathematical background issues. Moreover, we extend some of the most practical and applied digital signature algorithms which are ElGamal signature and RSA signature from the domain of natural integers to the domain of gaussian integers and also to the domain of the rings of polynomials over finite fields. An overview of the prerequisite math involved in cryptographic applications is done. In addition, the mathematical issues in the domain of gaussian integers and the domain of polynomial rings over finite fields are also examined.Having done this, we explored the problems and insecurities involved in their use in addition to the advantages and disadvantages with respect to the classical signatures.

In chapter three,we presented the classic ElGamal signature scheme and four modifications to it, namely, the ElGamal signature scheme in Z_n, in the domain of Gaussian integers, $Z[i]$, over finite fields, and over quotient rings of polynomials over finite fields. We implemented these algorithms and tested their efficiency, reliability, and security. The results obtained showed that all the algorithms applied the ElGamal signature scheme correctly and generated public and private key using different mathematical concepts. Messages were then signed using the signature scheme and were sent to a verification procedure which verify the signature.

We also built attack scenarios directly aimed at solving the discrete logarithm problem that these algorithms utilize. We modified the Baby-step Giant-step algorithm to handle the modified algorithms. We observed that the classical ElGamal scheme is the weakest to attack and one of the modified methods should be used. The ElGamal scheme in the multiplicative group Z_n^*, where $n = 2p^2$, is the easiest to apply and the weakest among the modified method. The ElGamal scheme in the domain of Gaussian is superior to that of Z_n^* since it requires less time to generate a key, about the same time to sign and verify a signature, and is more secure. The ElGamal scheme in the setting of a finite field has a disadvantage in finding an irreducible polynomial. The reducible polynomial scheme was the most challenging to attack due to the mathematical complexity of arithmetic modulo a polynomial. Finally we found that the time for generating the key depends on finding the generator θ and not the prime p. The time for generating a prime number is negligible. The average time need for generating a 100-digit (recommended size) random prime is approximately 0.1 sec. Also it took more time to find the key in the case of polynomials. This will not be a problem if common system-wide parameters are used. In such a case, all entities may elect to use the same cyclic group G and generator ϕ. Also, once a generator ϕ for a given prime p is found, all other generators can be easily obtained. Moreover, the time needed to sign and to verify the signature for the classical, modified $2p^t$ and Gaussian is better than the time needed for the polynomials. However, the time is very reasonable even for larger primes. Overall, the Gaussian integers methods performed very well. The algorithms can be made more efficient by modifying Mathematica built-in functions to take advantage of the arithmetic modulo the Gaussian prime p of the form $4k + 3$. The key generation time in the case of Gaussian integers is less than that of the modified $2p^t$ method. This is due to the fact that the number of generators in $Z^*_{2p^2}$, which is $\phi(p(p-1))$, is almost always more than the number of generators in G^*_p, which is $\phi(p^2-1)$. In fact, among the first 200,000 primes, there are only 7 primes p of the form $4k+3$ for which $\phi(p^2-1) > \phi(p(p-1))$. Finally, the reducible polynomial method is little slower but provide more security. The irreducible polynomial method is not recommended since it is as secure as the reducible case but requires more time especially in finding the key.

In chapter four, a comparitive study of RSA based digital signature algorithm is done. Using the well known and arithmetics in the domain of Gaussian integers $\mathbf{Z}[i]$, the RSA signature scheme in the domain of natural integers was extended to $\mathbf{Z}[i]$. The computational properties in the new setting were described and the advantages of the new scheme were pointed out. The following are some of the advantages. First, generating the odd primes p and q of RSA signature in the domain of natural integers and in the domain of Gaussian integers requires the same amount of effort. However, the complete residue system $\mathbf{Z}_n$ has pq elements, while the complete residue system G_η has $\delta(pq)$ elements. If p and q are of the form $4k+3$, $\delta(pq) = p^2q^2$. If p is of the form $4k+3$ and q is of the form $4k+1$, $\delta(pq) = p^2q$. If p and q are of the form $4k+1$, which is similar to the classical method, $\delta(pq) = pq$. Hence, one of the two primes p and q should be of the form $4k+3$. Therefore, we deduce that the extended RSA over the domain $\mathbf{Z}[i]$ provides an extension to the range of chosen messages, which make trials more complicated. Second, note that the Euler phi function in the domain of natural integers is $\phi(n) = (p-1)(q-1)$, while the extended Euler phi function in the domain of Gaussian integers is $\phi(p\pi) = (p^2-1)(q-1)$ or $\phi(p_1p_2) = (p_1^2-1)(p_2^2-1)$

where π is a Gaussian integer that divides a prime integer $q = 4k+1$ such that $\pi\overline{\pi} = q$, and the prime integer $p = 4k + 3$ are the Gaussian primes. Hence RSA signature scheme in the domain of Gaussian integers is much better than that in the domain of natural integers if we choose at least one of the Gaussian primes p and q of the form $p = 4k + 3$. In this case, the value of $\phi(\eta)$ is larger than that in the domain of natural integers by a multiple of $(p+1)(q+1)$ times. Thus, the trials for finding the private key d will be too complicated. Fourth, the value of the public key e could be large enough to make solving the RSA problem more complicated. Finally, we note that the computations involved in the modified method do not require computational procedures that are different from those used in the classical method. For the RSA algorithms using polynomials, we generated a public-key using a prime p of various sizes and irreducible polynomials $f(x)$ and $g(x)$ of different degrees d. The attack was conducted by factoring $f(x)$ using the built-in function $Factor[f, \mod ulus- > p]$ to recover the irreducible factors. The Euler phi-function was then computed. Finally, the private exponent was obtained.After running the attack algorithms, we observed that all the attack programs are reliable so that they can forge any message by finding the private key. Attacking the classical and Gaussian RSA algorithms is easy if we are dealing with small prime numbers. However, when it comes to 100-digit prime numbers or higher, it needs about many computers working in parallel processing to compute the prime factorization of the multiplication of two 100-digit prime numbers. As a conclusion, attacking the RSA polynomial algorithm becomes more difficult as the size of p or the degree of the irreducible polynomials become larger.

As for future work, we plan to compare and evaluate the efficiency of the modified algorithms using very large numbers by using parallel computing techniques. We plan to run the programs in parallel on many computers and split the complex mathematical calculations between these computers. We plan to write a function that is capable of finding any random irreducible equation with respect to a specific prime number p. We also plan to apply the modified algorithms in many fields such as communications and network security.

REFERENCES

[1] Awad, Y. *Applications of Number Theory to Cryptography*, M.S. Thesis, Beirut Arab University, 2001.

[2]Cross, J. T. *The Euler's $\varphi - function$ in the Gaussian Integers*, American Mathematics Monthly 90, pp. 518-528, 1983.

[3] Diffie, W. and Hellman, M. E. *New Directions in Cryptography*, IEEE Transaction on Information Theory, IT-22, pp. 472-492, 1978.

[4] ElGamal, T. *A Public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory IT-31, pp. 469-472, 1985.

[5] ElGamal,"A public key cryptosystem and a signature sheme based on discree logarithms", IEEE Transactions on Information Theory IT-31 (1985), 469-472.

[6] El-Kassar, A. N., Haraty, Ramzi, *ElGamal Public – Key Cryptosystem in Multiplicative Groups of Quotient Rings of Polynomials over Finite Fields*, Journal of Computer Science and Information Systems, ConSIS Vol. 2, pp. 63-77, 2005.

[7] El-Kassar, A. N., Ramzi A. Haraty, *ElGamal Public – Key Cryptosystem Using Reducible Polynomials Over a Finite Field*, Proceedings of the 13th International Conference on Intelligent & Adaptive Systems and Software Engineering (IASSE-2004), pp. 189-194, 2004.

[8] El-Kassar, A. N., Ramzi Haraty, and Yehia Awad. *Modified RSA in the*

Domains of Gaussian Integers and Polynomials over Finite Fields. Proceedings of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA'04). Cairo, Egypt, 2004.

[9] El-Kassar, A. N., Haraty, Ramzi, Awad,Yehia, *Rabin Public–Key Cryptosystem in Rings of Polynomials over Finite Fields*, Proceedings of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA'04). Cairo, Egypt, 2004.

[10] El-Kassar, A. N., Mohamed Rizk, N. M. Mirza, Y. A. Awad, "*ElGamal Public-Key Cryptosystem in the domain of Gaussian Integers*",International Journal of Applied Mathematics, Vol. 7, no. 4, pp. 405-412, 2001.

[11] El-Kassar, A. N., Chihadi H., and Zentout D. *Quotient rings of polynomials over finite fields with cyclic group of units*, Proceedings of the International Conference on Research Trends in Science and Technology, pp. 257-266, 2002.

[12] El-Kassar, A. N., and Haraty, Ramzi. ElGamal publickey cryptosystem using reducible polynomials over a finite field, Proceedings of the ISCA 13th International Conference on Intelligent and Adaptive Systems and Software Engineering, ISCA 2004, Nice, France, 189-194, 2004.

[13] El-Kassar, A.N., *Modified RSA in the Domain of Gaussian Integers.* Proceedings of the 14th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE-2005). Toronto, Canada. July 2005

[14] El-Kassar, A. N.; Rizk, Mohamed; Mirza, N. M.; Awad, Y. A., *El-Gamal public key cryptosystem in the domain of Gaussian integers*, Int. J. Appl. Math. 7

(2001), no. 4, 405–412.

[15] El-Kassar A.N.,Doctorate Dissertation,University of Southwestern,Louisiana,1991.

[16] Haraty, R, Otrok,H., El-Kassar A.N., *A Comparative Study of ElGamal Based Cryptographic Algorithms*", Proceedings of the Sixth International Conference on Enterprise Information Systems (ICEIS 2004) vol. 3 , pp. 79-84, 2004.

[17] Haraty, R., El-Kassar, A. N., Otrok, H., *Attacking ElGamal Based Cryptographic Algorithms Using Pollard's Rho Algorithm*, Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2005). Cairo, Egypt. January 2005.

[18] Haraty, Ramzi, Hadi Otrok, A. N. El-Kassar, *A Comparative Study of RSA Based Cryptographic Algorithms*", Proceedings of the Sixth International Conference on Enterprise Information Systems (ICEIS 2004) vol. 3 , pp. 79-84, 2004.

[19] ISO/IEC 9796, "Information Technology-Security Techniques -Digital Signature Scheme Giving Message Recovery", International Organization for Standardization,1991.

[20] Kenneth, A. R. *Elementary Number Theory and its Applications*, AT&T Bell Laboratories in Murray Hill, New Jersey, 1988.

[21] Kojok,B., El-Kassar A.N., Raad, F., *Elgamal Signature Scheme In The Domain Of Gaussian Integers*, the Proceedings of the International Conference on Research Trends in Science and Technology, RTST 2002, Lebanese American University, Beirut Lebanon, 275-282, (2002).

[22] Menezes, A. J., Van Oorshot, and Vanstone, P. C. S. A. *Handbook of Applied Cryptography*, CRC press, 1997.

[23] Otrok, H. *Security Testing and Evaluation of Cryptographic Algorithms*, M.S. Thesis, Lebanese American University, June 2003.

[24] Prencil B., Covaerts R., Vandewall J., "Information Authentication: Hash functions and Digital Signatures", 1996.

[25] Preneel B., "Cryptographic hash functions", European transactions on Telecommunications,5, 431 - 448,1994.

[26] Rivest, R., Shamir A.,and Aldeman, L., *A Method for Obtaining Digital Signatures and PublicKey Cryptosystems*, Communications of the ACM 21, pp. 120-126, 1978.

[27] Raad, F. *Digital Signatures Algorithms and Cryptography Based on Finite Groups*, M.S. Thesis, Beirut Arab University, 2002.

[28] Rizk, M, El-Kassar, A. N., and Mirza, N., *Investigation And Comparison Between Elgamal Public-Key Cryptosystem And Its Extension To The Gaussian Integers Domain*, First International Conference on Current Issues in Business and Information Technology.

[29] Smith J. L. and Gallian, J. A. *FactoringFiniteFactorRings*, Mathematics Magazine 58: pp. 93-95, 1985.