A KNAPSACK-TYPE CRYPTOGRAPHIC SYSTEM USING ALGEBRAIC NUMBER RINGS

By

NATHAN THOMAS MOYER

A dissertation submitted in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
Department of Mathematics

MAY 2010

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of NATHAN THOMAS MOYER find it satisfactory and recommend that it be accepted.

William A. Webb, Ph.D., Chair

Bala Krishnamoorthy, Ph.D.

Judith J. McDonald, Ph.D.

A KNAPSACK-TYPE CRYPTOGRAPHIC SYSTEM USING ALGEBRAIC NUMBER RINGS

Abstract

by Nathan Thomas Moyer, Ph.D.
Washington State University
May 2010

Chair: William A. Webb

Since the advent of the public-key cryptosystem, the search for a secure and efficient scheme has been ongoing. One such system makes use of the classical knapsack problem to implement security. Though very efficient, this system was shown to be insecure. This paper aims to use the underlying knapsack protocol, without the vulnerabilities of previous systems.

To achieve such a result, two methods of constructing privates weights and encoding knapsack messages are posited. The first utilizes a new scheme to uniquely represent integers with recurrence sequences. The second uses specific congruence conditions of the weights to represent messages. Both techniques introduce constraints that cannot be modeled by Basis reduction. Thus eliminating its effectiveness.

In order to defend against any other specialty attacks, modular multiplication within an algebraic number ring is established and developed to disguise the private weights. This is ultimately a generalization of the disguising in the Merkle-Hellman system to integers in quadratic, bi-quadratic, and higher dimensional rings. Included in this development are issues of complete residue systems, calculating inverses, and number representations. The analysis of this modular multiplication reveals its immunity from attacks which seek to reverse or undo the disguising.

Table of Contents

Introduction

Cryptography can be simply defined as the practice and study of hiding information. Its primary goal is to achieve a secure means of transmitting information across an insecure communication channel. The general situation can be stated as follows:

Bob wants to send a secret message to Alice. However, he and Alice know that a third party, Eve, has the ability to eavesdrop on the message enroute. How can Bob be assured that the content of his message is *only* made known to Alice? This is where a secure method for hiding information is vital. Bob begins by converting the secret message, known as plaintext, into an unintelligible message, known as ciphertext. Through this encryption process, Bob conceals the content of the original message. The ciphertext is now transmitted to Alice, who having secret knowledge of the encryption process, is able to decrypt back to plaintext. Thus, Alice has retrieved and revealed the original message from Bob. Now if Eve were to intercept a copy of the transmission, she would only get the ciphertext. Without Alice's secret knowledge of decryption, Eve could not determine the original message. Hence, Bob and Alice would have achieved secure communication.

For thousands of years, methods of encryption and decryption have been developed based on this simple principle. The descriptions and implementations of these methods are called cryptographic codes or cryptosystems. The techniques for creating

these systems have varied dramatically over the years and require a closer look.

0.1 A Brief History of Cryptography

Modern cryptography has come a long way from its origins. Prior to the twentieth century, most ciphers converted plaintext into ciphertext by means of scrambling letters or substituting one letter for another. Of course, this had to be done in a systematic and predictable way, so the legitimate recipient could decrypt properly. These methods all require that both the sender and recipient have precise knowledge of the cryptographic scheme used, i.e. how the letters are permuted. This information is called the key and must be agreed upon by both parties before secure communication can take place. The key must be kept secret because it reveals all the information needed to decrypt an intercepted message.

Until 30 years ago, this framework, known as *secret key cryptography*, was the only way new codes were created. Yet, with the advent of computers and digital communication over insecure networks, a different framework was required to provide efficient and practical security. Secret key codes were not up to the task because they brought with them an inherent problem of key distribution.

A unique secret key is required for every pair of people who wish to communicate securely. If the number of pairs is large, the logistics of creating and distributing all the required keys becomes infeasible. For 100,000 people to securely communicate with each other over an insecure network it would require $\binom{100000}{2} \approx 5$ billion keys. The impracticality of distributing so many keys brought forth the quest for a new breed of cryptosystem.

In 1976, Whitfield Diffie and Martin Hellman published their ground breaking

paper "New Directions in Cryptography" which introduced the concept of *public key cryptography*[11]. This brand new paradigm opened up a whole new field of research within the cryptographic community. Within this framework, Alice has both a public key and a private key. To encrypt a plaintext message, Bob must access and use Alice's public key. That message can then only be decrypted by the private key, possessed by Alice. The two keys are mathematically related, but it is computationally infeasible to derive the private from the public. These types of cryptosystems are based on intractable mathematical problems such as factoring large numbers, solving a knapsack problem, or finding a discrete logarithm. Aside from being difficult to solve, the problems must also be easy to create, i.e. it is easy to multiply two large numbers together, but difficult to factor the product. These problems provide the underlying one-way function necessary for security. Due to the scarcity of these types of problems, it has proven to be a challenging task to develop brand new public key cryptosystems.

0.2 The RSA Code

One of the first public key codes introduced was the RSA (named after its inventors Rivest, Shamir, and Adleman)[33]. To create the keys with this system, Alice chooses two large prime numbers p and q and computes their product $n = pq$. She then selects a number, d, between $\min(p, q)$ and $(p-1)(q-1)$ that is relatively prime to $(p-1)(q-1)$ and computes its inverse, e, mod $(p-1)(q-1)$. So $ed \equiv 1 \bmod ((p-1)(q-1))$ or there is an r such that $ed = 1 + r(p-1)(q-1)$. The private key will consist of the numbers p, q, and d. The public key will be the numbers e and n. Bob represents his message in binary and converts it to the decimal message M. He then uses the

public key to compute the ciphertext $C \equiv M^e \pmod{n}$ and sends the message C. To decrypt, Alice uses the private key d and calculates M by $M \equiv C^d \pmod{n}$. The effectiveness of decryption is based on a well know result in number theory known as Euler's Theorem. It states that if $\gcd(x, n) = 1$, then $x^{\phi(n)} \equiv 1 (\text{mod } n)$, where $\phi(n)$ is the number of integers less than n that are relatively prime to n. For $n = pq$, we see that $\phi(n) = (p-1)(q-1)$. Thus, the decryption is successful because

$$\begin{aligned} C^d \mod n &\equiv (M^e)^d (\text{mod } n) \\ &\equiv M^{ed} (\text{mod } n) \\ &\equiv M^{1+r\phi(n)} (\text{mod } n) \\ &\equiv M(M^{\phi(n)})^r (\text{mod } n) \\ &\equiv M(1)^r (\text{mod } n) = M \end{aligned}$$

If Eve were to perform a factorization of $n = pq$ efficiently, then the private value of d could easily be found as well by solving the congruence

$$ex \equiv 1 (\text{mod } (p-1)(q-1)) \tag{0.2.1}$$

This would render the entire system insecure because the private key would now be known to Eve. Thus, the security of the RSA code rests on the fact the public value of n cannot easily be factored to yield the private values of p and q. For sufficiently large choices of n, this has shown to hold true over the years. Significant research has been done over the past decades in the area of factoring large integers and numerous techniques have been implemented [21]. However, the success of these methods remains incremental rather than dramatic. Thus, RSA has gained significant prominence as the most widely used and trusted public key cryptosystem.

In 1991, RSA Laboratories posed a challenge to the mathematical community to factor some specific large integers that are the product of two primes. As a result, there has been some success for numbers up to the range of about 200 digits (663 bits). Yet, these factorizations required significant time and computational power to acheive. A 232 digit integer was factored in December 2009 by a collection of parallel computers with the CPU time approximately equivalent to 1500 years on a single core 2.2 GHz processor with 2 GB RAM [34]. Due to the current limitations of factoring algorithms, the National Institute of Standards and Technology recommended a minimum RSA key size of 1024 bits to ensure security through 2010. Currently, those are being phased out for a key size of 2048 bits which is recommended through 2030 [31].

0.3 Knapsack Cryptosystems

The knapsack problem is a longstanding and well studied problem in the field of combinatorial optimization. Given a set of positive integer weights $A = \{a_1, a_2, ..., a_n\}$ and a positive integer T, find a subset of A whose elements sum to T. Stated an equivalent way, find a vector $\mathbf{x} = (x_1, x_2, ..., x_n)$ such that

$$\sum_{i=1}^{n} x_i a_i = T \tag{0.3.1}$$

where $x_i \in \{0, 1\}$ for $i = 1, 2, ..., n$.

Even though the knapsack problem is known to be NP-hard (nondeterministic polynomial-time hard), there are simple instances that are easily solvable. For example, if $a_i = 2^i$ then solving the knapsack problem is equivalent to finding the binary representation of T. The following definition provides another simple instance.

Definition 0.3.1. A sequence of positive integers $s_1, s_2, ..., s_n$ is called superincreasing

if $s_j > \sum_{i=1}^{j-1} s_i$ for $j = 1, ..., n$.

A superincreasing sequence has the property that every term of the sequence is strictly larger than the sum of the previous terms. This ensures that the sums of every possible subset of terms are distinct. Given a set S of knapsack weights with this structure and a number T, a solution (if it exists) will be unique and simply found by using a greedy algorithm. This provides the starting point the Merkle-Hellman system.

0.3.1 The Merkle-Hellman System

The Merkle-Hellman Cryptosystem [23], proposed in 1978, was the first public-key system introduced and made use of the classical knapsack problem. It is described as follows.

Alice wants to receive messages securely from Bob, so she begins by randomly creating a superincreasing set of knapsack weights $S = \{s_1, ..., s_n\}$. As shown above, this is an easy knapsack for Alice to solve. She will then transform S via a modular multiplication. That is, Alice will choose a multiplier c and a modulus m with gcd(c, m)=1, $m > \sum_{i=1}^{n} s_i$ and form the set $W = \{w_i\}$ where $w_i \equiv cs_i (\text{mod } m)$ for $1 \leq i \leq n$. The private key stored by Alice consists of the secret parameters that she chose, i.e., (S, c, m). The public key that is published for open access is the set W. Note that the set W will appear to consist of random integers, retaining none of the superincreasing structure of S.

To encode a message Z Bob must first convert it to binary in blocks of length n. To encrypt one of the blocks $\mathbf{x} = (x_1, x_2, ..., x_n)$ he computes $\sum_{i=1}^{n} x_i w_i = Z'$. He then transmits the encoded message Z' to Alice.

Once the message is received, Alice decrypts it by utilizing the private key information that she has available. First, she computes $d \equiv c^{-1} (\text{mod } m)$ so that $dc \equiv 1 \pmod{m}$. Since c and m are relatively prime,the inverse d exists. She can then compute

$$\begin{aligned} Z &\equiv dZ' (\text{mod } m) \\ &\equiv d \sum_{i=1}^{n} x_i w_i (\text{mod } m) \\ &\equiv d \sum_{i=1}^{n} x_i (c s_i) (\text{mod } m) \\ &\equiv (dc) \sum_{i=1}^{n} x_i s_i (\text{mod } m) \\ &\equiv \sum_{i=1}^{n} x_i s_i (\text{mod } m). \end{aligned}$$

Recall that m was chosen such that $m > \sum_{i=1}^{n} s_i$. Hence, Z is equal to $\sum_{i=1}^{n} x_i s_i (\text{mod } m)$ in standard integer arithmetic as well as mod m. The final step in the decoding process is to solve this easy superincreasing knapsack problem for $\mathbf{x}$. The resulting binary vector is exactly the original message from Bob.

Assume that Eve is able to intercept this message en route to Alice. What difficulties does she face in trying to decipher this message? Since she only knows the public knapsack W and the encrypted message Z', she must find the vector $\mathbf{x} = (x_i)$ where $\sum_{i=1}^{n} x_i w_i = Z'$. As mentioned before this is an NP-hard problem in general, so for sufficiently large values of n it is infeasible for Eve to determine $\mathbf{x}$. This fact provided the perceived security of the Merkle-Hellman system. Yet, as will be shown in the next chapter, this did not hold up well to further scrutiny.

Chapter 1

Attacks on Knapsack Codes

1.1 Introduction

The Merkle-Hellman knapsack cryptosystem survived for only a few years before it was ultimately defeated by various attacks. Similar knapsack systems were created soon after, but almost all have been shown to be insecure as well [6]. Assuming the parameters of a knapsack code are chosen large enough that any brute force attempt to solve the knapsack is infeasible, there are two main classifications for successful knapsack cryptosystem attacks.

1. *The man in the middle attack*: An encrypted message is intercepted and it's corresponding knapsack weights are determined. How the weights were created or disguised are irrelevant in this attack.

2. *Specialty attack*: The precise methods of creating and disguising the weights are examined, and a back door way is discovered to determine the private weights.

The actual implementation of these attacks are based on the concepts of *basis reduction* and *diophantine approximation*, respectively. These specific attacks will now be examined more closely.

1.2 Basis Reduction

Definition 1.2.1. An *integer lattice* L is an additive subgroup of $\mathbb{Z}^n$ that contains n linearly independent vectors over $\mathbb{R}^n$.

Definition 1.2.2. The set $\{\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_n\} \subset L$ is a *basis of* L, if $L = \text{span}\{\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_n\}$.

For any particular lattice, there are many different sets that qualify as bases, yet particular interest is given to a basis with very short vectors. Here, length is defined as the Euclidean norm $\|\mathbf{v}\|$ of the vector $\mathbf{v}=(v_1, v_2, ..., v_n)$ by

$$\|\mathbf{v}\|^2 = \sum_{i=1}^{n} v_i^2.$$

The LLL Algorithm, named after its creators Lenstra, Lenstra, and Lovasz, takes any basis of a lattice as input and finds a basis with short vectors, called a reduced basis [22]. In most cases the absolute shortest vector in the lattice will appear within the reduced basis. Thus, this algorithm may often find the shortest vector in a lattice.

By the construction of the Merkle-Hellman code, there is one unique solution to (0.3.1). That is, there is only one 0-1 vector $\hat{\mathbf{e}} = (e_1, e_2, ..., e_n)$ such that $\sum_{i=1}^{n} e_i w_i = Z'$. As it turns out, the shortest vector in a particular lattice generally corresponds to the solution vector $\hat{\mathbf{e}}$. As will be demonstrated, the knapsack problem can be transformed into the problem of finding the shortest vector in a lattice. Hence, the LLL algorithm can be seen as a direct method of solving the knapsack and breaking the security of the cryptosystem.

Lagarias and Odlyzko made use of the LLL algorithm in the following way to do that very thing. Their method for solving (0.3.1) is described in [19] and given below.

First, create a basis for an $n+1$ dimensional lattice L by using the vectors

$$\begin{aligned}
\mathbf{b}_1 &= (1, 0, ..., 0, -a_1) \\
\mathbf{b}_2 &= (0, 1, ..., 0, -a_2) \\
&\vdots \\
\mathbf{b}_n &= (0, 0, ..., 1, -a_n) \\
\mathbf{b}_{n+1} &= (0, 0, ..., 0, T)
\end{aligned}$$

Notice that L contains the solution vector $\hat{\mathbf{e}} = (e_1, e_2, ..., e_n, 0)$ since

$$\sum_{i=1}^{n} e_i \mathbf{b}_i + \mathbf{b}_{n+1} = (e_1, ..., e_n, -\sum_{i=1}^{n} a_i e_i + T) = \hat{\mathbf{e}}.$$

Also, $\|\hat{\mathbf{e}}\|$ is small, i.e. $\hat{\mathbf{e}}$ is short since $e_i \in \{0, 1\}$ for $1 \leq i \leq n$. In fact, $\hat{\mathbf{e}}$ is the shortest vector in L in most cases (especially with large weights a_i) and hence is likely to appear in a reduced basis. Thus, LLL is used on $\{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{n+1}\}$. Then the resulting reduced basis is checked for containment of the vector $\hat{\mathbf{e}}$, that is a solution to (0.3.1). Empirical results have shown that, in most situations, this basis reduction technique is successful in finding the solution vector. Additionally, theoretical bounds have been established to guarantee success in most cases([8],[19]).

The concept of density is now introduced to quantify the likelihood of success of basis reduction.

Definition 1.2.3. The *density* of a set of weights $\{a_1, a_2, ..., a_n\}$ is given by

$$d = \frac{n}{log_2(A)}$$

where $A = \max_{1 \leq i \leq n} a_i$.

For unique encryption, the density cannot exceed a certain limit. Any subset sum $\sum_{i=1}^{n} x_i a_i$ lies in $[0, nA]$ and there are 2^n ways to select the x_i values. Thus, if $2^n > nA$

or equivalently, $d > \frac{n}{n-log_2 n}$ there must be a value for which $\sum_{i=1}^{n} x_i a_i = \sum_{i=1}^{n} x'_i a_i$. By the superincreasing nature of the Merkle-Hellman private weights, clearly d cannot exceed 1. Plus, the disguise of modular multiplication will increase the size of the weights. Thus, we will always have $d < 1$. The size of d is negatively related to the effectiveness of Basis Reduction. With very small densities, the Basis Reduction attack has more success at breaking the code. The qualifier 'very small' has been quantified with an upper threshold. Lagarias and Odlyzko originally stated that any knapsack problem with $d < 0.6463...$ could be solved in polynomial time[19]. The most recent results found in [8] indicate that "almost all" knapsack problems with $d < 0.9408...$ can be solved in polynomial time.

1.3 Small Linear Dependencies

One potential weakness that arises from disguising relatively small weights with modular multiplication is the existence of many small linear dependencies among the weights. Among small numbers, it is common to find linear combinations with small coefficients that sum to zero. Since modular multiplication is a linear operation, this property carries over to the much larger disguised weights. This provides an avenue of attack to the cryptanalyst. For example, assume the set S_1 is transformed into the set S_2 via the following transformation.

$$S_1 = \{2, 5, 9, 19, 37\} \xrightarrow{43S_1 (\text{mod } 73)} S_2 = \{13, 69, 22, 14, 58\}$$

Notice that in $S_1, 2 \cdot 2 + 5 - 9 = 0$. Thus, correspondingly in $S_2, 2 \cdot 13 + 69 - 22 \equiv 0 (\text{mod } 73)$. From the cryptanalyst's point of view, a guess of small coefficients for weight in S_2 may yield *2*, *1*, and *-1*. This would be sufficient to determine that 73

$\equiv 0 (\text{mod } m)$, where m is unknown. Hence, the secret modulus is revealed as a factor of 73. Even though this is a small example, it illustrates a potential point of attack that must be eliminated in any future knapsack code.

1.4 Diophantine Approximation

Another successful attack against the Merkle-Hellman system is based on a simultaneous diophantine approximation problem. Unlike the basis reduction method, this attack does not attempt to solve the knapsack problem from an intercepted message. Instead, it seeks to reverse the disguising process on the weights, i.e. modular multiplication, thus revealing a superincreasing sequence and destroying the security of the code. This process is described in [18]. In the previous chapter, the description of the code reveals that Bob's decryption congruence is $dw_i \equiv s_i (\text{mod } m)$. This shows that the superincreasing sequence $\{s_i\}$ has the following structure:

$$s_i = dw_i - mk_i \qquad 1 \leq i \leq n \tag{1.4.1}$$

for nonnegative integers k_i. The goal of this attack is to determine a pair of integers $(\hat{d}, \hat{m})$ such that the weights $\hat{d}w_i (\text{mod } \hat{m})$ have a superincreasing structure. It is not necessary to find the original pair (d, m). It is only required that $\frac{\hat{d}}{\hat{m}} \approx \frac{d}{m}$. To observe this , let $\hat{d} = \tau d$ and $\hat{m} = \tau m + \epsilon$ where τ is a scaling factor and ϵ is a small error. Then,

$$\hat{d}w_i - \hat{m}k_i = \tau(dw_i - mk_i) - \epsilon\hat{k_i} = \tau s_i - \epsilon k_i$$

For a sufficiently small value of ϵ, this will retain a superincreasing structure. The field of Diophantine Approximation provides a method for attaining a ratio close to the unknown fraction $\frac{d}{m}$.

In general, the study of Diophantine Approximation deals with approximating real numbers with rational numbers. In particular, this analysis requires approximating a vector $\mathbf{a} = (\frac{a_1}{A}, ..., \frac{a_n}{A})$ of rational numbers with the same denominator, by another vector $\mathbf{p} = (\frac{p_1}{p}, ..., \frac{p_n}{p})$ of rational numbers with a smaller denominator. The vector to be approximated comes about in the following manner.

Dividing both sides of (1.4.1) by mw_i we obtain

$$\frac{s_i}{mw_i} = \frac{d}{m} - \frac{k_i}{w_i} \qquad 1 \leq i \leq n \tag{1.4.2}$$

Notice that the left side of this equality is very small for small values of i due to the fact that by construction $s_i < 2^{i-n}m$ and $m > \sum_{i=1}^{n} s_i$. Therefore, $\frac{d}{m} \approx \frac{k_1}{w_1}$. As described in [4], k_1 can be determined by solving an integer programming problem with a fixed number of variables. Hence, the known quantity $\frac{k_1}{a_1}$ is a very good approximation to $\frac{d}{m}$. Performing a search in a small interval around $\frac{k_1}{a_1}$ will yield a fraction $\frac{\hat{d}}{\hat{m}}$ which produces a superincreasing sequence, as shown above. The cryptanalyst can then use the decoding algorithm on the newly created sequence to reveal the message. In this way, all security has been lost and the system is rendered insecure.

In the creation of a new knapsack code, both of the above attacks must be considered. In Chapter 3, the framework for eliminating specialty attacks is established. In Chapter 4, two methods of constructing knapsack weights that are resistant to Basis Reduction are discussed.

Chapter 2

Other Knapsack-type Systems

After Merkle and Hellman's initial cryptosystem, numerous attempts have been made to utilize the knapsack problem as the means of achieving security. While the key generation varies from code to code, the constant across all of these systems is that the decryption process always involves finding a subset of weights that sum to a target number. Most of these systems have been shown to be insecure over time. These failures help to illuminate the inherent weaknesses within their designs and direct new code makers beyond where others have gone. Some recent systems are still facing scrutiny even today.

These systems are presented as examples of the type of ongoing research in this field. Some of the concepts introduced will be examined further along in this paper. A brief discussion of the important characteristics that will be incorporated into the development of a new code will follow most system descriptions.

2.1 Chor-Rivest Cryptosystem

One of the most famous and seemingly viable knapsack systems was created in 1988 by Ben-Zion Chor and his advisor Ronald Rivest. This was the first system of its type

that did not rely on a modular multiplication for disguising the weights. Instead, the public key generation relies on intricate finite field computations. To describe the system, we consider the finite field $GF(p^h)$ for some prime p and an integer h. Parameters of $p = 197$ and $h = 24$ are suggested in [7]. The field elements are represented as polynomials modulo $P(x)$, where $P(x)$ is a degree h irreducible polynomial over $GF(p)$. Additionally, the ordering of the subfield $GF(p)$ is public, i.e. $\{\alpha_0, ..., \alpha_{p-1} = GF(p)\} \subset GF(p^h)$. The secret key consists of the following:

- an element $t \in GF(p^h)$ of degree h
- a generator g of $GF(p^h)$
- an integer $d \in \mathbb{Z}_{p^h-1}$
- a permutation π of $\{0, 1, ..., p-1\}$

The public key consists of the following weights:

$$c_i = d + log_g(t + \alpha_{\pi(i)}) \bmod(p^h - 1) \qquad \text{for } i = 0, 1, ..., p-1.$$

Clearly, computing discrete logarithms in $GF(p^h)$ is a required operation to create the public weights. To ease the computations, p and h must be chosen carefully so that p is small and h has only small factors. With these conditions, the Pohlig-Hellman algorithm will be effective in these computations [32]. In order to encrypt a message, it must first be written as a 0-1 vector $\mathbf{m} = (m_0, ..., m_{p-1})$ such that $m_0 + ... + m_{p-1} = h$, that is, exactly h ones must be used. Note that this representation is different than that of Merkle-Hellman. It requires a special set of weights such that all sums of exactly h elements are distinct. The construction for such a sequence is given in [**?**].

The message is then encrypted in the usual way as

$$E(m) = m_0c_0 + ... + m_{p-1}c_{p-1} \bmod (p^h - 1).$$

In order to decrypt the message, one must compute

$$\begin{aligned} p(t) &= g^{E(m)-hd} \\ &= g^{\sum_{i=0}^{p-1} m_i c_i - hd} \\ &= \prod_{i=0}^{p-1} (g^{log_g(t+\alpha_{\pi(i)})})^{m_i} \end{aligned}$$

as a polynomial of degree at most $h-1$ over $GF(p)$, which equals

$$\prod_{i=0}^{p-1} (t + \alpha_{\pi(i)})^{m_i}.$$

Notice that exactly h of these terms will contribute to the product since $m_i = 1$ occurs h times and $m_i = 0$ occurs $p - h - 1$ times. Now for each i, m_i can be computed as the multiplicity of $\alpha_{\pi(i)}$ as a zero of this polynomial. Thus the original message $\mathbf{m}$ is recovered.

2.2 A Quantum Public-Key Cryptosystem

In [30], the authors introduced the idea of combining a public-key cryptosystem with the quantum computing model. The underlying quantum mechanism is Shor's discrete log algorithm, which is employed in the key generation stage (see [36] for details). The practical scheme proposed has an encryption and decryption process virtually identical to the Chor-Rivest system. The differences lie in the type of number structures being used. This new system uses a ring of integers over an algebraic number

field, whereas Chor-Rivest utilizes a ring of polynomials over a finite field. Essentially, this system generalizes the key generation process of Chor-Rivest to an arbitrary number ring K. Thus, instead of selecting an element $t \in GF(p^h)$ and a generator g of $GF(p^h)$, they select a prime ideal, $\mathbf{p}$, of the ring of integers O_K and a generator of the multiplicative group of the finite field $O_K/\mathbf{p}$. In addition, no information is revealed as to which number field is being considered. So, the authors claim there are exponentially many number fields, from which to select.

Both this system and the Chor-Rivest make use of the difficulty of computation within abstracted rings and fields to provide security. Conceptually, this is the same rationale behind the new systems presented in this paper. However, practically the differences of implementation are great.

2.3 A Knapsack System using the Chinese Remainder Theorem

In 2006, Yasuyuki Murakami and Takeshi Nasako developed a knapsack public-key cryptosystem that makes use of the Chinese Remainder Theorem to disguise their weights [26]. The integer weights are originally created in two distinct sets: $\mathbf{s}_P = \{s_i^{(P)}\}$ and $\mathbf{s}_Q = \{s_i^{(Q)}\}$ for $i = 1, ..., n$. The numbers in these sets are created to have a very precise bit structure so that the most and least significant bits are specifically

engineered to be 0 or 1 (See Figure 2.1). Then integers P and Q are chosen so that

$$P > \sum_{k=1}^{n} s_k^{(P)}$$
$$Q > \sum_{k=1}^{n} s_k^{(Q)}$$
$$\gcd(P, Q) = 1$$

and $N = PQ$.

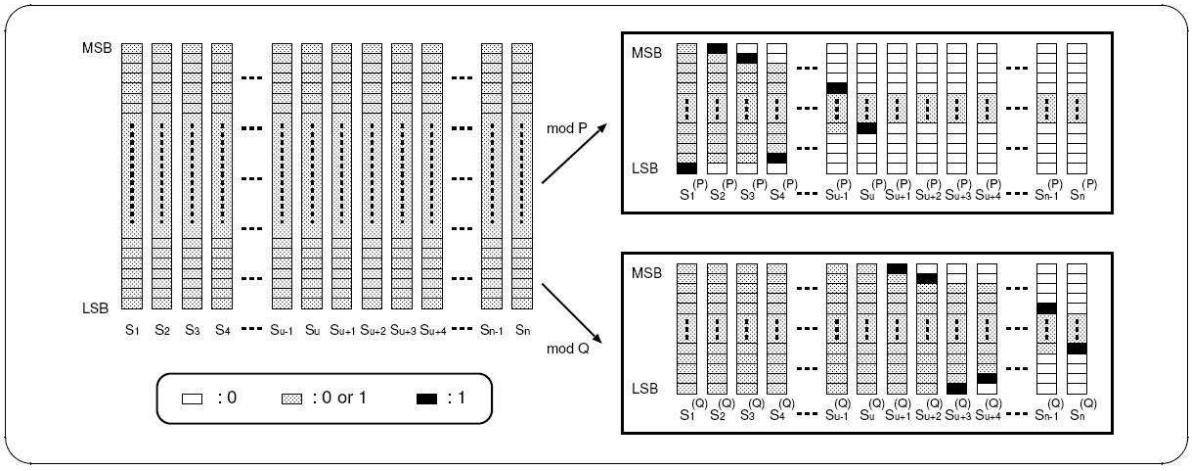


Figure 2.1: Concept of trapdoor using CRT

These two sets are then combined into one set $\mathbf{s} = \{s_i\} \in \mathbb{Z}_N^n$ by the Chinese Remainder theorem as follows:

$$s_i \equiv \begin{cases} s_i^{(P)} (\text{mod } P) \\ s_i^{(Q)} (\text{mod } Q) \end{cases}$$

These now become the public weights after a permutation σ. The public key consists of $\mathbf{s}$ and the private key is composed of $\mathbf{s}_P, \mathbf{s}_Q, P, Q, N$, and σ. Similar to the Chor-Rivest system, the pre-encrypted message is a 0-1 vector $\mathbf{m} = (m_1, ..., m_n)$ with a

fixed number of ones, and the encryption process is performed as follows:

$$C = \sum_{k=1}^{n} s_i m_i.$$

To decrypt, simply compute

$$\begin{aligned} C_P &= C(\text{mod } P) \\ C_Q &= C(\text{mod } Q) \end{aligned}$$

Then, because of the bit structure in the weights, a short decryption algorithm is used to reveal the original message **m**. Using the Chinese Remainder Theorem can be very effective as a means of disguising weights. A method similar to this will be discussed further in Section 4.1.3.

2.4 A Knapsack System built on NP-hard Instances

Recently, a cryptosystem was developed that claimed to be based upon NP-hard instances of a knapsack [12]. It differs from the other codes because it makes use of a bounded version of the knapsack problem. It seeks to find integers $\epsilon_i, 0 \leq \epsilon_i \leq M$ such that $\sum \epsilon_i a_i = T$ for weights $\{a_i\}$ and target T. Obviously, this is the basic knapsack problem when $M = 1$. The system also makes use of a one-way function based on the following remark: it is easy to produce divisions $n_i = qx_i + r_i$ with small remainders r_i, but it is more difficult to recover such divisions once the numbers n_i are given.

The message to be encrypted is a column vector $\mathbf{m} = \{m_1, ..., m_s\}$ where $m_i \in \{0, 1, ..., M-1\}$. To create the weights, begin with an $s \times s$ invertible integer matrix ϵ where the norm of the i^{th} row is given by $\|\epsilon_i\|_1 = \sum_{j=1}^{s} \epsilon_{ij}$. Then construct s positive

rational numbers $\lambda_i = \frac{p_i}{q_i}$ such that $(M-1)\lambda_i\|\epsilon_i\|_1 < 1$. The private key consists of ϵ and the λ_i's. The public key is created recursively as follows:

- Create a random row vector $\mathbf{x}_0$ with positive integer entries.
- Define the row vector $\mathbf{x}_i$ by $\mathbf{x}_i = q_i\mathbf{x}_{i-1} + p_i\epsilon_i$ for $i = 1, .., s$.

This gives the public key $\mathbf{x}_s$. To encrypt a message, simply compute the number $C(\mathbf{m}) = \mathbf{x}_s\mathbf{m}$.

To decrypt the message, do the following:

- Compute $N_{s-1}, ..., N_1$ with the formula $N_{i-1} = \lfloor \frac{N_i}{q_i} \rfloor$ where $\lfloor . \rfloor$ is the floor function.
- Compute $O_i = (N_i - q_iN_{i-1})/p_i$ and let $\mathbf{O}$ be the column vector with entries $O_1, ..., O_s$.
- Solve the system $\epsilon\mathbf{m} = \mathbf{O}$.

By construction (and verified in [12]) we have a system $\epsilon\mathbf{m} = \mathbf{O}$ which can be solved for the original message $\mathbf{m}$.

The security of the system is based on the difficulty of finding divisions with small remainders. From the private key, we know that $x_s = q_sx_{s-1} + p_s\epsilon_s$ with $q_s > p_s\|\epsilon_i\|_1(M-1)$. That is, $p_s\epsilon_{si}$ is a very small remainder of the division of x_{si} by q_s. In general, the sum of the remainders $p_s\epsilon_{si}$ for $1 \leq i \leq s$ is at most sq_s since each remainder is at most q_s. However, by this construction, the sum $\sum_i p_s\epsilon_{si} = p_s\|\epsilon_i\|_1$ of all remainders is at most $\frac{q_s}{M-1}$, which is unusually small.

This cryptosystem introduces two unique factors in play a role in design considerations. First, it provides a denser knapsack with coefficients larger than 0 and 1.

Second, it generalizes the idea of a single linear knapsack to a system of simultaneous knapsacks.

2.5 A Three Term Public-Key System

A new class of public-key cryptosystem is introduced in [25]. For this system, only three knapsack weights are used. The encryption process involves exponentiation of these weights. So, in a strict sense, this is not a knapsack code, but it is close enough in design to be classified as such.

The key generation process goes as follows:

- Generate random relatively prime $(h+1)$-bit integers b_1, b_2, b_3.
- Generate a random multiplier u and modulus N that are relatively prime. These along with the b_i constitute the secret key.
- Let $b'_i = (b_1 b_2 b_3)/b_i$ and create your public key weights a_i via the modular multiplication $a_i = ub'_i (\text{mod } N)$.

The messages to encrypt are h-bit positive integers m_1, m_2, and m_3. The ciphertext C is computed as $C = a_1 m_1^e + a_2 m_2^e + a_3 m_3^e$ where $e \geq 3$ is a fixed integer.

To decrypt, let $M = u^{-1} C (\text{mod } N)$. Then we see that M can be written in the form $M = b'_1 m_1^e + b'_2 m_2^e + b'_3 m_3^e$. Let $d_i \equiv e^{-1} (\text{mod } \lambda(b_i))$ where $\lambda(x)$ is the Carmichael Function i.e. the smallest positive integer t such that $c^t \equiv 1 (\text{mod } x)$ for every integer c relatively prime to x. We can obtain the messages m_1, m_2, and m_3 as follows:

$$m_i = (b_i'^{-1} M)^{d_i} (\text{mod } b_i).$$

For example, to decode m_1 we know that $ed_1 = k\lambda(b_1) + 1$ for some multiple k. Thus the decoding of m_1 will look like this:

$$\begin{aligned}
(b_1'^{-1}M)^{d_1} &= (m_1^e + b_1'^{-1}b_2'm_2^e + b_1'^{-1}b_3'm_3^e)^{d_1} (\text{mod } b_1) \\
&= m_1^{ed_1} (\text{mod } b_1) \\
&= m_1^{k\lambda(b_i)+1} (\text{mod } b_1) \\
&= m_1.
\end{aligned}$$

Chapter 3

Modular Arithmetic in Algebraic Number Rings

The purpose of this chapter is to develop the properties necessary to disguise weights in $\mathbb{Z}$ to weights in $\mathbb{Z}^n$. At it's core, this method is a generalization of the modular multiplication found in the Merkle-Hellman cryptosystem. What was constrained to the realm of integers, is now extended to an algebraic number ring. The chief motivation of this is to eliminate the effectiveness of diophantine approximation or specialty techniques of cryptanalysis. To combat BR attacks we will need to address the properties of the original undisguised weights. This will be discussed further in Chapter 4.

Definition 3.0.1. The complex number α is an *algebraic number of degree n* if it satisfies an n^{th} degree polynomial $p_\alpha(x)$ with rational coefficients.

For any algebraic number α we can create an extension field $\mathbb{Q}(\alpha)$ over $\mathbb{Q}$.

Definition 3.0.2. The $n-1$ other zeros of $p_\alpha(x), \alpha_2, \alpha_3, ..., \alpha_n$ are called the *conjugates* of $\alpha = \alpha_1$.

Definition 3.0.3. The *norm* of α, $N(\alpha)$, is the product of all the conjugates of α,

i.e. $N(\alpha) = \alpha_1, \alpha_2, ..., \alpha_n$.

With the following added restriction, we can identify a subset of algebraic numbers, called algebraic integers.

Definition 3.0.4. The number α is an *algebraic integer of degree* n if it is an algebraic number of degree n and $p_\alpha(x)$ is monic, irreducible over $\mathbb{Q}$, and has integral coefficients.

The set of all such numbers in the field $\mathbb{Q}(\alpha)$ make up $\mathbb{Z}[\alpha]$, the ring of integers in $\mathbb{Q}(\alpha)$.

Definition 3.0.5. For α, $\beta \in \mathbb{Z}[\theta]$, α *divides* β, or $\alpha|\beta$, if there is a $\gamma \in \mathbb{Z}[\theta]$ such that $\beta = \alpha\gamma$.

Some of the properties of norms are as follows within a degree n extension $\mathbb{Z}[\theta]$:

- $N(\alpha\beta) = N(\alpha)N(\beta)$
- if $\alpha|\beta$, then $N(\alpha)|N(\beta)$
- if $c \in \mathbb{Z}$, then $N(c) = c^n$
- if $N(\alpha) \in \mathbb{Z}$, then α is an algebraic integer

Definition 3.0.6. Let α, β, and $\gamma \in \mathbb{Z}[\theta]$. We say α is *congruent* to β modulo γ, written $\alpha \equiv \beta(\text{mod}\gamma)$, if there exists an integer $\delta \in \mathbb{Z}[\theta]$ such that $\alpha - \beta = \delta\gamma$.

Analogous to the concept of congruence in $\mathbb{Z}$, this produces an equivalence relation for a fixed γ that partitions the set of all numbers in $\mathbb{Z}[\theta]$. Each equivalence class is defined as follows.

Definition 3.0.7. A *complete residue system* modulo $\gamma \in \mathbb{Z}[\theta]$ is a nonempty subset S of $\mathbb{Z}[\theta]$ such that

1. No two numbers in S are congruent modulo γ.

2. Every number $\mathbb{Z}[\theta]$ is congruent to some element in S.

A complete residue system modulo γ will be will be abbreviated as CRS (mod γ). For rational integers in $\mathbb{Z}$, the standard CRS mod n is simply the set $\{0, 1, 2, ...n-1\}$. Of course, there are also other representations that allow for all members to be distinct mod n. In general, the question of how to describe the structure of complete residue systems of algebraic integers becomes more complicated. Geometrically, instead of being restricted to a number line, one must now consider regions of dimension 2, 3, and higher. Classifying and describing complete residue systems within the Gaussian integers $\mathbb{Z}[i]$ is given in [16]. Fortunately, the representation of a CRS becomes much more manageable when its modulus has a norm that is prime.

Theorem 3.0.1. *If γ is an algebraic integer of degree n and $N(\gamma)$ is prime, then a CRS (mod γ) $= \{0, 1, 2, ..., |N(\gamma)| - 1\}$.*

Proof. Let $S = \{0, 1, 2, ..., |N(\gamma)| - 1\}$. Let $a, b \in S, a > b$ such that $a \equiv b (\text{mod } \gamma)$. Then $c = a - b \equiv 0 (\text{mod } \gamma)$. Therefore, $\gamma | c$ and $N(\gamma) | c^n$. Since $N(\gamma)$ is prime, $N(\gamma) | c$. However, this is a contradiction since $c \in S$. Hence, $a \not\equiv b (\text{mod } \gamma)$. So S consists of $N(\gamma)$ non-equivalent numbers mod γ. □

Theorem 3.0.2. *Let $a, b \in \mathbb{Z}$ and let γ be an algebraic integer of degree n with $N = N(\gamma)$ prime. Then, $a \equiv b (\text{mod } \gamma)$ if, and only if $a \equiv b (\text{mod } N)$.*

Proof. $\Rightarrow$ Assume that $a \equiv b (\text{mod } \gamma)$. Then, $\gamma | (a - b)$ and hence $N | (a - b)^n$. Since N is prime, $N | (a - b)$. Therefore, $a \equiv b (\text{mod } N)$.

$\Leftarrow$ Assume that $a \equiv b(\text{mod } N)$. Then, $N|(a-b)$ or equivalently $\gamma\gamma_2\gamma_3\cdots\gamma_n|(a-b)$ where $\gamma_2\gamma_3\ldots\gamma_n$ are the conjugates of γ. Then clearly, $\gamma|(a-b)$. So $a \equiv b(\text{mod } \gamma)$.

□

For the scope of this paper, the focus will primarily be on algebraic integers of degrees 2 and 4, known as quadratic and biquadratic integers respectively. Higher degree algebraic integers will also be considered. The purpose of this development is not to provide an exhaustive and thorough analysis of these multi-dimensional rings. It is to build up some of the theories and properties necessary for cryptographic use such as computations, congruences, and complete residue systems.

3.1 Quadratic Integers

If a number α satisfies the equation $x^2 + bx + c = 0$ with $b, c \in \mathbb{Q}$, it must be of the form $\alpha = \frac{-b\pm\sqrt{b^2-4c}}{2}$. Let r be the discriminant, $b^2 - 4c$, of the polynomial with all square factors removed. Then $\alpha = l + m\sqrt{r}$ for $l, m \in \mathbb{Q}$. The set of all such quadratic numbers with a fixed value r forms a field denoted by $\mathbb{Q}(\sqrt{r})$. The subset of all quadratic integers within this field form a ring under addition and multiplication denoted by $\hat{\mathbb{Z}}(\sqrt{r})$. Further steps can be made in classifying quadratic integers by the following well-known result [37].

Theorem 3.1.1. *For any square-free $r \in \mathbb{Z}$, the following cases hold:*

1. *If $r \not\equiv 1(mod\ 4)$, then $\hat{\mathbb{Z}}(\sqrt{r}) = \mathbb{Z}(\sqrt{r}) = \{l + m\sqrt{r} \mid l, m \in \mathbb{Z}\}$*

2. *If $r \equiv 1(mod\ 4)$, then $\hat{\mathbb{Z}}(\sqrt{r}) = \mathbb{Z}(\sqrt{r}) \bigcup \{\frac{j+k\sqrt{r}}{2} \mid$ j,k are odd integers $\}$*

Depending on the value of $r(\text{mod } 4)$, every quadratic integer is in one of the two forms given. For the cryptographic purposes at hand, we will only be concerned with working in $\mathbb{Z}(\sqrt{r})$ which is a subring of $\hat{\mathbb{Z}}(\sqrt{r})$. Even though every quadratic integer may not be accounted for in this set, $\mathbb{Z}(\sqrt{r})$ still preserves the properties of a ring that will be vital. For this reason, we will disregard the remainder of r upon division by 4.

Since each quadratic integer α has only one conjugate, it is denoted by $\bar{\alpha}$. Based on the definitions above, we now specify the concept of conjugate and norm to the quadratic integers.

- $\overline{a + b\sqrt{r}} = a - b\sqrt{r}$
- $N(a + b\sqrt{r}) = a^2 - b^2 r$

Theorem 3.1.2. *If $N(a + b\sqrt{r})$ is prime, then gcd(a,b)=1.*

Proof. Assume that $gcd(a, b) = g \neq 1$. Then, $a = gh_1$ and $b = gh_2$ for $h_1, h_2 \in \mathbb{Z}$. Therefore, $a^2 - b^2 r = (gh_1)^2 - (gh_2)^2 r = g^2(h_1^2 - h_2^2 r)$. So $N(a + b\sqrt{r})$ is not prime. □

3.1.1 Modular Arithmetic

We now examine the concept of congruences of quadratic integers.

In order to begin classifying the congruence classes of $\mathbb{Z}(\sqrt{r})$, the following theorem is provided.

Theorem 3.1.3. *Let $N = N(m_1 + m_2\sqrt{r}) = m_1^2 - m_2^2 r$.*
If $a + b\sqrt{r} \equiv c + d\sqrt{r}(mod\ m_1 + m_2\sqrt{r})$, then $\begin{vmatrix} a & m_1 \\ b & m_2 \end{vmatrix} \equiv \begin{vmatrix} c & m_1 \\ d & m_2 \end{vmatrix} (mod\ N)$.

Proof. Suppose $a+b\sqrt{r} \equiv c+d\sqrt{r} (\text{mod } m_1+m_2\sqrt{r})$. Then there is an $x_1+x_2\sqrt{r} \in \mathbb{Z}(\sqrt{r})$ such that

$$(a+b\sqrt{r})-(c+d\sqrt{r}) = (m_1+m_2\sqrt{r})(x_1+x_2\sqrt{r}).$$

Thus, by equating the corresponding components of the numbers we get the following system:

$$a-c = m_1x_1+m_2x_2r \tag{3.1.1}$$

$$b-d = m_2x_1+m_1x_2 \tag{3.1.2}$$

Then, multiplying (3.1.1) by m_2 and subtracting the product of (3.1.2) and m_1 yields the equation $m_2(a-c)-m_1(b-d) = N(-x_2)$, which can be written as the congruence $am_2-bm_1 \equiv cm_2-dm_1 (\text{mod } N)$. Therefore, $\begin{vmatrix} a & m_1 \\ b & m_2 \end{vmatrix} \equiv \begin{vmatrix} c & m_1 \\ d & m_2 \end{vmatrix} (\text{mod } N)$. □

The converse of this theorem does not hold in general. To see this, consider $3+1\sqrt{2}$ and $6+3\sqrt{2}$ mod $6+4\sqrt{2}$. Clearly, $\begin{vmatrix} 3 & 6 \\ 1 & 4 \end{vmatrix} \equiv \begin{vmatrix} 6 & 6 \\ 3 & 4 \end{vmatrix} (\text{mod } 4)$, but $3+1\sqrt{2} \not\equiv 6+3\sqrt{2} \pmod{6+4\sqrt{2}}$, since this congruence would imply $3+2\sqrt{2} \equiv 0 \pmod{6+4\sqrt{2}}$. That would require integers x and y such that $6x+8y = 3$, which is clearly not possible. For this example, the modulus norm was 4. As shown in the next Theorem, the norm must be a prime for the converse to hold.

Theorem 3.1.4. *Suppose $N = N(m_1+m_2\sqrt{r})$ is prime. Then the converse of Theorem 3.1.3 holds.*

Proof. Assume $\begin{vmatrix} a & m_1 \\ b & m_2 \end{vmatrix} \equiv \begin{vmatrix} c & m_1 \\ d & m_2 \end{vmatrix} (\text{mod } N)$. Then, $am_2-bm_1 \equiv cm_2-dm_1 (\text{mod } N)$. This can be written as the congruence $(a-c)m_2-(b-d)m_1 \equiv 0 (\text{mod } N)$. Hence,

there is an $x \in \mathbb{Z}$ such that $(a-c)m_2 - (b-d)m_1 = x(m_1^2 - m_2^2 r)$. Thus, $m_1(xm_1 + b - d) = m_2(xrm_2 + a - c)$. Since N is prime, $gcd(m_1, m_2) = 1$ by Theorem 3.1.2. Now, m_1 divides the left side of the equation, so it must divide the right side. So, $m_1|(xrm_2 + a - c)$. Similarly, $m_2|(xm_1 + b - d)$. Then, since these divisibilities come from a single equation, there is one $y \in \mathbb{Z}$ such that

$$y = \frac{xrm_2 + a - c}{m_1} = \frac{xm_1 + b - d}{m_2}.$$

This yields the two equations:

$$\begin{aligned} a - c &= m_1 y + m_2 r(-x) \\ b - d &= m_2 y + m_1(-x) \end{aligned}$$

As seen in the proof of Theorem 3.1.3, this is equivalent to the desired congruence. □

Combining Theorems 3.1.3 and 3.1.4, we get the following condition for determining congruences.

Corollary 3.1.5. *Suppose $N = N(m_1 + m_2\sqrt{r})$ is prime. Then $a + b\sqrt{r} \equiv c + d\sqrt{r} (mod\ m_1 + m_2\sqrt{r})$ if and only if* $\begin{vmatrix} a & m_1 \\ b & m_2 \end{vmatrix} \equiv \begin{vmatrix} c & m_1 \\ d & m_2 \end{vmatrix} (mod\ N)$.

The following corollary is a specific case of Theorem 3.0.1 for a degree 2 extension.

Corollary 3.1.6. *Suppose $N = N(m_1 + m_2\sqrt{r})$ is prime. Then a CRS $(mod\ m_1 + m_2\sqrt{r})$ is the set $\{0, 1, 2, ..., N-1\}$.*

Proof. Let $a + b\sqrt{r} \in \mathbb{Z}(\sqrt{r})$ be arbitrary and let $k = \begin{vmatrix} a & m_1 \\ b & m_2 \end{vmatrix}$. Then the congruence $cm_2 \equiv k (\mathrm{mod} N)$ always has a solution for $c \in \{0, 1, ..., N-1\}$ since $gcd(m_2, N) = 1$. In addition, if $k_1 \equiv k_2$ for $k_1, k_2 \in \{0, 1, ..., N-1\}$, then $k_1 = k_2$. □

There are an infinite number of representations of residues given a particular modulus. What has been shown is that, given a prime norm N, every quadratic integer is congruent to a rational integer less than N. This provides the most convenient representation.

The next question that arises is how to calculate the rational integer representation. As in the previous proof, we see that $c \equiv m_2^{-1}k (\text{mod } N)$. Thus, given $a + b\sqrt{r}$ and $m_1 + m_2\sqrt{r}$ it is determined that $a + b\sqrt{r} \equiv c (\text{mod } m_1 + m_2\sqrt{r})$ where

$$c \equiv m_2^{-1} \begin{vmatrix} a & m_1 \\ b & m_2 \end{vmatrix} (\text{mod } N). \tag{3.1.3}$$

Another result that must be established with congruence classes is the existence and method of finding inverses. That is, given a quadratic integer $a + b\sqrt{r}$ find a nonzero $d \in \mathbb{Z}$ such that $(a+b\sqrt{r})d \equiv 1 (\text{mod } m_1 + m_2\sqrt{r})$. Notice that we only need to look for a *rational* integer inverse as long as the norm is prime. In order for the inverse d to exist it must satisfy the congruence $d \begin{vmatrix} a & m_1 \\ b & m_2 \end{vmatrix} \equiv m_2 (\text{mod } N)$.

If $\begin{vmatrix} a & m_1 \\ b & m_2 \end{vmatrix} \equiv 0 (\text{mod } N)$, then $a + b\sqrt{r}$ clearly has no inverse because it's CRS representation is 0. In that case, $a+b\sqrt{r}$ divides $m_1+m_2\sqrt{r}$. For any other instance, an explicit formula for calculating d is given as the following Theorem.

Theorem 3.1.7. *Suppose $N = N(m_1 + m_2\sqrt{r})$ is prime. Then the inverse, d, of $a + b\sqrt{r}$ (mod $m_1 + m_2\sqrt{r}$) is given by $d \equiv m_2 \begin{vmatrix} a & m_1 \\ b & m_2 \end{vmatrix}^{-1}$ (mod N).*

3.1.2 Quadratic Integers as Vectors

The ring $\mathbb{Z}(\sqrt{r})$ of quadratic integers can also be viewed as a lattice in the vector space $\mathbb{R}^2$ where each number $\alpha = a + b\sqrt{r}$ can be represented as the two-dimensional

vector $\mathbf{v}_\alpha = (a, b) \in \mathbb{Z}^2$. In this context, it is clear to see that any multiple of a number $a + b\sqrt{r}$ can be described as a linear combination of the two vectors (a, b) and (br, a), i.e.

$$(a + b\sqrt{r})(x + y\sqrt{r}) = x(a + b\sqrt{r}) + y(br + a\sqrt{r}).$$

Hence, these two vectors form the basis for this lattice.

Let $\alpha = a + b\sqrt{r}$ and $\beta = c + d\sqrt{r}$. We can create a correspondence between a number α and the 2×2 matrix $M_\alpha = \begin{pmatrix} a & br \\ b & a \end{pmatrix}$ so that multiplication within the ring of quadratic integers can be computed as a matrix-vector multiplication. That is,

$$\alpha\beta = \gamma \quad \Leftrightarrow \quad M_\alpha \mathbf{v}_\beta = \mathbf{v}_\gamma,$$

since $(a + b\sqrt{r})(c + d\sqrt{r}) = (ac + brd) + (bc + ad)\sqrt{r}$ and

$$\begin{pmatrix} a & br \\ b & a \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac + brd \\ bc + ad \end{pmatrix}.$$

Multiplication within the ring of quadratic integers can also be computed as a matrix-matrix multiplication. That is,

$$\alpha\beta = \gamma \quad \Leftrightarrow \quad M_\alpha M_\beta = M_\gamma$$

or

$$\begin{pmatrix} a & br \\ b & a \end{pmatrix} \begin{pmatrix} c & dr \\ d & c \end{pmatrix} = \begin{pmatrix} ac + brd & (bc + ad)r \\ bc + ad & ac + brd \end{pmatrix}.$$

In the matrix representation, we see that the columns form the basis for the CRS parallelogram. Additionally, the following convenient properties hold:

Theorem 3.1.8. *Let α and β be quadratic numbers and let M_α and M_β be their respective matrix representations. Then,*

- $M_\alpha + M_\beta = M_{\alpha+\beta}$
- $M_\alpha M_\beta = M_{\alpha\beta}$
- $det(M_\alpha) = N(\alpha)$
- *Eigenvalues of* $M_\alpha = \{\alpha, \bar{\alpha}\}$.

The matrix-vector multiplication will be more generally used throughout this paper. This provides an alternate context in which to discuss congruences. As we will see, this discussion will generalize more readily to higher dimensions.
A restatement of Corollary 3.1.5 can now be given as:

Theorem 3.1.9. *Let $\gamma = m_1 + m_2\sqrt{r}$ where $N(\gamma)$ is prime. Then $\alpha \equiv \beta (mod\ \gamma)$ if, and only if,*

$$\boldsymbol{v}_\alpha - \boldsymbol{v}_\beta = M_\gamma \boldsymbol{x} \text{ for some } \boldsymbol{x} \in \mathbb{Z}^2.$$

Proof. It is required that

$$M_\gamma^{-1}(\mathbf{v}_\alpha - \mathbf{v}_\beta) \in \mathbb{Z}^2.$$

This is equivalent to saying that

$$\tfrac{1}{N(\gamma)} \begin{pmatrix} m_1 & -m_2 r \\ -m_2 & m_1 \end{pmatrix} \begin{pmatrix} a-c \\ b-d \end{pmatrix} \in \mathbb{Z}^2$$

which creates the system

$$\begin{aligned} m_1(a-c) - m_2 r(b-d) &\equiv 0(\text{mod } N(\gamma)) \\ m_1(b-d) - m_2(a-c) &\equiv 0(\text{mod } N(\gamma)). \end{aligned}$$

So we have the two congruences:

$$\begin{vmatrix} a & m_1 \\ b & m_2 \end{vmatrix} \equiv \begin{vmatrix} c & m_1 \\ d & m_2 \end{vmatrix} (\text{mod } N(\gamma)) \tag{3.1.4}$$

$$\begin{vmatrix} a & m_2 r \\ b & m_1 \end{vmatrix} \equiv \begin{vmatrix} c & m_2 r \\ d & m_1 \end{vmatrix} (\text{mod } N(\gamma)). \tag{3.1.5}$$

Notice that (3.1.4) is simply the condition found in Corollary 3.1.5. Since $N(\gamma)$ is prime, (3.1.5) is an unnecessary added condition for two numbers to be congruent since it follows directly from (3.1.4). This is shown below. It can be assumed that $a \not\equiv c(\text{mod } N(\gamma))$ and $b \not\equiv d(\text{mod } N(\gamma))$, otherwise (3.1.5) follows trivially from (3.1.4).

Assume (3.1.4) holds. Then, $(b-d)m_1 \equiv (a-c)m_2(\text{mod } N(\gamma))$. Since $N(\gamma)$ is prime the inverse of m_1 mod $N(\gamma)$ exists. Thus we have

$$(b-d) \equiv (a-c)m_2 m_1^{-1}(\text{mod} N(\gamma)). \tag{3.1.6}$$

Now (3.1.5) is implied directly from this by the following argument:

$$\begin{aligned}
m_1^2 &\equiv m_2^2 r (\text{mod } N(\gamma)) \\
(a-c)m_1^2 &\equiv (a-c)m_2^2 r (\text{mod } N(\gamma)) \\
(a-c)m_1 &\equiv [(a-c)m_2 m_1^{-1}]m_2 r (\text{mod } N(\gamma)) \\
(a-c)m_1 &\equiv (b-d)m_2 r (\text{mod } N(\gamma)) \quad \text{(by substituting 3.1.6)} \\
\begin{vmatrix} a & m_2 r \\ b & m_1 \end{vmatrix} &\equiv \begin{vmatrix} c & m_2 r \\ d & m_1 \end{vmatrix} (\text{mod } N(\gamma)).
\end{aligned}$$

□

3.1.3 CRS of Quadratics

In [16], a depiction of various representations of complete residue systems is given for the Gaussian Integers, that is, numbers of the form $a + bi$. This is a subset of all quadratic integers of the form $a + b\sqrt{r}$ where $r = -1$. The purpose here is to generalize the structure of a CRS for any value of r. The view of quadratic integers as vectors helps to give a visual representation of a CRS. Consider the two vectors (a, b) and (br, a) plotted as arrows at the origin on the xy-plane. These vectors act as a basis for the set of vectors that are multiples of $a + b\sqrt{r}$. Hence, any integral linear combination of these vectors will produce a vector (algebraic integer) congruent to 0 mod $(a + b\sqrt{r})$. Adding an integral linear combination of (a, b) and (br, a) to any arbitrary (x, y) point in the plane will yield an integer congruent to $x + y\sqrt{r}$ mod $a + b\sqrt{r}$.

Then, the parallelogram whose diagonal is the sum of these vectors is the region containing a complete residue system (mod $a + b\sqrt{r}$). This parallelogram is called

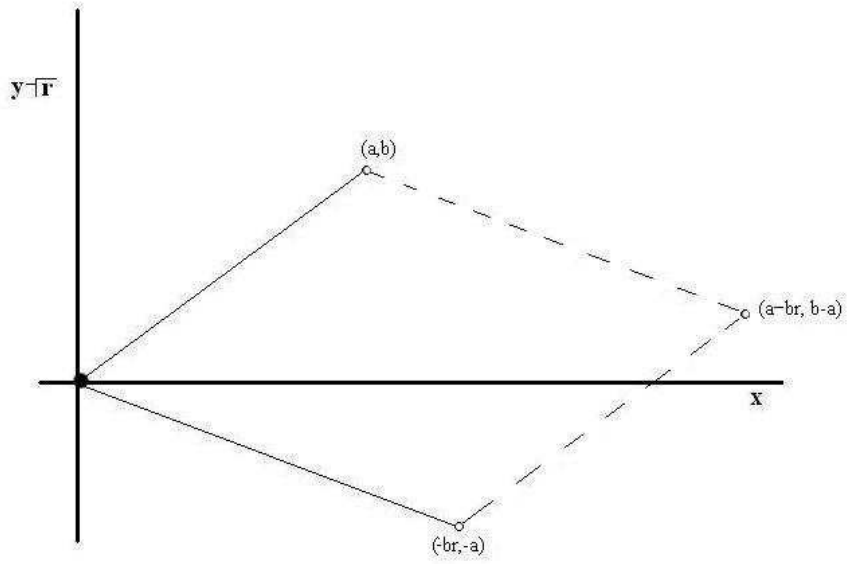


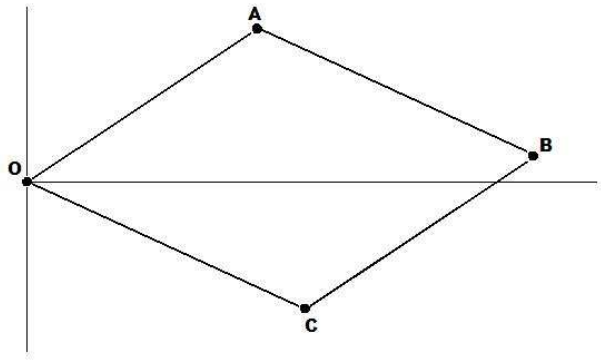
Figure 3.1: A quadratic CRS

the *fundamental CRS* mod $a + b\sqrt{r}$. Figure 3.1 shows such a region where $r < 0$ and $b > a > 0$.

Two numbers are congruent to each other mod $a + b\sqrt{r}$ if their difference can be expressed as a multiple of the vectors (a, b) and $(-br, -a)$. Thus, a number can be translated along these two vectors to reach any of its congruent forms. Since, (0,0) is congruent to (a, b), $(-br, -a)$, and $(a - br, b - a)$ these points are not included in the CRS. Also, since each pair of parallel line segments may contain congruent points, the solid lines will be included while the dashed are excluded. This is to ensure exactly one representative from each congruence class within the CRS. Then, every quadratic integer has one and only one point with integer coordinates as a representative mod $a + b\sqrt{r}$ within this parallelogram.

Theorem 3.1.10. *The norm of a quadratic integer α counts the number of incongruent quadratic integers modulo α.*

Proof. Consider the generic fundamental CRS given below.



This parallelogram is a simple polygon on a grid of integer coordinates with vertices on those grid points. Pick's Theorem [10] provides a formula for calculating the area A of a such a polygon in terms of the number I of interior points located in the parallelogram and the number of boundary points P on the parallelogram's perimeter. This formula is given as

$$A = I + \frac{P}{2} - 1.$$

The left hand side of this formula is given by $N(\alpha)$. Assume that the line segments $\overline{OA}$ and $\overline{CB}$ both have m boundary points and $\overline{OC}$ and $\overline{AB}$ both have n boundary points, not including the four vertices O, A, B, and C. Then, Pick's formula can be written as

$$N(\alpha) = I + \frac{2m + 2n + 4}{2} - 1 = I + m + n + 1$$

Now, the right hand side of the formula counts number of incongruent quadratic integers in the CRS as depicted in Figure 3.1, where the $+1$ indicates the inclusion of the origin. □

In order to maximize space efficiency, consider a rectangle with fixed bounds in the x and y coordinates that contains this CRS. This rectangle will provided a simple region that is guaranteed to contain all the residues of a particular modulus. One method of creating this rectangle is shown in Figure 3.2.

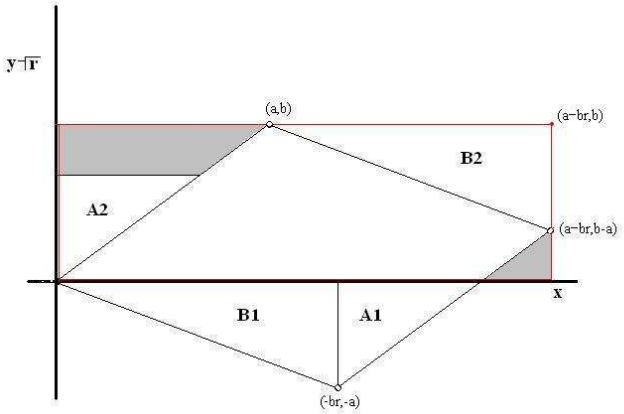


Figure 3.2: A minimal rectangle containing a quadratic CRS

The numbers in the region A1 are congruent to numbers in the region A2. To see this, simply add the vector (br, a) to any number in A1. Similarly, B1 and B2 contain congruent numbers. Therefore, considering only A2 and B2 along with the rest of the parallelogram, the rectangle $[0, a - br] \times [0, b]$ contains a CRS with limited wasted space in the shaded regions of the upper left and lower right corners. Recall that $r < 0$.

Another method of describing an efficient region with fixed bounds containing a CRS is to utilize a modulus $a + b\sqrt{r}$ such that a is significantly larger than br. In this case, our parallelogram will virtually take the shape of a rectangle as seen in Figure

3.3 with $a, b, r > 0$. Thus, the very little grey-shaded space is wasted in describing the rectangle $[0, a + br] \times [0, a + b]$ that contains a CRS.

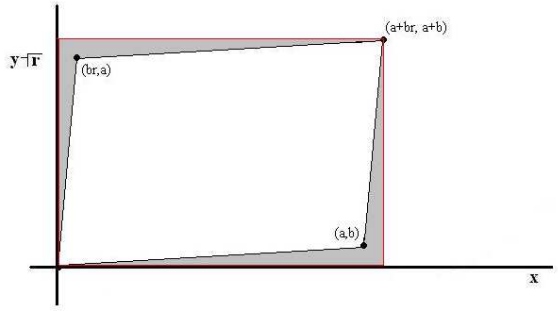


Figure 3.3: A quadratic CRS with large first component

To determine a particular quadratic integer representation that falls within the fundamental CRS (or bounding rectangle), it must satisfies Theorem 3.1.3 while restricting its components to within the desired range.

3.2 Bi-Quadratic Integers

Consideration will now be focused on algebraic integers that satisfy fourth degree polynomials. General descriptions of algebraic integers of higher degree are difficult to establish. In fact, beyond the fourth degree little, if any, is known in this area. The difficulty of representation is exhibited in a paper written in 1970 by Kenneth Williams[38]. In it he examined the form of bi-quadratic integers along with an integral basis in the field $\mathbb{Q}(\sqrt{m}, \sqrt{n})$. Without loss of generality, it can be assumed that $(m, n) \equiv (1, 1), (1, 2), (2, 3), (3, 3) (\text{mod} 4)$. Each of these cases produce a different

form of a bi-quadratic integer. For example, assuming m and n are relatively prime, if $(m, n) \equiv (1, 2)(\text{mod } 4)$, the integers are of the form:

$$\frac{1}{2}(x_0 + x_1\sqrt{m} + x_2\sqrt{n} + x_3\sqrt{mn}) \tag{3.2.1}$$

where $x_0 \equiv x_1(\text{mod } 2)$ and $x_2 \equiv x_3(\text{mod } 2)$. If $(m, n) \equiv (3, 3)(\text{mod } 4)$, the integers remain in the same form with the slightly different restriction that $x_0 \equiv x_3(\text{mod } 2)$ and $x_1 \equiv x_2(\text{mod } 2)$.

It is a much more difficult task to classify the set of all such numbers. Thus, we will only consider numbers in the following set with $gcd(r, s) = 1$:

$$\mathbb{Z}(\sqrt{r}, \sqrt{s}) = \{a + b\sqrt{r} + c\sqrt{s} + d\sqrt{rs} \mid a, b, c, d \in \mathbb{Z}\}$$

Clearly, this set does not account for all bi-quadratic integers, but it does consist exclusively of bi-quadratic integers. With addition and multiplication, this set becomes a sub-ring of the ring of all bi-quadratic integers.

Similar to the quadratic integers $\mathbb{Z}(\sqrt{r})$, this ring is viewed as a lattice in the vector space $\mathbb{R}^4$ where each number $\alpha = a + b\sqrt{r} + c\sqrt{s} + d\sqrt{rs}$ can be represented as the four-dimensional vector $\mathbf{v}_\alpha = (a, b, c, d) \in \mathbb{Z}^4$. In this context, it is clear to see that any multiple of a number $a + b\sqrt{r} + c\sqrt{s} + d\sqrt{rs}$ can be described as a linear combination of the four vectors $(a, b, c, d), (br, a, dr, c), (cs, ds, a, b)$ and (drs, cs, br, a), i.e.

$$\begin{aligned}
(a + b\sqrt{r} + c\sqrt{s} + d\sqrt{rs})(w + x\sqrt{r} + y\sqrt{s} + z\sqrt{rs}) &= \\
w(a + b\sqrt{r} + c\sqrt{s} + d\sqrt{rs}) &+ \\
x(br + a\sqrt{r} + dr\sqrt{s} + c\sqrt{rs}) &+ \\
y(cs + ds\sqrt{r} + a\sqrt{s} + b\sqrt{rs}) &+ \\
z(drs + cs\sqrt{r} + br\sqrt{s} + a\sqrt{rs}) &
\end{aligned}$$

Thus, we can identify the following matrix correspondence:

$$\alpha = a + b\sqrt{r} + c\sqrt{s} + d\sqrt{rs} \Longleftrightarrow M_\alpha = \begin{pmatrix} a & rb & sc & rsd \\ b & a & sd & sc \\ c & rd & a & rb \\ d & c & b & a \end{pmatrix} \tag{3.2.2}$$

Notice that each column of the matrix is produced by multiplying $\mathbf{v}_\alpha$ by $1, r, s$, and rs from left to right.

3.2.1 Bi-Quadratic Conjugates

Recall that $N(\alpha)$ is equal to the product of the conjugates of α. To determine the conjugate of $a + b\sqrt{r}$, one need only to negate the b term. A similar, but slightly more complicated method will be used to determine the three conjugates of $\alpha = a + b\sqrt{r} + c\sqrt{s} + d\sqrt{rs}$ with $a, b, c, d > 0$. Again, let a remain positive while b, c, and d vary between positive and negative. Since, $\alpha = \alpha_1$ and its conjugates $\alpha_2, \alpha_3, \alpha_4$ satisfy a fourth degree polynomial with integer coefficients, the coefficient of the x^3 term, $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$ must be in $\mathbb{Z}$. Hence, this sum must eliminate every term with $\sqrt{r}, \sqrt{s}$, and $\sqrt{rs}$. So two of the three conjugates must have a negative b, two of the three conjugates must have a negative c, and two of the three conjugates must have a negative d. This produces the following sign pattern for bi-quadratic conjugates:

$$\begin{array}{lcccc} \alpha_1: & + & + & + & + \\ \alpha_2: & + & - & - & + \\ \alpha_3: & + & - & + & - \\ \alpha_4: & + & + & - & - \end{array}$$

Now the product $\alpha_1\alpha_2\alpha_3\alpha_4$ is given symbolically as

$d^4r^2s^2 - 2b^2d^2r^2s + b^4r^2 - 2c^2d^2rs^2 - 2a^2d^2rs + 8abcdrs - 2b^2c^2rs - 2a^2b^2r + c^4s^2 - 2a^2c^2s + a^4$.

Calculating the symbolic determinant of M_α verifies the claim that $N(\alpha) = det(M_\alpha)$. Notice that b, c, and, d are all raised to even powers except for the $8abcdrs$ term, so negating two of them will not affect the norm.

3.2.2 Bi-Quadratic Representation

Consider the problem of finding an integer $k \in \mathbb{Z}$ such that $\alpha \equiv k(\mod \gamma)$, where $N(\gamma)$ is prime. This will occur if and only if, $\mathbf{v}_\alpha - \mathbf{v}_k = M_\gamma \mathbf{x}$ for some $\mathbf{x} = (x_1, x_2, x_3, x_4) \in \mathbb{Z}^4$, where $\mathbf{v}_k = (k, 0, 0, 0)$ and $\mathbf{v}_\alpha = (a, b, c, d)$. This process will extend the concepts introduced for quadratic integers. So it is required that

$$M_\gamma^{-1}(\mathbf{v}_\alpha - \mathbf{v}_k) \in \mathbb{Z}^4.$$

Obviously, M_γ^{-1} may not have integer entries, but since $N(\gamma) = det(M)$, we know that $S = N(\gamma)M_\gamma^{-1}$ does have integer entries. The matrix S is called the adjugate of the matrix M_γ. So, $\frac{1}{N(\gamma)}[S\mathbf{v}_\alpha - S\mathbf{v}_k] = \mathbf{x}$. Then if we represent the first row of S as the vector $\mathbf{s}$=(s_1, s_2, s_3, s_4), we see that $\mathbf{s} \cdot \mathbf{v}_\alpha - ks_1 = N(\gamma)x_1$. Hence we have the congruence

$$\mathbf{s} \cdot \mathbf{v}_\alpha \equiv ks_1 (\mathrm{mod} N(\gamma))$$

which can be solved for the unknown value of k as $k \equiv s_1^{-1}\mathbf{s} \cdot \mathbf{v}_\alpha (\mathrm{mod} N(\gamma))$. We know that s_1^{-1} exists since $N(\gamma)$ is prime. In determining k, we could have also considered the second row of S. In that case, a solution would have come in the form $k \equiv t_1^{-1}\mathbf{t} \cdot \mathbf{v}_\alpha (\mathrm{mod} N(\gamma))$ where t_1 is the first component of the second row vector $\mathbf{t}$.

Or we could have considered the third of fourth rows because they all produce the same solution. This now provides a computational method of reducing an arbitrary bi-quadratic integer to its equivalent representation in $\mathbb{Z}$.

Through a similar process we can demonstrate a method for determining inverses of an arbitrary bi-quadratic integer. Using the same notation as above, the goal is to find an $l \in \mathbb{Z}$ such that $l\alpha \equiv 1(\text{ mod } \gamma)$. Written in matrix-vector form, $l\mathbf{v}_\alpha - \mathbf{v}_1 = M_\gamma \mathbf{x}$ where $\mathbf{v}_1 = (1,0,0,0)$ and $\mathbf{x} \in \mathbb{Z}^4$. As above, this gives rise to $l\mathbf{s} \cdot \mathbf{v}_\alpha - s_1 = N(\gamma)x_1$ which produces the congruence $l \equiv s_1(\mathbf{s} \cdot \mathbf{v}_\alpha)^{-1}(\text{mod}N(\gamma))$. Notice that l is simply the inverse of $k(\text{mod } N(\gamma))$. This demonstrates that once a bi-quadratic integer is reduced to its rational integer form, inverses can be computed mod the norm quite simply.

3.2.3 CRS for Bi-Quadratics

A complete residue system of quadratic integers could be described as a set of rational integers or a set of quadratic integers within some fundamental parallelogram, called the fundamental CRS. The same is true of bi-quadratic integers, except now the fundamental CRS is a four dimensional parallelepiped. This is the natural extension of the quadratic case. The columns of the modulus matrix M_α determine the boundary of the parallelepiped. An important result found in [24] says that the number of integer points in a fundamental parallelepiped is equal to the volume of the parallelepiped. Within this context, the integer points are interpreted as elements of the complete residue system mod α and the volume of the parallelepiped is the norm of α, that is the $det(M_\alpha)$.

To illustrate a bi-quadratic integer CRS consider $1+\sqrt{2}+\sqrt{3}+\sqrt{6} \in \mathbb{Z}(\sqrt{2},\sqrt{3})$.

The norm of this integer is 4. Since 4 is not prime, the CRS is clearly not $\{0, 1, 2, 3\}$. To construct a CRS, determine a simpler representation of an arbitrary integer $5 + 2\sqrt{2} - \sqrt{3} + \sqrt{6} \pmod{1 + \sqrt{2} + \sqrt{3} + \sqrt{6}}$. That is, find $x_1, x_2, x_3, x_4 \in \mathbb{Z}$ such that

$$5 + 2\sqrt{2} - \sqrt{3} + \sqrt{6} \equiv x_1 + x_2\sqrt{2} + x_3\sqrt{3} + x_4\sqrt{6} \ (\text{ mod } 1 + \sqrt{2} + \sqrt{3} + \sqrt{6})$$

Writing this congruence as an equation and equating the components on both sides leads to the following system:

$$\begin{aligned} 5 - x_1 &= m_1 + 2m_2 + 3m_3 + 6m_4 \\ 2 - x_2 &= m_1 + m_2 + 3m_3 + 3m_4 \\ -1 - x_3 &= m_1 + 2m_2 + m_3 + 2m_4 \\ 1 - x_4 &= m_1 + m_2 + m_3 + m_4 \end{aligned}$$

for $m_1, m_2, m_3, m_4 \in \mathbb{Z}$. The solution for this system has the following form:

$$\begin{aligned} m_1 &= 5 - \frac{1}{2}x_1 + x_2 - 3x_3 + \frac{3}{2}x_4 \\ m_2 &= -\frac{9}{2} + \frac{1}{2}x_1 - \frac{1}{2}x_2 - \frac{3}{2}x_3 + \frac{3}{2}x_4 \\ m_3 &= -2 + \frac{1}{2}x_1 - x_2 - \frac{1}{2}x_3 + x_4 \\ m_4 &= \frac{5}{2} - \frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_4 \end{aligned}$$

The requirement of $m_i \in \mathbb{Z}$ forces x_1, x_3, and x_4 to have the same parity while x_2 has different parity. Thus, for the most simplified form, select $x_1 = x_3 = x_4 = 0$ and $x_2 = 1$. Therefore, $\sqrt{2}$ can be one representation in the CRS.

By the same technique, a different member of the CRS will require that x_1 and x_4 have one parity while x_2 and x_3 have a different parity. Thus, $1 + \sqrt{6}$ will be a representation in the CRS. All together, a CRS of $1 + \sqrt{2} + \sqrt{3} + \sqrt{6}$ can be written as $\{0, 1, \sqrt{2}, 1 + \sqrt{6}\}$.

CRS Bounding Box

Enclosing a fundamental parallelepiped within a four-dimensional rectangular prism provides a convenient method of describing a region containing all incongruent modular representations. This allows strict bounds in each dimension, yet can be inefficient due to enclosing multiple representations. This four-dimensional rectangular prism can be described as follows for the number $\alpha = a + b\sqrt{r} + c\sqrt{s} + d\sqrt{rs}$ with $a, b, c, d, r, s > 0$.

Let

$$\begin{aligned} B_1 &= max\{a, rb, sc, rsd\} \\ B_2 &= max\{b, a, sd, sc\} \\ B_3 &= max\{c, rd, a, rb\} \\ B_4 &= max\{d, c, b, a\}. \end{aligned}$$

Then every element of $\mathbb{Z}(\sqrt{r}, \sqrt{s})$ has a representative mod α within the region

$$\{(x, y, z, w) \in \mathbb{Z}^4 | 0 \leq x \leq B_1, 0 \leq y \leq B_2, 0 \leq z \leq B_3, 0 \leq w \leq B_4\}.$$

3.3 Higher Degree Representations

Every computation and procedure performed for quadratic and bi-quadratic integers may be extended to integers of higher degree. It is well known that finding a closed form for exact roots of arbitrary polynomials of high degree is impossible. Thus, we will restrict our scope of algebraic integers to account for those of a particular form, rather than a relation to a polynomial. This is necessary to establish a set whose integers preserve the first three properties listed in Theorem 3.1.8. For any $n \geq 2$, we

consider algebraic integers of degree n is of the form

$$\alpha = a_0 + a_1 r^{1/n} + a_2 r^{2/n} + ... + a_{n-1} r^{\frac{n-1}{n}} = \sum_{i=0}^{n-1} a_i r^{\frac{i}{n}}. \tag{3.3.1}$$

with $a_i \in \mathbb{Z}, r^{i/n} \notin \mathbb{Z}$ for $1 \leq i \leq n-1$. Notice the quadratic integers correspond to $n = 2$. The bi-quadratic integers, as well as many other large degree extensions of $\mathbb{Z}$ do not fit within this definition. This considers only numbers of the particular form given. The set of all numbers of the form in (3.3.1) constitutes a ring. Closure, the additive identity, and the additive inverses can easily be verified. Computationally, multiplication within this ring would be a dreadful task, with numbers in their current form. So we exhibit an isomorphic ring of matrices that allow for ease of practical computations. This is a generalization of 2×2 matrices used with quadratic integers back in Section 3.1.2. In general, each algebraic integer will have a matrix representation in a ring containing $n \times n$ matrices of the form:

$$M_\alpha = \begin{pmatrix} a_0 & a_{n-1}r & a_{n-2}r & \cdots & a_1 r \\ a_1 & a_0 & a_{n-1}r & \cdots & a_2 r \\ a_2 & a_1 & a_0 & \cdots & a_3 r \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{pmatrix}$$

This matrix is called persymmetric because it is symmetric in the northeast-to-southwest diagonal. As such, the product of two persymmetric matrices will be persymmetric. Due to its structure, M_α is completely determined by the n values $r, a_0, a_1, ..., a_{n-1}$. Consider a mapping ϕ defined such that $\phi(\alpha) = M_\alpha$. To ensure the operations of addition and multiplication are preserved, it will be verified that for any $\alpha = \sum_{i=0}^{n-1} a_i r^{\frac{i}{n}}$ and $\beta = \sum_{i=0}^{n-1} b_i r^{\frac{i}{n}}$,

1. $\phi(\alpha + \beta) = \phi(\alpha) + \phi(\beta)$

2. $\phi(\alpha\beta) = \phi(\alpha)\phi(\beta)$

Unlabeled sums are assumed to go from $i = 0$ to $n-1$. First,

$$\begin{aligned}
\phi(\alpha+\beta) &= \phi(\sum a_i r^{\frac{i}{n}} + \sum b_i r^{\frac{i}{n}}) \\
&= \phi(\sum (a_i + b_i) r^{\frac{i}{n}}) \\
&= \begin{pmatrix} a_0+b_0 & (a_{n-1}+b_{n-1})r & (a_{n-2}+b_{n-2})r & \cdots & (a_1+b_1)r \\ a_1+b_1 & a_0+b_0 & (a_{n-1}+b_{n-1})r & \cdots & (a_2+b_2)r \\ a_2+b_2 & a_1+b_1 & a_0 & \cdots & (a_3+b_3)r \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1}+b_{n-1} & a_{n-2}+b_{n-2} & a_{n-3}+b_{n-3} & \cdots & a_0+b_0 \end{pmatrix} \\
&= \begin{pmatrix} a_0 & a_{n-1}r & a_{n-2}r & \cdots & a_1 r \\ a_1 & a_0 & a_{n-1}r & \cdots & a_2 r \\ a_2 & a_1 & a_0 & \cdots & a_3 r \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{pmatrix} + \begin{pmatrix} b_0 & b_{n-1}r & b_{n-2}r & \cdots & b_1 r \\ b_1 & b_0 & b_{n-1}r & \cdots & b_2 r \\ b_2 & b_1 & b_0 & \cdots & b_3 r \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{n-1} & b_{n-2} & b_{n-3} & \cdots & b_0 \end{pmatrix} \\
&= M_\alpha + M_\beta = \phi(\alpha) + \phi(\beta)
\end{aligned}$$

To show the multiplicative condition, note that if $\sum a_i r^{\frac{i}{n}} \sum b_i r^{\frac{i}{n}} = \sum c_i r^{\frac{i}{n}}$ we must have $c_k = \sum_{i+j=k} a_i b_j + r \sum_{i+j=k+n} a_i b_j$ since

$$\begin{aligned}
\sum a_i r^{\frac{i}{n}} \sum b_i r^{\frac{i}{n}} &= (\sum_{i+j=0} a_i b_j + r \sum_{i+j=n} a_i b_j) + (\sum_{i+j=1} a_i b_j + r \sum_{i+j=n+1} a_i b_j) r^{\frac{1}{n}} + ... \\
&+ (\sum_{i+j=n-1} a_i b_j + r \sum_{i+j=2n-1} a_i b_j) r^{\frac{n-1}{n}}.
\end{aligned}$$

Observe that c_k can also be written as

$$c_k = \sum_{i=0}^{k} a_i b_{k-i} + r \sum_{i=k+1}^{n-1} a_{n+k-i} b_i \tag{3.3.2}$$

Thus,

$$\begin{aligned}\phi(\alpha\beta) &= \phi(\sum a_i r^{\frac{i}{n}} \sum b_i r^{\frac{i}{n}}) \\ &= \phi(\sum_{k=0}^{n-1} c_k r^{\frac{k}{n}})\end{aligned}$$

But the product

$$\begin{pmatrix} a_0 & a_{n-1}r & a_{n-2}r & \cdots & a_1 r \\ a_1 & a_0 & a_{n-1}r & \cdots & a_2 r \\ a_2 & a_1 & a_0 & \cdots & a_3 r \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{pmatrix} \begin{pmatrix} b_0 & b_{n-1}r & b_{n-2}r & \cdots & b_1 r \\ b_1 & b_0 & b_{n-1}r & \cdots & b_2 r \\ b_2 & b_1 & b_0 & \cdots & b_3 r \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{n-1} & b_{n-2} & b_{n-3} & \cdots & b_0 \end{pmatrix}$$

yields a matrix whose first column is exactly these c_k values. Hence,

$$\phi(\sum_{k=0}^{n-1} c_k r^{\frac{k}{n}}) = M_\alpha M_\beta = \phi(\alpha)\phi(\beta)$$

The process of finding an integer $k \in \mathbb{Z}$ such that $\alpha \equiv k(\mod \gamma)$, where $N(\gamma)$ is prime is now generalized to the high dimensional cases. This will occur if, and only if, $\mathbf{v}_\alpha - \mathbf{v}_k = M_\gamma \mathbf{x}$ for some $\mathbf{x} = (x_1, x_2, .., x_n) \in \mathbb{Z}^n$, where $\mathbf{v}_k = (k, 0, ..., 0) \in \mathbb{Z}^n$ and $\mathbf{v}_\alpha = (a_1, a_2, ..., a_n) \in \mathbb{Z}^n$. So we require that

$$M_\gamma^{-1}(\mathbf{v}_\alpha - \mathbf{v}_k) \in \mathbb{Z}^n.$$

As before, M_γ^{-1} does not have integer entries, but since $N(\gamma) =$det(M), we know that the matrix $S = [s_{ij}] = N(\gamma)M_\gamma^{-1}$ does have integer entries.

So, $\frac{1}{N(\gamma)}[S\mathbf{v}_\alpha - S\mathbf{v}_k] = \mathbf{x}$. This gives rise to the following congruences:

$$\begin{aligned}
\sum_{j=1}^{n} s_{1j}a_j &\equiv ks_{11}(\text{mod } N(\gamma)) \\
\sum_{j=1}^{n} s_{2j}a_j &\equiv ks_{21}(\text{mod } N(\gamma)) \\
&\vdots \\
\sum_{j=1}^{n} s_{nj}a_j &\equiv ks_{n1}(\text{mod } N(\gamma))
\end{aligned}$$

We know that each s_{i1}^{-1} exists since $N(\gamma)$ is prime. Hence, any one of these congruences can be solved for k. So k can be written as $s_{11}^{-1}\sum_{j=1}^{n} s_{1j}a_j(\text{mod } N(\gamma))$. This produces a unique representation k of α where $0 \leq k \leq N(\gamma) - 1$.

In the same way we can demonstrate a method for determining inverses of an arbitrary algebraic integer of the form given in (3.3.1). Using the same notation as above, the goal is to find a $l \in \mathbb{Z}$ such that $l\alpha \equiv 1(\text{ mod } \gamma)$. Written in matrix-vector form, $l\mathbf{v}_\alpha - \mathbf{v}_1 = M_\gamma\mathbf{x}$ where $\mathbf{v}_1 = (1, 0, ..., 0) \in \mathbb{Z}^n$ and $\mathbf{x} \in \mathbb{Z}^n$. As above, this gives rise to $\frac{1}{N(\gamma)}[lS\mathbf{v}_\alpha - S\mathbf{v}_1] = \mathbf{x}$. which produces the congruence $l \equiv s_{11}(\sum_{j=1}^{n} s_{1j}a_j)^{-1}(\text{mod } N(\gamma))$. Notice that l is simply the inverse of $k(\text{mod } N(\gamma))$. Hence, inverses can be computed mod the norm quite simply once the algebraic integer has been reduced to its integer form.

3.4 The Disguising Method

One of the fundamental weaknesses of knapsack codes has been the inability to adequately disguise the "easy" weights into the "hard" weights. This has been demonstrated in the success of Diophantine Approximation attacks, where the disguising

has been reversed. Thus, the efforts to combat this must lie in the complexity of the disguising process. To achieve added complexity, we consider disguising weights within the context of a ring of algebraic integers. There is a cryptographic purpose to removing modular multiplication from the ordinary integers and placing it in the setting of algebraic number rings. This is done to frustrate attacks that seek to reverse the disguising of weights in this manner. The added complexity serves to eliminate the possibility of modeling the modular multiplication, as was done with the Diophantine Approximation attacks on previous codes.

The goal is to transform a sequence of integers $\{a_i\} \in \mathbb{Z}$ into a sequence of vectors in $\mathbb{Z}^n$ through a system of modular multiplication using algebraic integers for the moduli. While the private weights remain integers, our public weights will now be a set of vectors of size n with integral components. Due to the linearity of such a transformation, sums will be preserved. That is, a sum of private weights will correspond to the exact same sum of public weights, and vice versa.

3.4.1 Modular Multiplication

Let $\{a_i\} \in \mathbb{Z}$ be a sequence. Then two secret parameters must be established prior to disguising. Let $c \in \mathbb{Z}$ be the multiplier and let the modulus be an algebraic integer γ of degree n. This description will be general enough to account for quadratic, bi-quadratic, or higher dimension algebraic integers. Define a function $f : \mathbb{Z} \to \mathbb{Z}^n$ by $f(a_i) = a_i c (\mathrm{mod} \gamma)$. This function is applied to each of the terms in the sequence, resulting in a sequence of vectors. Clearly, there are many possible representations of each $f(a_i)$. In order for this function to be well-defined, this representation must lie in the fundamental CRS (mod γ). (For practical implementation, it is sufficient

to select any representation in a bounding rectangle of the fundamental CRS.) Call that representative β_i.

In order to invert this function f one must first compute $c^{-1}(\text{mod } N(\gamma))$. We are justified in using the norm of γ by Theorem 3.0.2. Then define $f^{-1} : \mathbb{Z}^n \rightarrow \mathbb{Z}$ by $f^{-1}(\beta_i) = \beta_i c^{-1}(\text{mod}\gamma)$. Here of course, the representative is given as an ordinary nonnegative integer less than $N(\gamma)$.

This modulus γ must be chosen such that (1) $N(\gamma)$ is prime and (2) $N(\gamma) > \sum a_i$. Condition (1) is required so that β_i is congruent to a unique integer less than $N(\gamma)$. Condition (2) is necessary to ensure a unique inversion of any subset sum of the a_i weights. Consider some $T = \sum x_i a_i > N(\gamma)$ such that $f(T) = \beta$. Then $f^{-1}(\beta) < N(\gamma)$. So $f^{-1}(\beta) \neq T$ and the T could not be recovered through this process.

Quadratic Trees

In Section 3.1.1 modular multiplication was described in a quadratic number ring. It was shown that an integer in $\mathbb{Z}$ can be transformed into a quadratic integer located within a fundamental parallelogram. This was also shown to be a reversible transformation from $\mathbb{Z}$ to $\mathbb{Z}^2$. This transformation can also be applied to each component of vectors in $\mathbb{Z}^2$, to create another vector in $\mathbb{Z}^2$. This procedure can be repeated indefinitely.

To visualize this process, consider a binary tree. The nodes of the tree are integers and each of the child pairs are produced via a modular multiplication on the parent node. Thus, each internal vertex corresponds to a modular multiplication. Each child is a component of a vector in $\mathbb{Z}^2$. The root of the tree is the original integer that begins the process. This would be considered one of the private weights. So there

is an ability to create a wide variety of tree structures based upon which nodes are chosen to reproduce.

For example, Figure 3.4 depicts a transformation from $\mathbb{Z}$ to $\mathbb{Z}^4$ in the following way. The integer a is transformed into the vector (b_1, b_2). Then b_2 is transformed into the vector (c_1, c_2). Finally, c_1 is transformed into the vector (d_1, d_2). So, ultimately, a has been transformed into $(b_1, d_1, d_2, c_2) \in \mathbb{Z}^4$. The leaves of the tree, from left to right, constitute the final disguised vector.

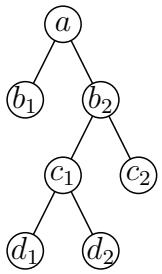


Figure 3.4: One possible transformation from $\mathbb{Z}$ to $\mathbb{Z}^4$

If there are k leaves, this process yields a transformation from $\mathbb{Z}$ to $\mathbb{Z}^k$, where each of the leaves is a component of a vector. In practice, using trees that are more full will provide more uniformity in the size of the disguised weights. It will also make more efficient use of the storage necessary to hold the disguised weights.

The number of rooted binary trees with $n = 1, 2, ..., 10$ internal vertices is given by the sequence $1, 1, 2, 3, 6, 11, 23, 46, 98, 207$ [28]. Thus, for $n = 30$, there are $\approx 10^{10}$ different ways to transform an integer into a vector in $\mathbb{Z}^{31}$. From a cryptanalytic point of view, reversal of this disguising requires knowledge of the parameters used in each modular multiplication as well as the particular tree structure. Without this information, modeling a sequence of modular multiplications would be infeasible. One could not analyze, let alone express, the equations in such a transformation.

Higher Degree Trees

As we have seen previously, the ideas presented regarding quadratic integers can be extended to integers of higher degree. Thus, the modular multiplication trees need not remain binary. We can discuss rooted trees with larger (arbitrary) degree vertices. Obviously, the number of such trees exceeds the number of binary trees. In fact, there is no exact count of trees having a specific number of leaves. However, we can get a rough estimate by counting the number of binary, ternary, or in general p-ary trees. In [2] it was shown that $\frac{1}{(p-1)n+1}\binom{pn}{n}$ counts the number of p-ary trees with n vertices. Since we want a count on the number of leaves we must observe that about half of the total vertices in a binary tree are leaves, about two-thirds of the total vertices of a ternary tree are leaves, and in general about $\frac{p-1}{p}$ of the vertices of a p-ary tree are leaves. So this will provide a rough number of trees with a specific leaf count. Since the number of p-ary trees only accounts for vertices with degree zero or p, only a lower bound for the tree count can be achieved. Even so, for $n = 30$ and $p = 4$, there are more than 10^{26} such tree structures. Again, the number of trees limits a cryptanalyst's ability to enumerate all such options.

Chapter 4

New Codes

In the creation of a new secure cryptographic code, the previous chapter had demonstrated a new method of disguising private weights. This chapter will describe in detail the process of creating the private weights, disguising the private weights, encoding a message, and decoding a message. Two different methods of creating the secret weights will be discussed. One is based on recurrence sequences and the other is based on congruence properties. Both can be used in conjunction with the disguising technique of modular multiplication mentioned previously. In this sense, the creation of the weights and the disguising of the weights can be viewed as independent of each other. For both methods, different processes will be used to convert a message M into a vector $\mathbf{v}_M=(\epsilon_1, \epsilon_2, ..., \epsilon_n)$. Throughout this description, n is the number of public/private weights.

4.1 Code Description

4.1.1 Method 1: Congruence Condition

Setup

Create an increasing sequence $\{r_{ij}\}$ in the following manner. The sequences has n terms grouped into k blocks of size P. The j subscript corresponds to the jth term in the ith block. That is, $\{r_{ij}\} = \{r_{11}, r_{12}, ..., r_{1P}/r_{21}, r_{22}, ..., r_{2P}/.../\ r_{k1}, r_{k2}, ..., r_{kP}\}$. With j fixed, $\{r_{ij}\}$ should increase with a growth rate of approximately 2.

Create the increasing sequence $\{s_{ij}\}$ in a similar manner to $\{r_{ij}\}$, subject to the following two conditions:

- $s_{ij} \equiv r_{ij} (\text{mod } P)$ (4.1.1)
- $s_{h1} > \sum_{i<h} s_{iP}$ for all $h = 2, ..., k$ (4.1.2)

The second condition requires that the smallest term in each group must be larger than the sum of the largest terms of all previous groups. The message, in its various forms, will be represented as sums of r_{ij} or s_{ij}. The sequence $\{s_{ij}\}$ is the set of private weights and $\{r_{ij}\}$ serves as the representation weights for the plaintext message.

Vector Representation

Let $M = M_1$ be the secret message. This message will be converted into a vector $\mathbf{v}_M = (\epsilon_1, \epsilon_2, ..., \epsilon_n) = (\epsilon_{11}, \epsilon_{12}, ..., \epsilon_{1P}/\epsilon_{21}, \epsilon_{22}, ..., \epsilon_{2P}/.../\ \epsilon_{k1}, \epsilon_{k2}, ..., \epsilon_{kP})$ through a process that generates a sequence $M_1, M_2, M_3, ...$ The value of each M_t (mod P) will determine how a "greedy" rule will be used at each state. To begin, set $\mathbf{v}_M = (0, 0, ..., 0)$.

The rule states that if $M_t \equiv j$ (mod P), then use the largest possible term $r^* =$

$r_{ij}, i = 1, ..., k$. Since the value of j corresponds to the j^{th} term in a block, consider j in the set $\{1, 2, ..., P\}$ not the set $\{0, 1, ..., P-1\}$. That is, use the largest possible jth number from one of the blocks. Indicate this usage by incrementing the corresponding ϵ component in $\mathbf{v}_M$. Next, let $M_{t+1} = M_t - r^*$. Continue the process until the smallest r_{ij} grouping has been used. Then save the current value of M_t as C which will be sent in the clear upon transmission of the encoded message. Thus, $\mathbf{v}_M$ is the vector representation of M with respect to $\{r_{ij}\}$ such that $\sum \epsilon_{ij} r_{ij} + C = M$. Or, disregarding the groupings, we can represent M as the sum $\sum_{i=1}^{n} \epsilon_i r_i + C$. Once the message has been represented as a vector, it is encoding by the process described in Section 4.1.4. Then the message is sent.

When the encoded message is transmitted, the receiver reverses the disguising process,as described in Section 4.1.7, to express the message as a sum of the s_{ij} weights. This process is described in Section 4.1.5. Then the receiver must express the message as a sum of the r_{ij} weights. So for a known value of $\sum \epsilon_{ij} s_{ij}$, the ϵ_{ij} need to be recovered. This is done in the same way that M was converted to $\mathbf{v}_M$ above. In this instance, a sequence $T_1, T_2, T_3, ...$ is generated to decode. Thus, $M = \sum \epsilon_{ij} r_{ij} + C$ can be recovered from $T = \sum \epsilon_{ij} s_{ij} + C$.

Example

To illustrate how a message M is converted into a vector $\mathbf{v}_M$, consider the following example. Let $n = 15, P = 3$ and define the sets:

$$\{r_i\} = \{18, 19, 20/46, 47, 48/79, 81, 82/168, 170, 174/339, 342, 344/681, 684, 688\}$$

$$\{s_i\} = \{192, 193, 197/403, 407, 411/784, 789, 793/1458, 1466, 1472/$$
$$3015, 3027, 3077/5991, 6003, 6025\}$$

Notice that $r_i \equiv s_i(\text{ mod } 3)$.

Consider the message $M = M_1 = 900$. Here are the required steps to encode this as the vector $\mathbf{v}_M$.

$$\begin{aligned} M_1 &= 900 \equiv 3(\text{mod } 3) \rightarrow \text{use } 688 \qquad \text{increment } \epsilon_{6,3} \\ M_2 &= 212 \equiv 2(\text{mod } 3) \rightarrow \text{use } 170 \qquad \text{increment } \epsilon_{4,2} \\ M_3 &= 42 \equiv 3(\text{mod } 3) \rightarrow \text{use } 20 \qquad \text{increment } \epsilon_{1,3} \\ C &= 22 \rightarrow \text{send in clear} \end{aligned}$$

So, $900 = 688 + 170 + 20 + C$, where $C = 22$, and

$$\mathbf{v}_M = \{0, 0, 1/0, 0, 0/0, 0, 0/0, 1, 0/0, 0, 0/0, 0, 1\}.$$

Notice that the corresponding s_{ij} weights are 6025, 1466, and 197, which sum to $T = T_1 = 7688$.

For decoding, this process will be reversed to recover the ϵ_i such that $\sum \epsilon_i s_i = 7688 + C$. This is shown below.

$$\begin{aligned} T_1 &= 7688 + 22 \equiv 3(\text{mod } 3) \rightarrow \text{use } 6025 \\ T_2 &= 1663 + 22 \equiv 2(\text{mod } 3) \rightarrow \text{use } 1466 \\ T_3 &= 197 + 22 \equiv 3(\text{mod } 3) \rightarrow \text{use } 197 \\ C &= 22 \end{aligned}$$

Once the ϵ_i have been recovered, the original message M can now be computed as $\sum \epsilon_i r_i + C = 878 + 22 = 900$.

Unique Decoding

It must be verified that the use of this greedy rule to decode will always produce the same $\mathbf{v}_M$ as was encoded. Using the r_{ij} weights, this construction yields a standard $\{0,1\}$ knapsack with some additional constraints added. By construction, the greedy algorithm always chooses the largest possible weight such that the following conditions are satisfied: (1) at most one weight is chosen from each grouping and (2) each weight is selected from the appropriate position within a group (determined by a congruence). A weight that satisfies (2) is said to be *allowable*.

Within this context, consider the following argument. If it is determined that $T = s_{i_1 j_1} + s_{i_2 j_2} + ... + s_{i_q j_q}$ by the decoding algorithm, then this same algorithm should yield $M = r_{i_1 j_1} + r_{i_2 j_2} + ... + r_{i_q j_q}$with the same representation for uniqueness to hold. Suppose, at some point in the process there is agreement in representing T_k such that

$$M_k = r_{i_1 j_1} + r_{i_2 j_2} + ...$$
$$T_k = s_{1_1 j_1} + s_{i_2 j_2} + ...$$

Assume $r_{i_k j_k}$ is used at this stage in representing M_k. Then for uniqueness, the weight $s_{i_k j_k}$ must be used to represent T_k. Since $r_{i_k j_k}$ was chosen, it is allowable and the next larger allowable $r_{(i_k+1) j_k}$ is too large, i.e. larger than M_k. Hence $s_{i_k j_k}$ is also allowable by (4.1.1). Because $s_{i_k j_k} \leq T_k$, it will be chosen provided that the next allowable weight $s_{(i_k+1) j_k}$ is greater than T_k. This is verified by (4.1.2) since

$$s_{(i_k+1) j_k} \geq s_{(i_k+1)1} > \sum_{i \leq i_k} s_{iP} \geq s_{1_1 j_1} + s_{i_2 j_2} + ... = T_k.$$

Therefore, the greedy algorithm, along with conditions (1) and (2), will always yield a unique decoding.

4.1.2 Method 2: Recurrence Relations

Background

Let $\{r_i\}$ be a d^{th} order recurrence sequence satisfying:

$$r_n = a_1 r_{n-1} + a_2 r_{n-2} + ... + a_d r_{n-d} \tag{4.1.3}$$

In [13], is was shown that there is a unique representation for any positive integer N, given by

$$N = \sum_{i=0}^{k} \epsilon_i r_i \tag{4.1.4}$$

provided that

$$a_1 \geq a_2 \geq ... \geq a_d \geq 1. \tag{4.1.5}$$

Here, the ϵ_i digits must satisfy $\epsilon_j ... \epsilon_{j-k} \leq a_1 ... a_d$ in the lexographical order for all $j \geq 0$ and $1 \leq k \leq d$.

A common example of this type of representation is the well known Zeckendorf representation using the Fibonacci numbers [39]. This process allows for unique representation along with a high density sequence (large ϵ_i), which is a beneficial step for defeating Basis Reduction. Furthermore, as shown recently in [15], the restriction given in (4.1.5) can be eliminated, thus increasing the density, while preserving uniqueness. If $a_1 + a_2 + ... + a_d = A$, define a set of sequences $\mathsf{S} = \{S_0, S_1, ..., S_{A-1}\}$ as follows. Let $S_0 = 0$ and S_i for $i \geq 1$ consist of all strings of length l, for each l, such that $a_1 ... a_{l-1} \leq S_i < a_1 ... a_l$. For example, if $a_1 a_2 a_3 a_4 = 2013$, then S $= \{0, 1, 200, 2010, 2011, 2012\}$.

The main result of [15] states that if $\{r_i\}$ is a sequence satisfying (4.1.3), then any positive integer N can be uniquely represented as in (4.1.4) where the string of digits $\epsilon_k\epsilon_{k-1}...\epsilon_0$ is composed of blocks of digits in the set S. Thus, a representation is unique if $\epsilon_k\epsilon_{k-1}...\epsilon_0$ can be parsed into blocks in S reading left to right. This provides the justification for the code construction that now follows.

Vector Representation

Select nonnegative integers d and $a_1, ..., a_d$ to create a recurrence sequence $\{r_i\}$ that satisfies (4.1.3). From the above comments, it is trivial to determine if a representation is allowable, but how can the appropriate ϵ_i values be determined for a given message M? Before we can answer this question, an equivalent formulation must be considered. Let the sequence $\{w_i\}$ consist of all numbers of the form $b_1r_n + b_2r_{n-1} + ... + b_jr_{n-j+1}$ arranged in increasing order. The b_i are chosen such that:

$$\begin{aligned} 1 \le b_1 \le a_1 \quad &\text{if} \quad j = 1 \\ a_1...a_{j-1} \le b_1...b_j \le a_1...a_j \quad &\text{if} \quad 1 \le j \le d \\ a_1...a_{d-1} \le b_1...b_d \le a_1...(a_d - 1) \quad &\text{if} \quad j = d. \end{aligned} \tag{4.1.6}$$

The w_i consist of the all the allowable representations of sums of r_i. They are directly related to the set S of sequences above. Now the greedy algorithm can be used to express M as

$$M = \sum_{i=0}^{k} \epsilon_i' w_i.$$

where $\epsilon_i' \in \{0, 1\}$. From this, M can also be represented as (4.1.4) with $\epsilon_i \ge 0$.

Example

Consider the recurrence sequence $\{r_i\} = \{1, 2, 4, 9, 23, 56, 133, 316...\}$ that satisfies

$$r_n = 2r_{n-1} + r_{n-3} + 3r_{n-4}.$$

Then by (4.1.6), the allowable sums of r_n are $r_n, 2r_n, 2r_n + r_{n-2}, 2r_n + r_{n-2} + r_{n-3}$, and $2r_n + r_{n-2} + 2r_{n-3}$. Therefore,

$$\begin{aligned}\{w_i\} &= \{1; 2; 4, 8; 9, 18, 20, 21, 21; 23, 46, 50, 52, 54; 56, 112, 121, 125, 129; \\ &\quad 133, 266, 289, 298, 307; 316, 632...\}\end{aligned}$$

Consider the message $M = 600$. The greedy algorithm applied to $\{w_i\}$ yields $600 = 316 + 266 + 18$. Hence, $\epsilon_i' = 1$ only for $i = 5, 20, 24$ otherwise $\epsilon_i' = 0$. Notice also that $\epsilon_3 = 2, \epsilon_6 = 2$, and $\epsilon_7 = 1$, since $600 = w_5 + w_{20} + w_{24} = r_7 + 2r_6 + 2r_3$. Thus, M has been converted into the vector $\mathbf{v}_M$=(ϵ_i).

Unique Decoding

As shown in [15], the representation of an integer must be unique using this recurrence method. Thus, the decoding process will always yield one, and only one, correct message.

4.1.3 Disguising

Convert each $\{s_i\}$ weight into an integer vector with m components, $T_i = \{t_{ji}\}, 1 \leq j \leq m, 1 \leq i \leq n$ through modular multiplications in algebraic number rings. Let the $m \times n$ matrix T consist of the columns T_i. For each of the m rows of T, choose a set of primes $\{p_{jk}\}_{k=1}^{g_j}$ such that $\prod_{k=1}^{g_j} p_{jk} > \sum_{i=1}^{n} t_{ji}$. Technically, the numbers need only

be relatively prime to each other for the Chinese Remainder Theorem, but choosing primes will be simple and sufficient. The number of primes, g_j, chosen for each row may vary. Then, for each row compute the g_j sets

$$\{t_{ji}(\text{mod } p_{jk})\} \text{ for } k = 1, ..., g_j.$$

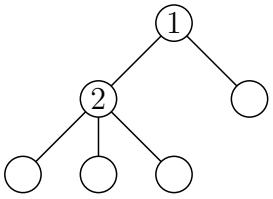
Let $G = \sum g_j$ be the total number of primes used for all rows combined. These sets, in order, form the rows $\mathbf{w}_i$ of a $G \times n$ matrix W. The rows of W are then randomly permuted to form the matrix W' which is the public set of weights. The weights $\{s_i\}$ are kept private.

To summarize, here are the matrices produced in the process of disguising:

$$\{s_i\} \Longrightarrow T \Longrightarrow W \Longrightarrow W' \tag{4.1.7}$$

A Small Disguising Example

For the set $\{s_i\} = \{3, 8, 17, 36, 79\}$ use the following modular multiplication tree



with these given parameters:

1. multiplier: $19 + 9\sqrt{2}$, modulus: $23 + 13\sqrt{2}$

2. multiplier: $5 + 9(3)^{1/3} + 6(3)^{2/3}$, modulus: $7 + 8(3)^{1/3} + 4(3)^{2/3}$.

Notice that the norm of the moduli (191 and 439 respectively) are prime and larger than the sum of the weights to ensure unique decoding, as in the standard modular

multiplication of Merkle-Hellman. The quadratic transformation (1) produces the weights

$$\begin{pmatrix} 14 & 20 & 13 & 18 & 34 \\ 11 & 14 & 11 & 14 & 23 \end{pmatrix}. \tag{4.1.8}$$

Each column relates to its corresponding s_i value. So, for example,

$$3(19 + 9\sqrt{2}) \equiv 14 + 11\sqrt{2} (\text{mod } 23 + 13\sqrt{2})$$

and so on. Then the cubic transformation acts on the first row of (4.1.8) to produce the weights

$$\begin{pmatrix} 28 & 7 & 30 & 30 & 23 \\ 19 & 4 & 18 & 17 & 16 \\ 14 & 4 & 12 & 12 & 10 \end{pmatrix}. \tag{4.1.9}$$

Now combine the second row of (4.1.8) with (4.1.9) to create the matrix

$$T = \begin{pmatrix} 28 & 7 & 30 & 30 & 23 \\ 19 & 4 & 18 & 17 & 16 \\ 14 & 4 & 12 & 12 & 10 \\ 11 & 14 & 11 & 14 & 23 \end{pmatrix}$$

Select the following primes to use:

$$\{p_{1k}\} = \{2, 3, 5, 7\}, \quad \{p_{2k}\} = \{7, 13\}, \quad \{p_{3k}\} = \{3, 5, 7\}, \quad \{p_{4k}\} = \{2, 5, 11\}$$

Each one of the rows of T then breaks down into multiple rows which comprise the following matrix:

$$W = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 3 & 2 & 3 & 3 & 4 \\ 0 & 0 & 6 & 4 & 6 \\ 5 & 4 & 4 & 3 & 2 \\ 6 & 4 & 5 & 4 & 3 \\ 2 & 1 & 0 & 0 & 1 \\ 4 & 4 & 2 & 2 & 0 \\ 0 & 4 & 5 & 5 & 3 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 4 & 1 & 4 & 3 \\ 0 & 3 & 0 & 3 & 1 \end{pmatrix}$$

A secret permutation is then applied to the rows of W to yield the public weights W'.

4.1.4 Encoding

Once M has been completely transformed into $\mathbf{v}_M$, compute $\mathbf{v}'_M = W'\mathbf{v}_M$. The vector $\mathbf{v}'_M$ is the encoded message that can be sent to the receiver. The corresponding knapsack problem is given by the system $W'\mathbf{x}=\mathbf{v}'_M$.

Two quick observations will be helpful to refer to for the reversal of the disguising.

By construction, the dot product of $\mathbf{v}_M$ and $\{r_i\}$ is M. Consider the dot product of $\mathbf{v}_M$ and $\{s_i\}$. We will call this value K. That is,

$$\sum_{i=1}^{n} \epsilon_i s_i = K. \tag{4.1.10}$$

Second, the vector $\mathbf{z} = \mathrm{T}\mathbf{v}_M$ has components $(z_1, z_2, ..., z_m)$ that satisfy the following relations:

$$\begin{aligned} \sum_{i=1}^{n} \epsilon_i t_{1i} &= z_1 \\ &\vdots \\ \sum_{i=1}^{n} \epsilon_i t_{mi} &= z_m \end{aligned} \tag{4.1.11}$$

4.1.5 Disguising Reversal

The receiver has knowledge of the secret permutation and thus may apply its inverse to W' to yield the matrix W. In the original order, one knows that the first g_1 rows of W correspond to the first row of T, the next g_2 rows of W correspond to the second row of T, and so on. Upon receiving the encoded message vector $\mathbf{v}'_M$=$(z'_1, z'_2, ..., z'_G)$, the Chinese Remainder Theorem is used to compute

$$z_1 \equiv \begin{cases} z'_1 & (\mathrm{mod}\ p_{11}) \\ \vdots & \\ z'_{g_1} & (\mathrm{mod}\ p_{1g_1}) \end{cases}$$

$$z_2 \equiv \begin{cases} z'_{g_1+1} & (\mathrm{mod}\ p_{21}) \\ \vdots & \\ z'_{g_1+g_2+1} & (\mathrm{mod}\ p_{2g_2}) \end{cases}$$

and so on.

Now these $z_1, ..., z_m$ are related to the ϵ_i as shown in (4.1.11).

The goal now is to express a sum involving $\{t_{ji}\}$ as a sum involving $\{s_i\}$. All of the parameters of moduli and multipliers in the disguising are known to the receiver and reversible as shown in Section 3.1. One must apply the inverse of the modular transformations to the m equations until one single equation remains. This will yield

a sum involving the known $\{s_i\}$ weights. Hence, we can determine K such that satisfies (4.1.10).

Small Disguising Example - Continued

First, the inverses of the multipliers used in the modular multiplications are below as computed from techniques in Chapter 3:

1. inverse multiplier: 177, modulus: $23 + 13\sqrt{2}$, norm = 191.
2. inverse multiplier: 242, modulus: $7 + 8(3)^{1/3} + 4(3)^{2/3}$, norm = 439.

By (4.1.11), the first three rows of T can be expressed as the single equation

$$\begin{aligned}(28 + 19(3)^{1/3} + 14(3)^{2/3})\epsilon_1 \quad &+ \quad (7 + 4(3)^{1/3} + 4(3)^{2/3})\epsilon_2 + \\ (30 + 18(3)^{1/3} + 12(3)^{2/3})\epsilon_3 \quad &+ \quad (30 + 17(3)^{1/3} + 12(3)^{2/3})\epsilon_4 + \\ (23 + 16(3)^{1/3} + 10(3)^{2/3})\epsilon_5 \quad &= \quad z_1 + z_2(3)^{1/3} + z_3(3)^{2/3}\end{aligned}$$

Multiplying this equation by 242 and reducing in $\mathbb{Z}$ mod 439 yields the equation

$$14\epsilon_1 + 20\epsilon_2 + 13\epsilon_3 + 18\epsilon_4 + 34\epsilon_5 = z^*$$

where z^* is a known value. This equation, along with the equation determined by the last row of T, can now be viewed as

$$\begin{aligned}(14 + 11\sqrt{2})\epsilon_1 + (20 + 14\sqrt{2})\epsilon_2 + (13 + 11\sqrt{2})\epsilon_3 \quad &+ \quad (18 + 14\sqrt{2})\epsilon_4 + \\ (34 + 23\sqrt{2})\epsilon_5 \quad &= \quad z^* + z_4\sqrt{2}\end{aligned}$$

Multiplying this equation by 177 and reducing in $\mathbb{Z}$ mod 191 yields the equation

$$3\epsilon_1 + 8\epsilon_2 + 17\epsilon_3 + 36\epsilon_4 + 79\epsilon_5 = z^{**}$$

where z^{**} is a known value. In this manner, a sum involving $\{t_i\}$ can now be expressed as a sum involving $\{s_i\}$.

4.1.6 Summary

To summarize the previous steps consider the following short-hand version of the process. It provides the four major steps for the implementation of the system.

Create Weights $\{r_i\}$ and $\{s_i\}, 1 \leq i \leq n$

Disguise Weights $\{s_i\} \Longrightarrow T \in \mathbf{M}_{m\times n}(\mathbb{Z}) \Longrightarrow W \in \mathbf{M}_{G\times n}(\mathbb{Z}) \Longrightarrow W' \in \mathbf{M}_{G\times n}(\mathbb{Z})$

Encode Message
1. Express message $M = \sum \epsilon_i r_i$ as $\sum \epsilon_i s_i$ where $\mathbf{v}_M{=}(\epsilon_i) \in \mathbf{Z}^n$
2. Compute secret message $\mathbf{v}'_M = W'\mathbf{v}_M \in \mathbf{Z}^G$

Decode Message
1. Reverse the disguising to express $\mathbf{v}'_M$ as $\sum \epsilon_i s_i = K(\text{known})$
2. Determine ϵ_i through congruence condition to compute $M = \sum \epsilon_i r_i$

Chapter 5

Code Cryptanalysis

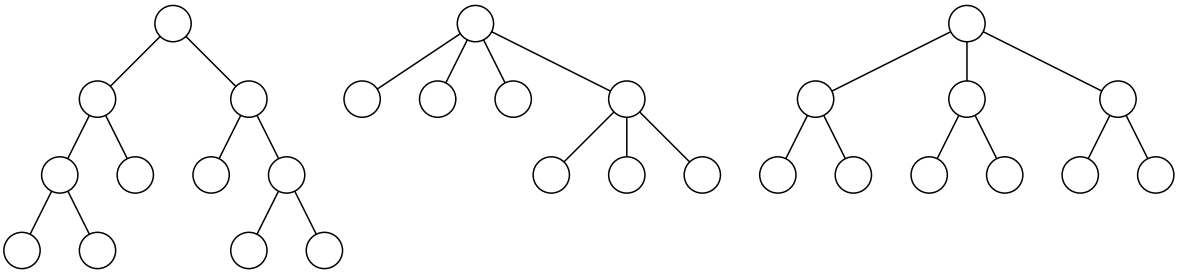
As mentioned in Chapter 1, the two primary method of attack on knapsack-type codes are Diophantine Approximation and Basis Reduction. Therefore, this new system must, at a minimum, withstand the techniques of these attacks. In this chapter the security of this system will be demonstrated through its ability to resist these cryptanalytic techniques.

5.1 Diophantine Approximation

The purpose of this attack is to reverse the disguising process and reveal the secret weights $\{s_i\}$. The initiation of this method requires an equation relating the secret weights to the public weights through the modular multiplication process, see (1.4.1). For this code, the secret weights are integers and the public weights are integer vectors in $\mathbb{Z}^n$. This necessitates that a system of at least n equations be used to relate the weights. As will be shown, there are too many possible ways to create the public vector for the cryptanalyst to analyze.

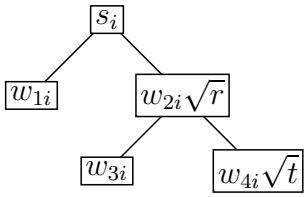
In addition, there is great deal of flexibility on the part of the code maker for which the cryptanalyst will be unable to account. As described in Section 3.4.1, a

sequence of modular multiplications (of varying degrees) can be performed on the weights. A specific representation of this sequence is given by the tree structure, which is unknown to the cryptanalyst. Therefore, the correct systems of equations cannot be expressed. For example, if the public weights are of degree 6, it is not known whether they came from any of the following tree structures, or possibly another.



Each of these trees has 6 leaves and thus provides a valid transformation from $\mathbb{Z}$ to $\mathbb{Z}^6$.

Suppose a modular multiplication transformation on $\{s_i\}$ yields vectors in $\mathbb{Z}^3$. First, assume this was done via two quadratic extensions as shown below. The two multiplier-modulus pairs are given by $(k_1, a + b\sqrt{r})$ and $(k_2, c + d\sqrt{t})$ in that order. The x_{1i}, x_{2i}, y_{1i}, and y_{2i} terms given below are integers.



Then,

$$\begin{aligned} s_i &= k_1(w_{1i} + w_{2i}\sqrt{r}) + (a + b\sqrt{r})(x_{1i} + x_{2i}\sqrt{r}) \\ &= (k_1 w_{1i} + a x_{1i} + b x_{2i} r) + (k_1 w_{2i} + a x_{2i} + b x_{1i} r)\sqrt{r} \end{aligned}$$

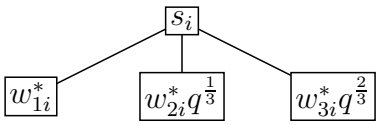
and

$$\begin{aligned} w_{2i} &= k_2(w_{3i} + w_{4i}\sqrt{t}) + (c + d\sqrt{t})(y_{1i} + y_{2i}\sqrt{t}) \\ &= (k_2 w_{3i} + c y_{1i} + d y_{2i} t) + (k_2 w_{4i} + c y_{2i} + d y_{1i} r)\sqrt{t} \end{aligned}$$

Together, these produce the equations:

$$\begin{aligned} k_1 w_{1i} + a x_{1i} + b x_{2i} r &= s_i \\ k_1 w_{2i} + a x_{2i} + b x_{1i} r &= 0 \\ k_2 w_{3i} + c y_{1i} + d y_{2i} t &= w_{1i} \\ k_2 w_{4i} + c y_{2i} + d y_{1i} r &= 0 \end{aligned} \tag{5.1.1}$$

Next, assume that $\{s_i\}$ is disguised by one modular multiplication in a cubic ring as shown below with the single multiplier-modulus pair $(k, f + gq^{\frac{1}{3}} + hq^{\frac{2}{3}})$. The z_{1i}, z_{1i}, and z_{1i} terms given below are integers.



Then,

$$s_i = k(w_{1i}^* + w_{2i}^* q^{\frac{1}{3}} + w_{3i}^* q^{\frac{2}{3}}) + (f + gq^{\frac{1}{3}} + hq^{\frac{2}{3}})(z_{1i} + z_{2i}q^{\frac{1}{3}} + z_{3i}q^{\frac{2}{3}}) \qquad (5.1.2)$$

which yields the equations

$$\begin{aligned} kw_{1i}^* + fz_{1i} + gx_{3i}q + hz_{2i}q &= s_i \\ kw_{2i}^* + fz_{2i} + gx_{1i}q + hz_{3i}q &= 0 \\ kw_{3i}^* + fz_{3i} + gx_{2i}q + hz_{1i} &= 0 \end{aligned} \qquad (5.1.3)$$

In both cases, the only public information are triples of integers (w_{1i}, w_{3i}, w_{4i}) or $(w_{1i}^*, w_{2i}^*, w_{3i}^*)$ and the analyst has no idea which system of equations to model, (5.1.1) or (5.1.3).

The two major pieces of information the cryptanalyst needs to know are (1) the value of each of the parameters and (2) the structure of the tree used in disguising. The second produces the "template" or form of the system of equations and the first provides the coefficients to solve the system. Neither of these are public. What complicates this even more is that, in certain instances, knowledge of (2) is not sufficient to determine the form of the system.

Another problem the cryptanalyst encounters is not knowing what "type" of algebraic numbers are being used. For example, assume that one modular multiplication took place to yield a vector in $\mathbb{Z}^4$. This could have been produced either by a biquadratic extension or another type of fourth degree extension, but there is no way of knowing which was used. Furthermore, each type will give rise to different systems of equations. Consider the following example.

Convert $\{s_i\} \in \mathbb{Z}$ into $\{(a_i, b_i, c_i, d_i)\} \in \mathbb{Z}^4$ with a multiplier of k and a modulus of $\gamma = (m_1, m_2, m_3, m_4)$.

First assume that γ is the bi-quadratic integer $m_1 + m_2\sqrt{r} + m_3\sqrt{t} + m_4\sqrt{rt}$. Then for each s_i, we have

$$\begin{pmatrix} s_i \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a_i & rb_i & tc_i & rtd_i \\ b_i & a_i & td_i & tc_i \\ c_i & rd_i & a_i & rb_i \\ d_i & c_i & b_i & a_i \end{pmatrix} \begin{pmatrix} k \\ 0 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} m_1 & rb_2 & tc_3 & rtm_4 \\ m_2 & m_1 & tm_4 & tm_3 \\ m_3 & rm_4 & m_1 & rm_2 \\ m_4 & m_3 & m_2 & m_1 \end{pmatrix} \begin{pmatrix} w_i \\ x_i \\ y_i \\ z_i \end{pmatrix}$$

This produces the system of equations:

$$\begin{aligned} a_i k &= m_1 w_i + r m_2 x_i + t m_3 y_i + rt m_4 z_i + s_i \\ b_i k &= m_2 w_i + m_1 x_i + t m_4 y_i + t m_3 z_i \\ c_i k &= m_3 w_i + r m_4 x_i + m_1 y_i + r m_2 z_i \\ d_i k &= m_4 w_i + m_3 x_i + m_2 y_i + m_1 z_i \end{aligned} \tag{5.1.4}$$

where the known values are a_i, b_i, c_i, and d_i, the fixed unknowns are r, t, k, m_1, m_2, m_3, and m_4 and the unknowns for each s_i are w_i, x_i, y_i, and z_i.

Next assume that γ is the algebraic integer $m_1 + m_2 r^{1/4} + m_3 r^{2/4} + m_4 r^{3/4}$. Then for each s_i, we have

$$\begin{pmatrix} s_i \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a_i & rd_i & rc_i & rb_i \\ b_i & a_i & rd_i & rc_i \\ c_i & b_i & a_i & rd_i \\ d_i & c_i & b_i & a_i \end{pmatrix} \begin{pmatrix} k \\ 0 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} m_1 & rm_4 & rm_3 & rm_2 \\ m_2 & m_1 & rm_4 & rm_3 \\ m_3 & m_2 & m_1 & rm_4 \\ m_4 & m_3 & m_2 & m_1 \end{pmatrix} \begin{pmatrix} w_i \\ x_i \\ y_i \\ z_i \end{pmatrix}$$

This produces the system of equations:

$$\begin{aligned} a_i k &= m_1 w_i + r m_4 x_i + r m_3 y_i + r m_2 z_i + s_i \\ b_i k &= m_2 w_i + m_1 x_i + r m_4 y_i + r m_3 z_i \\ c_i k &= m_3 w_i + m_2 x_i + m_1 y_i + r m_4 z_i \\ d_i k &= m_4 w_i + m_3 x_i + m_2 y_i + m_1 z_i \end{aligned} \tag{5.1.5}$$

where the known values are a_i, b_i, c_i, and d_i, the fixed unknowns are r, k, m_1, m_2, m_3, and m_4 and the unknowns for each s_i are w_i, x_i, y_i, and z_i.

Notice that in each case, a completely different system of equations is generated, both of which are non-linear. Yet, the trees associated with both transformations are identical and given below.

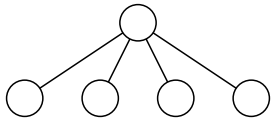


Figure 5.1: The tree associated with a single transformation from $\mathbb{Z}$ to $\mathbb{Z}^4$

This system only represents one single modular multiplication transformation. In a full scale implementation, the disguising process would consist of multiple layers of transformations in an unknown order. This introduces the complexity of adding an unknown number of unknown parameters, effectively eliminating the possibly of expressing a system of equations to represent the disguising process.

Recall that the Diophantine Approximation attack described in Section 1.4 reversed the modular multiplication disguising by determining a ratio of integers sufficiently close to the ratio of multiplier to modulus. To replicate that procedure in

this setting would require division of algebraic integers to be meaningful. It is easy to express an algebraic integer in its decimal form, but it is impossible to take a decimal representation and express it as an algebraic integer with unknown parameter values (i.e. r and/or s). So finding ratios of algebraic integers becomes an infeasible task. Thus, the techniques of Diophantine Approximation used with the standard knapsack in $\mathbb{Z}$ do not apply here.

It is true that a_i, b_i, c_i, and d_i are known values, but because they are permuted in the disguising process, the cryptanalyst is not given them in the correct order. If, for example, a_i and b_i are switched, the equations will solve to and incorrect solution. The four weights must be selected in the correct order. Obviously, there are $4! = 24$ different orderings from which to choose. In general, with k sets of weights, there are $k!$ possible orderings. Furthermore, only recovering the correct ordering will yield a solution capable of reversing the disguising.

Though there is a very small likelihood of this occurring, even such a feat would be insufficient to break the code. Recall that in the previous example, the cryptanalyst would not be able to determine whether to use (5.1.4) or (5.1.5) to invert the disguising. Thus, the same tree in Figure 5.1 gives rise to two different systems. In addition, there are many other systems that could produce vectors with four components. Each one is associated with a rooted tree with four leaves. In total there are 11 different trees with 4 leaves (one of which yields 2 systems). For the cryptanalyst to assure the correct ordering of the weights and the correct equations (tree), he must consider at least $24 \cdot 11 \cdot 2 = 528$ distinct possibilities. Assuming all the equations can be solved, only one out of the 528 will yield the correct reversal of the disguise.

The assumption is that even with the correct permutation and "form" of the

equations, the cryptanalyst will be unable to actually solve the non-linear equations. Yet, to prove such a statement would be extremely difficult in all situations. Hence, the other arguments presented in this section provided the basis for security with this type of disguising.

5.1.1 Specialized Attack

In addition to the previous discussion, we also show that the public weights (columns of W') cannot be utilized to recover the rows of T. Recall that each row of T is broken down modulo various primes to form the rows of W. The columns of W are then permuted to form W'. Thus, one does not know which, or how many, of the rows must be recombined with the Chinese remainder theorem. The approximate size of the primes can be deduced from the sizes of the row entries. In this sense, the primes are not assumed to be completely hidden from the cryptanalyst.

For the sake of a simplified example, assume the prime values are known and that each column of T is a vector in $\mathbb{Z}^{15}$. If 5 primes are used for each row of T, there would be

$$\binom{75}{5}\binom{70}{5}\cdots\binom{5}{5} \approx 10^{78}$$

choices to recombine the primes with the Chinese Remainder Theorem. Each group of 5 rows must be selected in precisely the right way. If any grouping is taken incorrectly, the resulting combination will be incorrect. Of course any number of primes can be used to break down each weight. This simply demonstrates that even for a basic case it is infeasible to begin reversing the disguising of the public weights without knowledge of the secret permutation.

5.2 Basis Reduction

In contrast with the Diophantine Approximation attack, the Basis Reduction (BR) approach has no regard for the various methods and techniques of disguising. Instead, the knapsack problem of finding $\epsilon_i \in \{0,1\}$ such that $\sum_{i=1}^{n} \epsilon_i w_i$ equals some target is interpreted as the shortest vector problem in a lattice. In the proposed cryptosystem, we have introduced a knapsack problem that involves a system of equations, rather than just a single equation. Hence, a modified procedure (see [17]) that is different from that of Section 1.2 is required. Notice that a solution to the $G \times n$ system $W'\mathbf{x}=\mathbf{v}'_M$ (introduced in Section 4.1.3) is also a solution to the $(G+n) \times (n+1)$ system

$$\begin{pmatrix} W' & -\mathbf{v}'_M \\ I_n & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{x} \end{pmatrix}$$

where $\mathbf{0}$ is the column vector $(0,0,...,0) \in \mathbf{Z}^G$. Thus, the solution vector will be an integer linear combination of the columns of

$$\begin{pmatrix} W' & -\mathbf{v}'_M \\ I_n & 0 \end{pmatrix} \tag{5.2.1}$$

that has zeros in the first G coordinates. The set of all vectors that are integer linear combinations of the columns of (5.2.1) form a lattice. Since these columns are linearly independent, they form a basis for this lattice. Therefore, applying a BR algorithm to (5.2.1) will yield a much shorter basis, that is, vectors with much smaller norms. Since, the solution vector $\mathbf{x}$ is short, it generally appears in the reduced basis. In this way, the solution to the system $W'\mathbf{x}=\mathbf{v}'_M$ can be recovered.

There are four possible outcomes when BR is run on this system:

1. The correct solution vector $\mathbf{v}_M$ is found

2. A false solution vector is found such that $W'\mathbf{x}=\mathbf{v}'_M$

3. A short vector is found such that $W'\mathbf{x}\neq\mathbf{v}'_M$

4. BR fails to complete the reduction

If outcome 1 occurs, the code is broken because the $\epsilon'_i s$ have been recovered and $M = \sum_{i=1}^{n} \epsilon_i r_i$ can be computed. Outcomes 2, 3 and 4 yield no information on the solution vector. It will be shown that the first outcome is insignificantly likely to occur, rendering the system secure against the BR attack.

The cryptanalytic success of BR lies in the fact that the solution vector is a "short" basis vector. Thus, it appears in the reduced basis. If, however, there exist many vectors in the basis that are shorter than the solution vector, BR will not find the solution. In this way, BR acts as a black box that seeks only to find the shortest basis vector. Based on our construction, we can guarantee not only that the solution vector is not the shortest, but that there are a significant number of shorter vectors in the basis. This will make up the argument that BR is unsuccessful against this cryptosystem.

This analysis will be restricted to the method of construction using the congruence conditions. The recurrence relation method requires more advanced modeling of BR constraints. The argument will be similar to the one given below, but omitted from this paper.

For a given value of n, P, and k, the expected length of an encoded message $\mathbf{v}_M$, that is $\sum \epsilon_i$, can be found experimentally. Call is ℓ. This counts the number of the k groups from which a weight is selected during encoding. Experimentally, $\ell = \frac{k}{2}$. Let

the vectors w_i be the columns of W'. Then the cryptanalyst tries to solve:

$$\sum x_i w_i = \mathbf{v}'_M$$

where $\mathbf{v}'_M$ is the public target. The correct solution is $(x_i) = (\epsilon_i)$.

The size of the public weights must now be considered in order to determine BR's effectiveness. Let $A =$ the number of possible public targets and let $B =$ the number of vectors $(x_1, ..., x_n)$ of length $< \ell$ that satisfy construction constraints that can be modeled by BR. As will be shown, A is determined from the size of the private weights and the amount of disguising, whereas the calculation of B requires knowledge of the specific method of constructing the private weights. For example, in the congruence construction, BR can model the constraint of at most one weight being selected from each group. However, it cannot account for a specific congruence condition being satisfied. Thus, the value of B can be computed as $P^\ell \binom{k}{\ell}$ where P^ℓ counts the number of possible weights selected from ℓ groups and $\binom{k}{\ell}$ counts the number of ways to select the ℓ groups. Recall, k is the number of blocks or groups that partition all n weights. A few values of B are given in Table 5.1.

The size of A is roughly the size of the public weights (vectors). That is, each message is encoded by adding up a particular subset of these weights. Therefore, there are two considerations necessary for the calculation of A: the size of the undisguised private weights, and the amount of increase in the size of the weights due to disguising. In this particular construction, the weights roughly double for each of the k groups. Therefore the private weights are approximately 2^k.

Due to the significant number of parameter choices in the disguising, the nature of this analysis does not allow for very sharp bounds. This argument will consist of rough estimates rather than precise sizes of weights. However, some general comments

can be made as to the amount of increase in size due to disguising.

The process of disguising consists of two distinct parts: modular multiplications and the breakdown modulo various primes. To invert each modular multiplication $(\text{mod } \alpha)$ requires that $N(\alpha) > \sum \epsilon_i w_i$. Thus, it suffices to have $N(\alpha) > max(w_i) \sum \epsilon_i$.

Here, w_i stands for any generic weights being disguised. On average, $N(\alpha)$ can be estimated as $\ell \cdot max(w_i)$. To give a more general bound assume that each $\epsilon_i = 1$. Then, $N(\alpha) > k \cdot max(w_i)$. Similarly, each prime breakdown of the disguised weights using moduli $\{p_1, ..., p_t\}$ requires that $p_1 \cdots p_t > k \cdot max(w_i)$.

In both cases, each step in disguising increases the weights by a factor of k.

Table 5.1: n=2,000 weights

P	k	B	A with disguising of Fig. 5.2	$\frac{B}{A}$
4	500	$\approx 10^{300}$	$\approx 10^{180}$	$\approx 10^{120}$
5	400	$\approx 10^{259}$	$\approx 10^{150}$	$\approx 10^{109}$
8	250	$\approx 10^{187}$	$\approx 10^{102}$	$\approx 10^{85}$
10	200	$\approx 10^{160}$	$\approx 10^{86}$	$\approx 10^{74}$
20	100	$\approx 10^{95}$	$\approx 10^{53}$	$\approx 10^{42}$

As described previously, the disguising procedure may be represented as a tree where each internal vertex corresponds to a modular multiplication and each leaf is split apart modulo a set of primes. Then each vertex increases the size of the weights by a factor of k. Thus, if the tree has d vertices, the total increase in size due to disguising is k^d.

A set of private weights with parameters $P = 10$ and $k = 200$ will be roughly of size $2^{200} \approx 10^{60}$. Consider the small disguising tree given in Figure 5.2.

This disguising will increase the private weights to a size of $2^{200} \cdot 200^{11} \approx 10^{85}$ since the tree has 11 vertices. This is the value of A, the size of the public target

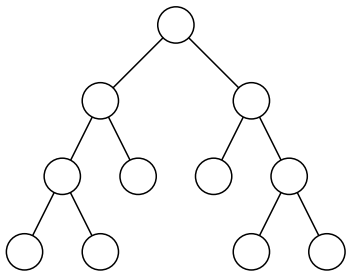


Figure 5.2: Five Modular Multiplications

vector. This is provided, along with two other parameter choices in Table 5.1.

A gauge of BR's effectiveness can be determined by the ratio $\frac{B}{A}$. The larger this value, the less likely BR is to find the correct solution vector. This can be seen through the following argument.

Each vector of B is matched with one target of A. Since the public weights are essentially random (in the sense that they lack any particular structure), it can be assumed that they are uniformly matched with targets in the target space. Thus, if B is sufficiently larger than A, then each target should expect to match up with many vectors. For example, from the last column of Table 5.1, *each* target would expect anywhere from 10^{42} to 10^{120} vectors that "map" to it.

Of course, the solution to the knapsack problem involving the public weights is one unique vector that"maps" to the public target. Hence, to be successful BR would be required to find one out of all of these vectors. It would be difficult for BR to make the number of possibilities smaller due to its inability to model the congruence constraints.

There is a natural trade-off between providing security against both BR and specialized attacks. There is a limit to the amount of disguising that should be applied

to the private weights since every added disguise increases A, and hence decreases $\frac{B}{A}$. Yet sufficient disguising is warranted to ensure the reversal of the disguising remains an intractable problem. The balance between these two concerns should yield a suitable range of parameter values for practical use.

Chapter 6

Concluding Remarks

Cryptographic systems making use of the knapsack problem have had their highs and lows over the past 30 years. While many in the mathematical community have all but given up on the practicality of a secure code, the concepts and methods brought forth in this paper give hope for the future. The generalization of modular multiplication beyond the integers opens many door for further research. This new linear and reversible method of disguising weights brings with it numerous parameter choices, all of which potentially play a role in the effectiveness of the disguise. Thus, the search for an optimal selection of parameter values may yield interesting results.

Due to the versatility of this new modular multiplication, it is compatible with any sequence of private weights. So it provides a framework for disguising knapsack weights beyond just the two given in this paper. In this sense, the search for a secure knapsack code is reduced to finding a method of constructing weights that is only immune to Basis Reduction attacks.

The two major successful attacks on knapsack codes are methods based on Basis Reduction and Diophantine Approximation. Yet many open questions remain. What other methods of attack could be effective against this type of cryptosystem?

Are there any inherent weaknesses in the construction of the code that could be exploited? Could advances in computational efficiencies allow for any secrete data to be uncovered? While these questions pertain to the particular code described in this paper, they are not unique to this code. These questions are asked of all purposed cryptosystems. There is no 100% provably secure public-key cryptosystem. Thus, uncertainty will always be play a role. All that can be done is to provide sufficient evidence that potential attacks will be fruitless.

Bibliography

[1] S. Alaca and K.S. Williams, *Introductory Algebraic Number Theory*, Cambridge University Press, New York, 2004.

[2] J. Ava, Multivariate FussCatalan Numbers. *Discrete Mathematics*, 308(2008), 4660-4669.

[3] E. F. Brickell, The Cryptanalysis of Knapsack Cryptosystems, *Applications of Discrete Mathematics*, R. D. Ringeisen and F. S. Roberts, eds., SIAM (1988), 3-23.

[4] E. F. Brickell, J. A. Davis, and G. J. Simmons, A Preliminary Report on the Cryptanalysis of Merkle-Hellman Knapsack Cryptosystems, *Advances in Cryptology*, (1983), 289-301.

[5] E. F. Brickell, Solving Low Density Knapsacks, *Advances in Cryptology*, (1983), 2537.

[6] E. F. Brickell and A. M. Odlyzko, Cryptanalysis: A Survey of Recent Results, *Proc. IEEE*, 76(1988), 578-593.

[7] B. Chor and R. L. Rivest, A knapsack-type public key cryptosystem based on arithemetice in finite fields. *IEEE Trans. on Information Theory*, 34(1988), 901-909.

[8] M. J. Coster, B. A. LaMacchia, A.M. Odlyzko, and C.P. Schnorr, An improved low-density subset-sum algorithm, *Advances in Cryptology - EUROCRYPT'91* (LNCS 547), 54-67.

[9] T. Davis, "Calalan Numbers". 26 Nov. 2006. http://www.geometer.org/ mathcircles/catalan.pdf.

[10] D. DeTemple, Pick's Fromula: A Retrospective, Mathematics Noted from Washington State University, Vol. 32, Nos. 3-4, 1989.

[11] W. Diffie, and M. Hellman, New directions in cryptography, *IEEE Trans. Inform. Theory*, 22(1976), 644-654.

[12] L. Evain, Knapsack cryptosystems built on NP-hard instances. *Cryptology ePrint Archive*, Report 2008/106.

[13] A. S. Frankel, Systems of Numeration, *American Mathematical Monthly*, 92(1985), no. 2, 105-114.

[14] A. Frieze, On the Lagarias-Odlyzko Algorithm for the Subset Sum Problem. *SIAM J. Comput.*, 15(1986), no. 2, 536-539.

[15] N. Hamlin and W. Webb, Representing Positive Integers as a Sum of Linear Recurrence Sequences. (not published yet).

[16] J. H. Jordan and C. J. Potratz, Complete residue systems in the Gaussian integers, *Mathematics Magazine*, 38(1965), 1-12.

[17] P. Kaski and P. R. J. stergrd, Classification Algorithms for Codes and Designs, Springer, New York, 2006.

[18] J. C. Lagarias, Knapsack Public Key Cryptosystems and Diophantine Approximation, *Advances in Cryptology: Proceedings of Crypto 83*, (1984), 3-24.

[19] J. C. Lagarias and A. M. Odlyzko, Solving Low Density Subset Sum Problems, *J. Assoc. Comp. Mach.*, (32)1985, 229-246.

[20] M. Lai, Knapsack Cryptosystems: The Past and the Future. Technical report, Department of Infor- mation and Computer Science,University of California, 2001. http://www1.ics.uci.edu/mingl/ knapsack.html.

[21] A. Lenstra, Integers Factoring, *Designs, Codes, and Cryptography*, 19(2000), 101-128.

[22] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovasz, Factoring Polynomials with Rational Coefficients, *Mathematische Annalen*, 261(1982), 515-534.

[23] R. C. Merkle and M. E. Hellman, Hiding Information and Signatures in Trapdook Knapsacks, *IEEE Transactions on Information Theory*, 24(1978), 525-530.

[24] E. Miller, V. Reiner, and B. Sturmfels, Geometric Combinatorics, American Mathematical Society, 2007.

[25] Y. Murakami and M.Kasahara, A New Class of Public-Key Cryptosystem with Several Terms of Product-Sum Operation. Proceedings of SCIS, 2006.

[26] Y. Murakami and T. Nasako, Knapsack Public-Key Cryptosystem Using Chinese Remainder Theorem. Proceedings of the Symposium on Information Theory and Its Applications, 2006.

[27] N. J. A. Sloane, Ed. (2008), The On-Line Encyclopedia of Integer Sequences, published electronically at www.research.att.com/ njas/sequences/, Sequence A000081.

[28] N. J. A. Sloane, Ed. (2008), The On-Line Encyclopedia of Integer Sequences, published electronically at www.research.att.com/ njas/sequences/, Sequence A001190.

[29] P. Nguyen and J. Stern, Adapting density attacks to low-weight knapsacks, *Advances in Cryptology - ASIACRYPT'05* (LNCS 3788), 41-58.

[30] T. Okamoto, K. Tanaka, and S. Uchiyama, Quantum Public-Key Cryptosystems. *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology* (August 20 - 24, 2000). M. Bellare, Ed. Lecture Notes In Computer Science, vol. 1880. Springer-Verlag, London, 147-165.

[31] Recommendation for Key Management Part 1: General, NIST Special Publication 800-57. March, 2007.

[32] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(q) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1978), 106110.

[33] R. L. Rivest, A. Shamir, and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Commun. ACM*, 21(1978), 120-126.

[34] RSA Factoring Challenge. (2007, July 14). In Wikipedia, The Free Encyclopedia. Retrieved 19:05, November 21, 2007, http://en.wikipedia.org/w/ index.php?title=RSAFactoringChallenge.

[35] A. Shamir, A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem, *IEEE Symposium on Foundations of Computer Science*, (1982), 145152.

[36] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(1997), no. 5, 1484-1509.

[37] H. M. Stark, An Introduction to Number Theory, Mit Press, 1978.

[38] K. S. Williams, Integers of Biquadratic Fields, *Canadian Mathematical Bulletin*, 13(1970), 519-526.

[39] E. Zeckendorf, Reprsentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas *Bull. Soc. R. Sci. Lige*, 41(1972), 179182.