ABSTRACT

ACITELLI, CATHERINE BARBARA. The Design of an Undergraduate Cryptography Course with Python and SageMath. (Under the Direction of Ernest Stitzinger).

The field of Cryptography is rapidly evolving, and the development of quantum computers is on the rise. Lattice-based cryptosystems are promising candidates for quantum resistance, and it is the responsibility of institutions to prepare students for related careers. Undergraduate students in mathematics and mathematics-adjacent fields can – and should – study lattice-based cryptography. However, the field lacks a comprehensive undergraduate curriculum that emphasizes the mathematical depth and the practical applications of cryptography. Thus, we focus on the undergraduate accessibility of lattice-based cryptography with Python through the development of a self-contained course that relies solely only on a background in linear algebra. We wove the use of Python, the most widely used computer language, and SageMath, a Python-based computer algebra system, throughout the course to complement the mathematical theory. The main focuses of the course are lattice-based cryptosystems and lattice reduction algorithms. A standard 15—week semester begins with a linear algebra review and an introduction to Python and Sage. Necessary abstract algebra and number theory are introduced as they become relevant. By expanding on decades of cryptography research and taking a novel application-based approach, completion of this course accomplishes five main outcomes for students: it enhances their algebraic thinking, allows them to learn the Python computing language, gives them the opportunity to explore multiple facets of algorithm design, enables them to see the fluidity of mathematics, and prepares them well for a variety of different career paths.

Our work seeks to prepare students for the growing competitive landscape of career paths necessitating expertise in the totality of cryptosystems, so we have also included complementary focal areas on Digital Signature Schemes, Blind Signature Schemes, and Zero Knowledge Proofs. A Digital Signature binds a signer to a document and allows the recipient to verify the authenticity of the document. A Blind Signature first conceals the message before a third party signs it. A Zero Knowledge Proof allows a prover to prove that they know a piece of information without revealing it to the verifier. We created signature schemes, blind signature schemes, and zero knowledge proof protocols to accompany the GGH and NTRU cryptosystems. Inclusion of these elements into our work further underscores our dedication to delivering a well-rounded and thorough undergraduate-level course in Cryptography.

The Design of an Undergraduate Cryptography Course with Python and SageMath

by
Catherine Barbara Acitelli

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Mathematics

Raleigh, North Carolina

2022

APPROVED BY:

Jo-Ann Cohen	Alina Duca
Erin Krupa	Kailash Misra

Ernest Stitzinger
Chair of Advisory Committee

DEDICATION

To my parents, AJ and Beth Acitelli, for their constant love and support during my academic career.
To my grandfather, Mario Acitelli, Ph.D., for his encouragement and enthusiasm throughout my Ph.D. journey.

BIOGRAPHY

Catie Acitelli was born in Binghamton, New York to AJ and Beth Acitelli, the third of four children. She grew up in Weddington, North Carolina and graduated from Charlotte Catholic High School in 2008. Catie was a North Carolina Teaching Fellow at North Carolina State University, where she earned Bachelors of Science degrees in Mathematics and Secondary Mathematics Education. As an undergraduate student, she began her teaching career in 2009 as a recitation leader for various calculus courses. After graduating in 2012, Catie began teaching high school mathematics and coaching cheerleading in Charlotte, North Carolina. She wrote district curricula for the Advanced Functions and Modeling and Pre-Calculus courses and trained teachers on the effective implementation of the curricula. She earned the Most Valuable Teacher Award at two different high schools. After teaching and coaching for six years, Catie returned to North Carolina State University to pursue a Ph.D. in Mathematics under the direction of Dr. Ernest Stitzinger. During her graduate work, she served as a Teaching Assistant for various calculus and discrete math courses and as the Instructor of Record for a number of calculus and linear algebra courses. Catie was awarded the 2021 Graduate Student Association Outstanding Teaching Assistant Award and the 2021 Mathematics Department Maltbie Award for Excellence in Teaching. As a Preparing the Professoriate Fellow, Catie redesigned the Introduction to Linear Algebra curriculum to emphasize the importance of proof writing. She was also a significant contributor to the new curriculum for first year graduate students in the Mathematics Department, which emphasizes both teaching and leadership. She leverages her expertise in curriculum writing for her research at North Carolina State University. When Catie is not doing math, she is watching a sports game, cooking, listening to a true crime podcast, or hanging out with her dog, Bella.

ACKNOWLEDGEMENTS

I would like to thank everyone at North Carolina State University who believed in me, encouraged me, pushed me, and ultimately helped me grow as a mathematician. First and foremost, I would like to thank my dissertation advisor, Dr. Ernest Stitzinger, for your constant support and guidance and for our many helpful (and fun) conversations. You believed in me when I didn't believe in myself, and you always provided me with the encouragement and enthusiasm I needed to keep going. Thank you for trusting me with this project and for reminding me that it is okay to take breaks.

To my family, thank you for your unwavering support of my academic endeavors. Thank you to my siblings, Rachelle, Mario, and Angelo, for always understanding when I had to say "no" to a birthday party or family event. Thank you for being excited about every important milestone in my Ph.D. journey. And thank you to Sam, Sara, and Caroline, for your encouragement along the way. I could not have done it without your support. Thank you to my parents, Beth and AJ, for helping in every way you knew how, including watching your granddog, Bella, pretty frequently. Thank you for always listening, for always being invested, and for always being in my corner. Thank you especially to my sister, Rachelle Reed, Ph.D., for just "getting it" when I felt like nobody else did, reminding me that I am not alone, and setting an example for what it looks like to be a woman in STEM. To my neices and nephews - Leona, Addie Grace, AJ, Ellie, Graydon, and Madison - thank you for being a light and for rooting Aunt Miss Catie on, even if you didn't realize it. Thank you to my grandpa, Mario Acitelli, Ph.D., for your constant support and excitement. I always enjoyed hearing about your experiences as a graduate student, and I am so grateful to have had such a great role model. I hope that I make you proud. And to Bella, thank you for helping me get through it all and for loving me no matter what.

To all of the teachers that helped me get to where I am today, I cannot thank you enough. To the first math teacher that really challenged me, Trish Wendover, thank you for helping me see the beauty of math and for seeing something special in that awkward fourth grader in your classroom. To my high school teachers, Cat Jordan and Joanne Winters, you were instrumental in ensuring a solid mathematical foundation and a love for the subject. You gave me incredible examples of how to teach and inspired me to do just that. To John Griggs, thank you for believing in me since day one and for providing me with my first teaching opportunity that would set me on the path to doing what I love most. Thank you to my coworkers from my high school teaching days, for helping to shape me into the educator I am today. And to all of my other teachers and professors along the way - thank you.

Thank you to the other members of this Ph.D. program at NC State. You each are an

inspiration, and you kept me going. I appreciate all the helpful conversations, both about math and otherwise, and I could not have done it without you all. In particular, thank you to Michael, Ben, and Dana, for reminding me to take breaks and to have a little fun. And thank you to my “non-math” support system of friends. You all kept me grounded and helped me navigate the emotional ride of graduate school. In particular, thank you to Carolyn and Kassidy, for always uplifting me, cheering me on, and reminding me that I can do it. I am forever grateful.

Thank you to the other Mathematics Education Masters and Ph.D. students, for welcoming me into your program and classes with open arms and minds. I appreciate all the pedagogical conversations, and I am grateful for your excitement about - and passion for - education.

Thank you to Panera Bread and Starbucks, for providing me with the spaces in which I was most productive. To the employees, thank you for your kindness and your interest in my work. And for the never-ending supply of brain fuel in the form of coffee.

Lastly, thank you to North Carolina State University and the Department of Mathematics, for letting me call you home for eight academic years - some of the best of my life. Go Pack!

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

1.1 A Brief Introduction to Public Key Cryptography

Two entities, call them Alice and Bob, want to exchange information without anyone else knowing what it is that they are saying. They encrypt messages to send to each other, and they decrypt the messages that they receive from one another. The encrypted message is called the *ciphertext* and the decrypted message is called the *plaintext*. In other words, the plaintext is the message they want to send, and the ciphertext is the message that actually gets sent. For a long time, Alice and Bob did this by agreeing on a *secret key* that their adversary (or eavesdropper), Eve, did not have. This meant that Alice would encrypt a message using a secret key and send it to Bob. Bob would use that same secret key to decrypt the message and recover the plaintext. Since Eve did not know the secret key, she was unable to decrypt a ciphertext that she intercepted. We have just described a *private key cryptosystem*. In these cryptosystems, Alice and Bob know the same amount of information. For this reason, these ciphers are known as *symmetric ciphers*.

In 1976, Whitfield Diffie and Martin Hellman introduced the idea of *public key cryptography* and made major contributions to the field. [8] A public key cryptosystem has the advantage that anyone can encrypt a message, but only the key creator can decrypt it.

Let us assume now that Bob is sending messages to Alice, and Alice needs to decrypt the ciphertext messages to recover the plaintext messages. In a public key cryptosystem, Alice has a *private key* that only she knows and a *public key* that everyone knows. Bob uses the public key to encrypt a message to send to Alice, and Alice uses the private key to decrypt it. Note that anyone can encrypt a message, but only Alice can decrypt it. Eve has a hard time decrypting an intercepted ciphertext, because she does not have the private key. Since Alice and Bob know different amounts of information in a Public Key Cryptosystem, this idea is called an *asymmetric cipher*.

A public-key cryptographic scheme has three parts:

- *Key generation*: Given a security parameter, this outputs a public key and a private key.
- *Encryption algorithm*: This takes a public key and a plaintext message as input and outputs a ciphertext message.
- *Decryption algorithm*: This takes as input a private key and ciphertext message and outputs a plaintext message.

We will focus solely on public key cryptography, with an emphasis on more modern mathematical constructions.

1.2 Proposed Course and Course Design Aspects

Secure Public Key Cryptosystems have an encryption algorithm that is easy to compute with the private key and a decryption algorithm that is hard to compute with only the public key. This idea is known as a *one-way function*. In other words, the security of Public Key Cryptosystems relies on the hardness of their underlying problems. For example, the RSA Cryptosystem [25], which is the most widely used encryption scheme, relies on the hardness of integer factorization.

It is easy to multiply two prime numbers, say 263 and 839, and get a composite number, 220,657. It is much more difficult to begin with the composite number 220,657, and factor it into 263×839. This is the *integer factorization problem*. When numbers are small, it is not difficult to solve by brute force. However, the largest known prime number has 24,862,048 digits. [17] You can imagine how this gets difficult quickly.

There is currently not a way for a classical computer to find the prime factors of a composite number in less than exponential time, as classical computers cycle through cases individually, testing one number at a time. However, Shor's algorithm factors integers on a quantum computer in polynomial time. [26]

As soon as quantum computers have enough stable quantum bits (qubits), many known cryptosystems will become insecure and therefore pose a security risk. Companies like IBM, Google, and Microsoft have invested billions of dollars into the research and development of quantum computers, racing to achieve quantum supremacy. In fact, IBM promises a 1000 qubits quantum computer by 2023. [28] This gives rise to a need for quantum-safe - or at least quantum-resistant - cryptosystems, and constructions based on integer lattices (defined below and discussed in Section 5.1) are proving promising. [21] The National Security Agency affirms that constructions of cryptosystems based on hard lattice problems are "among the most efficient post-quantum designs." [20]

Definition 1 (Integer Lattice)**.**
Let $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_n \in \mathbb{Z}^n$ *be a set of linearly independent vectors. The integer lattice (or integral lattice) is given by*

$$\mathscr{L} = \left\{ a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + a_3\mathbf{v}_3 + \cdots + a_n\mathbf{v}_n : a_i \in \mathbb{Z}, 1 \leq i \leq n \right\}.$$

Lattice-based cryptography is more relevant now than ever, and undergraduate students need to be exposed to it as early as possible. Until now, undergraduate mathematics cryptography courses have focused largely on historical cryptosystems rather than more modern ones. Aside from the obvious value of learning the history of cryptography, mathematics educators often use this type of course setting as a catch-all class for applications of Abstract Algebra, as many early cryptosystems make use of number theory and algebraic structures. The reason for the omission of more modern systems is likely twofold: many of the modern hardness proofs involve advanced graduate level mathematics, and cryptanalysis is deeply rooted in computer programming. This is why the vast majority of university cryptography courses are housed in graduate Computer Science or related programs and assume prior computer programming knowledge. Modern cryptography courses in mathematics departments are typically saved for graduate studies. However, much of this content is accessible to undergraduate students who have taken a course in Linear Algebra.

The course that we are proposing is designed as an introductory course for undergraduates to explore modern cryptography from both theoretical and application-based perspectives. Students will gain practical experience through their interactions with the course content, the SageMath Computer Algebra System, and Python Programming. The main goal of this course is to make introductory lattice-based cryptography accessible to undergraduate students in mathematics, computer science, engineering, and other related fields. By engaging students with the theoretical and practical applications of modern

cryptographic algorithms, students will be prepared to enter careers in related fields upon graduation.

1.2.1 Intended Audience

The basic study of modern lattice-based cryptography requires that students have a background in linear algebra and familiarity with basic proof structures. [14] This content is, however, reviewed as needed in the proposed course. For example, prior to even defining a lattice, there is a review of vector spaces from linear algebra. Students may also benefit from a familiarity with some elements of abstract algebra and number theory. However, this is a self-contained introductory course, and necessary background from the aforementioned content is included. While prior programming experience may also be beneficial, it is not required, as this course assumes no such prior knowledge.

Lattice-based cryptography has direct implications to cybersecurity, and students in a variety of academic programs may have interest in studying this closely. The Joint Task Force in Cybersecurity Education lists eight basic knowledge areas that all post-secondary Cybersecurity Curricula should include. [7] None of these main areas is mathematics. Each knowledge area is divided into knowledge units, and each knowledge unit is further divided into topics. One of these topics is devoted to "mathematical background" and consists of basic abstract algebra concepts and pre-quantum encryption algorithms. Knospe's *A Course in Cryptography* [16] asserts that "cryptography can easily be underestimated by mathematicians," as they do not consider the whole picture of the cryptosystem, but that "algebra can be a major stumbling block" for non-math majors. The biggest pitfall to current cryptography courses is the failure to recognize the complexity of cryptography and the depth of its mathematical foundation. Thus, students must understand the underlying mathematical theory. Since modern cryptography is based on computations that are impractical to do by hand, students must also know how to practically implement a variety of algorithms on a computer program. *An Introduction to Cryptography with Python* will approach the content from both angles, providing students from various programs the opportunity to explore modern cryptography.

1.2.2 The Use of Python and SageMath

Undergraduates in mathematics programs are typically exposed to computer programming by way of MATLAB, Mathematica, or Maple, if at all. It is common that mathematics and mathematics education majors take at most one introductory course in a computer language. They do not get the same type of practical or hands-on experience with applications

as their counterparts in computer science related fields, as mathematics degree programs are often more grounded in theory. Undergraduates in computer science, computer engineering, computer software development, cybersecurity, etc. (CS) begin writing code as soon as they enter college. It is also common that CS majors take the standard Calculus sequence and Linear Algebra; however, they do not get the same type of theoretical experience later on as their peers in mathematics programs. Extending the National Research Council's recommendation of a balance in conceptual and procedural aspects of K-12 mathematics curricula, [9] this post-secondary course will employ a parallel development of the theory behind cryptographic systems and the implementation of their algorithms.

The choices of Python and SageMath are purposeful. SageMath, or Sage for short, is a powerful Python-based Computer Algebra System that allows students to explore the basics of Python and computer programming in general. It also provides a bridge for the algebraic gaps that often occur for students. Python was originally designed by Guido van Rossum for readability. [22] Compared to other languages, it is easy to teach, easy to learn, and easy to understand because of its simple syntax. Historically, the first program one writes when learning a new language is "Hello World." The Python code to do this is

```
print(''Hello World'')
```

while the Java code is

```
class Hello {
    public static void main (String[] args) {
System.out.printIn(''Hello World'');
}
}
```

Java programming does not lend itself nicely to cryptography from a syntactical standpoint. Programmers spend more time trying to figure out nuances of the code than they do understanding the algorithms. A similar argument can be made for other languages as well. In a 2009 experimental study on the effectiveness of Maple Worksheets for Student Cryptography Exploration, it was found that students with more programming background opted to use languages like C and C++ for their projects, but they were unable to replicate the coding in those less forgiving environments. [19] It is important to note that Maple is used in educational settings, but not often in industry outside of engineering.

In designing a curriculum, it is necessary to consider the applicability of what students learn to related careers [7], and Python is one of the most widely used computer languages worldwide. As one of the most widely used languages, Python is compatible with a variety

of different operating systems and it is both free to use and open source, so it is accessible to all. It is also computationally powerful, as it has built-in support for exact arithmetic of large numbers. [22]

To supplement the use of Python, this course will also employ the power of SageMath, a Python-based, open source computer algebra system. It allows students to manipulate algebraic expressions and learn the language of Python along the way. Sage has many built-in functions that allow users to compute things rather quickly. Additionally, students who do not have a strong algebraic background are not prevented from course participation. In fact, bridging the algebraic gap could allow for more active participation in the classroom and, ultimately, further advances in the field. Students can focus on the concepts and applications without getting "hung up" on the algebra. In this course, Sage is used as a gateway to Python.

1.2.3 Content Organization

This new Introduction to Cryptography course uses an integrated approach via SageMath and Python, introducing both necessary mathematical and programming background as needed. For example, students at this stage of their mathematical careers are undoubtedly familiar with computing the greatest common divisor (gcd) of two integers. This is one of the first codes students will write in this course. Leveraging this prior knowledge in the beginning of the semester to explore programming loops allows students to both understand the structure and appreciate the power of a loop. The curriculum includes sample algorithms and ready to implement Python and Sage code. Students are also presented with general algorithms and asked to write corresponding programs as exercises.

After working through their own computer code, students will then be provided with actual Python code, so that they can implement it immediately. Since there are many ways to write a code that will take in two integers and return their gcd, for example, students can then engage in rich discussion around the content. In addition to standard practice exercises at the conclusion of each section, there are collaborative and coding exercises serving as embedded assessments that challenge students to use what they have learned to make their current code more efficient and to develop new code.

Another way that this course activates prior knowledge to build connections is through motivating examples and proof structures. For example, while the proof of the Cauchy-Schwarz Inequality for dot product in Euclidean Space can be done in a variety of ways, the method chosen in the background Linear Algebra chapter of this course hinges on the relationship between the discriminant and the roots of a quadratic function. This theme is carried throughout the course, as students are expected to draw on their linear algebra

background to build up to lattice reduction algorithms. One of the first cryptographically relevant algorithms students encounter is the Gram-Schmidt orthogonalization algorithm, a standard algorithm for any first-year linear algebra course. The Gram-Schmidt Orthogonalization Algorithm serves as the basis for algorithms that students encounter in this course, most notably the LLL algorithm of Lenstra, Lenstra, and Lovász. [18]

The course begins with an overview of basic linear algebra, following [2] and [12], and basic number theory, following [10] and [15]. Mathematical content is introduced as it is needed for successful progression in the course. Since the only required prerequisite is an introductory linear algebra course, concepts such as polynomial rings are introduced just before studying the NTRU Public Key Cryptosystem, for example. Concepts are introduced at a level of rigor that is consistent with what is necessary for understanding their cryptographic applications. [14]

A written curriculum accompanies the course. The beginning workings of this curriculum are presented in this dissertation and will continue to develop into a textbook. Each section in the written curriculum contains motivating examples, an encryption scheme, a decryption scheme, an adversarial attack scheme, algorithms, pseudocode, Python and Sage code, worked examples, and practice problems. Corresponding signature schemes, blind signature schemes, and zero knowledge proofs are discussed in their own chapters. In creating coding exercises, this course adopts the notion that practice exercises should contain problems that allow for computational practice and conceptual exploration. Algorithms and given blocks of code are presented in their entirety.

There are four possible tracks for teaching cryptography – mathematics, computer science, computer engineering, and information security – and they are typically disjoint. In order to merge these four tracks, cryptography courses should include the introduction of mathematical background and classical algorithms without a computer, step-by-step presentation of algorithms and pseudocode, and actual code implementation. Assigned problems should allow students to work through computations by hand before turning to a computer program so that they better understand the mathematical intricacies of the algorithm.

The exercises in the text for this introductory cryptography course are structured in a way that resembles this idea. The first exercises in each section are designed to be completed by hand, and they precede coding exercises for the same concepts and cryptosystems. The intention is that students gain a deep understanding of, and appreciation for, the power of the algorithm. As mentioned, this thesis includes course notes that will later be restructured into a textbook.

1.2.4 Course Objectives

The purpose of this course is to familiarize undergraduate students with lattice-based cryptography and prepare them for associated careers. By the end of the course, students will be able to make sense of cryptographic encryption and decryption algorithms based on (computationally) hard lattice problems and implement them in Python and SageMath. Computational hardness refers to the efficiency of the algorithm.

This course explores hard problems based on lattices. While they will be discussed in detail later, it is worth a preview now, as it illustrates the course progression. Two of the known hard problems with lattice constructions are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP), and they are closely related. Following Hoffstein, Pipher, and Silverman [14], we will define the SVP and CVP as follows:

Problem 1.2.1 (The Shortest Vector Problem)**.**
Find a shortest nonzero vector in lattice $\mathscr{L}$. In other words, find $0 \neq \mathbf{v} \in \mathscr{L}$ such that $\|\mathbf{v}\|$ is minimized.

Problem 1.2.2 (The Closest Vector Problem)**.**
Given a vector $\mathbf{w} \in \mathbb{R}^n$ that is not in $\mathscr{L}$, find a vector $\mathbf{v} \in \mathscr{L}$ that is closest to $\mathbf{w}$. In other words, find a vector $\mathbf{v} \in \mathscr{L}$ that minimizes the Euclidean norm $\|\mathbf{w}-\mathbf{v}\|$.

Students are first introduced to the 2-dimensional version of the SVP and the CVP, so that they are able to visualize the effect of orthogonality on the solvability of these problems. They will explore the relationship of a fundamental lattice domain to the CVP in order to describe the conditions under which Babai's Algorithm actually finds the closest vector. The closest lattice vector to a non-lattice target point is the closest vertex of a fundamental domain of a lattice when the fundamental domain meets specific criteria, of which students will be able to identify. This is discussed in detail in Section 5.3. This translates directly to the construction of the GGH public key cryptosystem. Alice is able to decrypt a ciphertext using the private key, formed by reasonably orthogonal lattice vectors, while Eve can likely not decrypt it with only the public key, formed by reasonably parallel vectors. As their first application of studying lattices, students will be able to describe both problems and solve them in low dimension by hand using 2—D Gaussian Lattice Reduction and Babai's Algorithm before using Sage or Python to extend these ideas to higher dimensions. [3] Much of the known lattice-based cryptography hinges on the hardness of the SVP and CVP and associated problems, so students should become intimately familiar with them via two-dimensional visualizations, worked examples, practice exercises, and collaborative work.

A natural next thought for students is that reasonably parallel vectors ought to be able to be turned into reasonably orthogonal ones. The progression of *An Introduction to Cryptography with Python* allows students to think creatively about next steps. Drawing on prior knowledge, students should recognize that the first exposure they had to an algorithm that mimics lattice basis reduction was the Gram-Schmidt Algorithm from their linear algebra course and the review section. Much of the coursework here requires that students have Gram-Schmidt mastered from both a theoretical standpoint and a practical standpoint. In fact, the LLL algorithm of Lenstra, Lenstra, and Lovász calls the Gram-Schmidt function to execute. The LLL algorithm is a lattice reduction algorithm that inputs a random basis and outputs an LLL basis. Though this is discussed in detail later in the course, roughly speaking, an LLL basis is a basis that can solve what is known as the approximate SVP. Students in this course will be able to understand theoretically and implement practically the LLL algorithm to reduce a "bad" basis to a "good" basis in order to solve the SVP, CVP, and closely related problems in a reasonable amount of time. Here,"reasonable" refers to polynomial time.

1.2.5 Comments on the Text

What follows is a collection of course notes, or portions of the written curriculum, for the aforementioned course. The intention is for these notes to form the foundation for a textbook. It is important to note that this is a working document. We do not claim that it is complete, and we intend to continue to rework and revise aspects of this written curriculum as the course runs in subsequent semesters.

Cryptography is concerned with the sending and receiving of messages. The goal is to get a message from point A to point B without an eavesdropper (E) figuring out what the message says. The notion of A to B without E evolved into "Alice" to "Bob" without "Eve." Throughout the following notes, we will follow convention and use Alice (she/her), Bob (he/him), and Eve (she/her) for the message recipient, sender, and eavesdropper, respectively. Signature schemes involve the signing of a document by Samantha (she/her), and the verification of the signature by Victor (he/him). Likewise, zero knowledge proof protocols involve the prover, Peggy (she/her), attempting to prove that she knows some information without the verifier, Victor (he/him), knowing the information.

The goal of this course is for students to becoming familiar with lattice-based cryptosystems. Since a lattice is very similar to a vector space, it is necessary that students review their linear algebra notions of vector spaces. For this reason, we begin with a linear algebra review.

CHAPTER

2

A REVIEW OF LINEAR ALGEBRA

This section is composed of well-known definitions and results in Linear Algebra. They appear in a number of introductory textbooks. See, for example, [2] or [12], which we have used here.

2.1 Vector Spaces

While it is not necessary to memorize the ten axioms of vector spaces, it is important to familiarize yourself with the properties before moving forward.

Definition 2 (Real Vector Space)**.**
Let V be a set equipped with addition as $\oplus$ and scalar multiplication as $\odot$. Then $(V, \oplus, \odot)$ is a real vector space if for all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ and for all $a, b \in \mathbb{R}$,

1. *(Closure under addition)* $\mathbf{u} \oplus \mathbf{v} \in V$
2. *(Commutativity of addition)* $\mathbf{u} \oplus \mathbf{v} = \mathbf{v} \oplus \mathbf{u}$
3. *(Associativity of addition)* $(\mathbf{u} \oplus \mathbf{v}) \oplus \mathbf{w} = \mathbf{u} \oplus (\mathbf{v} \oplus \mathbf{w})$
4. *(Additive identity) There exists a* $\mathbf{0} \in V$ *such that* $\mathbf{u} \oplus \mathbf{0} = \mathbf{u} = \mathbf{0} \oplus \mathbf{u}$
5. *(Additive inverse) There exists a* $-\mathbf{u} \in V$ *such that* $\mathbf{u} \oplus -\mathbf{u} = \mathbf{0} = -\mathbf{u} \oplus \mathbf{u}$

6. *(Closure under scalar multiplication)* $a \odot \mathbf{u} \in V$
7. *(Multiplicative identity) There exists a* $1 \in \mathbb{R}$ *such that* $1 \odot \mathbf{u} = \mathbf{u}$
8. *(Associativity of scalar multiplication)* $(a\,b) \odot \mathbf{u} = a \odot (b \odot \mathbf{u})$
9. *(Distributivity)* $a \odot (\mathbf{u} \oplus \mathbf{v}) = a \odot \mathbf{u} \oplus a \odot \mathbf{v}$
10. *(Distributivity)* $(a+b) \odot \mathbf{u} = a \odot \mathbf{u} \oplus b \odot \mathbf{u}$

Example 2.1.1 (Not a Vector Space)**.**
Let $V = \mathbb{R}$ *with the following addition and multiplication. For all* $a, b \in \mathbb{R}$ *and scalars* $k \in \mathbb{R}$,

$$a \oplus b = 4a + 4b \qquad k \odot a = ka.$$

Then, $(V, \oplus, \odot)$ *is not a vector space since, for example, axiom* 3 *does not hold. Observe that*

$$\begin{aligned}(1 \oplus 2) \oplus 3 &= (4 \cdot 1 + 4 \cdot 2) \oplus 3 \\ &= 12 \oplus 3 \\ &= 4(12) + 4(3) \\ &= 60\end{aligned}$$

but that

$$\begin{aligned}1 \oplus (2 \oplus 3) &= 1 \oplus (4 \cdot 2 + 4 \cdot 3) \\ &= 1 \oplus 20 \\ &= 4(1) + 4(20) \\ &= 84\end{aligned}$$

Since $(a \oplus b) \oplus c \neq a \oplus (b \oplus c)$ *for all* $a, b, c \in R$, $(V, \oplus, \odot)$ *is not a real vector space*

Example 2.1.2 (Vector Space [2])**.**
Let $V = \mathbb{R}^2 = \left\{ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} : u_1, u_2 \in \mathbb{R} \right\}$ *with addition and multiplication defined below. For all* $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ *and* $k \in \mathbb{R}$,

$$\mathbf{u} \oplus \mathbf{v} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \oplus \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} u_1 + v_1 + 1 \\ u_2 + v_2 + 1 \end{bmatrix} \qquad k \odot \mathbf{v} = k \odot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} k\,v_1 + k - 1 \\ k\,v_2 + k - 1 \end{bmatrix}.$$

We will show that $(V, \oplus, \odot)$ *is a real vector space by checking each of the axioms.*

1. $\mathbf{u}\oplus\mathbf{v}\in V$ *since* $\mathbf{u}\oplus\mathbf{v}=\begin{bmatrix}u_1\\u_2\end{bmatrix}\oplus\begin{bmatrix}v_1\\v_2\end{bmatrix}=\begin{bmatrix}u_1+v_1+1\\u_2+v_2+1\end{bmatrix}$ *is a vector in* $\mathbb{R}^2$.

2. $\mathbf{u}\oplus\mathbf{v}=\mathbf{v}\oplus\mathbf{u}$ *because* $\mathbf{u}\oplus\mathbf{v}=\begin{bmatrix}u_1+v_1+1\\u_2+v_2+1\end{bmatrix}$ *and* $\mathbf{v}\oplus\mathbf{u}=\begin{bmatrix}v_1+u_1+1\\v_2+u_2+1\end{bmatrix}$. *By commutativity of addition of real numbers, these two statements are equal.*

3. *We want to show that* $(\mathbf{u}\oplus\mathbf{v})\oplus\mathbf{w}=\mathbf{u}\oplus(\mathbf{v}\oplus\mathbf{w})$ *for all* $\mathbf{u},\mathbf{v},\mathbf{w}\in V$. *Considering each side separately, observe that*

$$\begin{aligned}(\mathbf{u}\oplus\mathbf{v})\oplus\mathbf{w}&=\begin{bmatrix}u_1+v_1+1\\u_2+v_2+1\end{bmatrix}\oplus\begin{bmatrix}w_1\\w_2\end{bmatrix}\\&=\begin{bmatrix}(u_1+v_1+1)+w_1+1\\(u_2+v_2+1)+w_2+1\end{bmatrix}\\&=\begin{bmatrix}u_1+v_1+w_1+2\\u_2+v_2+w_2+2\end{bmatrix}\qquad\qquad\textit{and}\\\mathbf{u}\oplus(\mathbf{v}\oplus\mathbf{w})&=\begin{bmatrix}u_1\\u_2\end{bmatrix}\oplus\begin{bmatrix}v_1+w_1+1\\v_2+w_2+1\end{bmatrix}\\&=\begin{bmatrix}u_1+(v_1+w_1+1)+1\\u_2+(v_2+w_2+1)+1\end{bmatrix}\\&=\begin{bmatrix}u_1+v_1+w_1+2\\u_2+v_2+w_2+2\end{bmatrix}\end{aligned}$$

4. *We want to show that there exists a* $\mathbf{0}_V\in\mathbb{R}^2$ *such that* $\mathbf{u}\oplus\mathbf{0}_V=\mathbf{u}=\mathbf{0}\oplus\mathbf{u}$ *for all* $\mathbf{u}\in V$. *That is, we want to find* $\mathbf{y}$ *such that* $\mathbf{u}\oplus\mathbf{y}=\mathbf{u}$. *Note that* $\mathbf{u}\oplus\mathbf{y}=\begin{bmatrix}u_1+y_1+1\\u_2+y_2+1\end{bmatrix}$. *We want this to equal* $\begin{bmatrix}u_1\\u_2\end{bmatrix}$. *Equating these gives that* $y_1=-1$ *and* $y_2=-1$. *So, there does exist a* $\mathbf{0}_V\in\mathbb{R}^2$ *such that* $\mathbf{u}\oplus\mathbf{0}_V=\mathbf{u}$. *Namely,* $\mathbf{0}_V=\mathbf{y}=\begin{bmatrix}-1\\-1\end{bmatrix}$. *One can verify that* $\mathbf{0}_v\oplus\mathbf{u}=\mathbf{u}$ *as well.*

5. *We want to show that there exists a* $\mathbf{w}\in\mathbb{R}^2$ *such that* $\mathbf{u}\oplus\mathbf{w}=\mathbf{0}_V$. *That is, we want to find* $w_1,w_2\in\mathbb{R}$ *such that* $\begin{bmatrix}u_1\\u_2\end{bmatrix}\oplus\begin{bmatrix}w_1\\w_2\end{bmatrix}=\begin{bmatrix}-1\\-1\end{bmatrix}$. *Setting* $\begin{bmatrix}u_1+w_1+1\\u_2+w_2+1\end{bmatrix}=\begin{bmatrix}-1\\-1\end{bmatrix}$ *yields* $\mathbf{w}=\begin{bmatrix}-2-u_1\\-2-u_2\end{bmatrix}$. *So, the additive inverse of* $\mathbf{u}$ *is* $\begin{bmatrix}-2-u_1\\-2-u_2\end{bmatrix}$.

6. $k \odot \mathbf{v} \in V$ *since* $k \odot \mathbf{v} = k \odot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} k v_1 + k - 1 \\ k v_2 + k - 1 \end{bmatrix}$ *is a vector in* $\mathbb{R}^2$.

7. $1 \odot \mathbf{v} = 1 \odot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 1 v_1 + 1 - 1 \\ 1 v_2 + 1 - 1 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \mathbf{v}$.

8. *We want to show that* $(a \cdot b) \odot \mathbf{v} = a \odot (b \odot \mathbf{v})$ *for all* $a, v \in \mathbb{R}$. *We see that*

$$(a\,b) \odot \mathbf{v} = \begin{bmatrix} a\,b\,v_1 + a\,b - 1 \\ a\,b\,v_2 + a\,b - 1 \end{bmatrix} \qquad \textit{and}$$

$$\begin{aligned} a \odot (b \odot v) &= a \odot \begin{bmatrix} b\,v_1 + b - 1 \\ b\,v_2 + b - 1 \end{bmatrix} \\ &= \begin{bmatrix} a(b\,v_1 + b - 1) + a - 1 \\ a(b\,v_2 + b - 1) + a - 1 \end{bmatrix} \\ &= \begin{bmatrix} a\,b\,v_1 + a\,b - 1 \\ a\,b\,v_2 + a\,b - 1 \end{bmatrix} \end{aligned}$$

9. *We want to show that* $k \odot (\mathbf{u} \oplus \mathbf{v}) = k \odot \mathbf{u} \oplus k \odot \mathbf{v}$ *for all* $\mathbf{u}, \mathbf{v} \in$ *and* $k \in \mathbb{R}$. *Observe that*

$$\begin{aligned} k \odot (\mathbf{u} \oplus \mathbf{v}) &= k \odot \begin{bmatrix} u_1 + v_1 + 1 \\ u_2 + v_2 + 1 \end{bmatrix} \\ &= \begin{bmatrix} k(u_1 + v_1 + 1) + k - 1 \\ k(u_2 + v_2 + 1) + k - 1 \end{bmatrix} \\ &= \begin{bmatrix} k\,u_1 + k\,v_1 + 2k - 1 \\ k\,u_2 + k\,v_2 + 2k - 1 \end{bmatrix} \qquad \textit{and} \\ k \odot \mathbf{u} \oplus k \odot \mathbf{v} &= k \odot \begin{bmatrix} u1 \\ u_2 \end{bmatrix} \oplus k \odot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\ &= \begin{bmatrix} k\,u_1 + k - 1 \\ k\,u_2 + k - 1 \end{bmatrix} \oplus \begin{bmatrix} k\,v_1 + k - 1 \\ k\,v_2 + k - 1 \end{bmatrix} \\ &= \begin{bmatrix} k\,u_1 + k - 1 + k\,v_1 + k - 1 + 1 \\ k\,u_2 + k - 1 + k\,v_2 + k - 1 + 1 \end{bmatrix} \\ &= \begin{bmatrix} k\,u_1 + k\,v_1 + 2k - 1 \\ k\,u_2 + k\,v_2 + 2k - 1 \end{bmatrix}. \end{aligned}$$

10. *Lastly, we want to show that* $(a + b) \odot \mathbf{v} = (a \odot \mathbf{v}) \oplus (b \odot \mathbf{v})$ *for all* $a, b \in \mathbb{R}$ *and* $\mathbf{v} \in V$. *We*

see that

$$(a+b)\odot\mathbf{v} = \begin{bmatrix}(a+b)v_1+(a+b)-1\\(a+b)v_2+(a+b)-1\end{bmatrix}$$
$$= \begin{bmatrix}a\,v_1+b\,v_1+a+b-1\\a\,v_2+b\,v_2+a+b-1\end{bmatrix} \qquad \textit{and}$$
$$(a\odot\mathbf{v})\oplus(b\odot\mathbf{v}) = \begin{bmatrix}a\,v_1+a-1\\a\,v_2+a-1\end{bmatrix}\oplus\begin{bmatrix}b\,v_1+b-1\\b\,v_2+b-1\end{bmatrix}$$
$$= \begin{bmatrix}(a\,v_1+a-1)+(b\,v_1+b-1)+1\\(a\,v_2+a-1)+(b\,v_2+b-1)+1\end{bmatrix}$$
$$= \begin{bmatrix}a\,v_1+b\,v_1+a+b-1\\a\,v_2+b\,v_2+a+b-1\end{bmatrix}.$$

We now restrict our discussion of vector spaces to Euclidean n—space, or $\mathbb{R}^n$, for positive integers n. Addition in $\mathbb{R}^n$ is given by

$$\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix}\oplus\begin{bmatrix}y_1\\y_2\\\vdots\\y_n\end{bmatrix} = \begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix}+\begin{bmatrix}y_1\\y_2\\\vdots\\y_n\end{bmatrix} = \begin{bmatrix}x_1+y_1\\x_2+y_2\\\vdots\\x_n+y_n\end{bmatrix},$$

and scalar multiplication, for $a\in\mathbb{R}$, is given by

$$a\odot\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix} = a\cdot\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix} = \begin{bmatrix}a\,x_1\\a\,x_2\\\vdots\\a\,x_n\end{bmatrix}.$$

Remark 2.1.1. *From now on, we will use $a\cdot\mathbf{u}$ (or simply $a\mathbf{u}$) to denote scalar multiplication and $\mathbf{u}+\mathbf{v}$ to denote vector addition.*

Example 2.1.3. *Let* $\mathbf{v} = \begin{bmatrix}1\\2\\3\end{bmatrix}$.

Python and Sage take vectors as rows. We must keep this in mind as we work.
To enter $\mathbf{v}$ *as a row vector into Python,*

```
import numpy as np
```

```
v = np.array([1,2,3])
```

To enter $\mathbf{v}$ *as a row vector into SageMath,*

```
sage: v = vector([1,2,3])
```

Now, let $\mathbf{w} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$. *Find* $\mathbf{v}+\mathbf{w}$ *and* $-2\cdot\mathbf{w}$.

```
sage: w = vector([4,5,6])
sage: v + w
(5, 7, 9)


sage: -2*w
(-8, -10, -12)
```

So, $\mathbf{v}+\mathbf{w} = \begin{bmatrix} 5 \\ 7 \\ 9 \end{bmatrix}$ *and* $-2\cdot\mathbf{w} = \begin{bmatrix} -8 \\ -10 \\ -12 \end{bmatrix}$.

Definition 3 (Real Subspace)**.**
If $S \subseteq \mathbb{R}^n$ *is also a vector space with the same addition and scalar multiplication, then* S *is called a* **subspace**.

Theorem 2.1.1. *A nonempty subset of* $\mathbb{R}^n$ *is a subspace if and only if it is closed under addition and scalar multiplication. That is, a nonempty* $S \subset \mathbb{R}^n$ *is a subspace if and only if* $\mathbf{u}+\mathbf{v} \in S$ *and* $a\mathbf{u} \in S$ *for all* $\mathbf{u}, \mathbf{v} \in S$ *and for all* $a \in \mathbb{R}$.

We leave the proof as an exercise for the reader.

Remark 2.1.2. *From now on, we refer to any subspace of* $\mathbb{R}^n$ *as simply a vector space. Likewise, when we say "vector space," it is safe to assume that we are talking about a subspace of* $\mathbb{R}^n$.

Definition 4 (Linear Combination)**.**
Let V *be a vector space, and let* $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_k \in V$. *Any vector of the form* $a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k$, *where* $a_1, a_2, \ldots, a_k \in \mathbb{R}$, *is called a* ***linear combination*** *of* $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_k$.

Definition 5 (Span)**.**
The set of all linear combinations is called the ***span*** *of* $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_k$ *and is denoted by*

$$span\{\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_k\} = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k : a_1, a_2, \ldots a_k \in \mathbb{R}\}.$$

Remark 2.1.3. *span*$\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ *is a subspace of* R^n.

Remark 2.1.4. *If span*$\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\} = V$, *we say*

1. $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ *span* V.
2. $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ *spans* V.
3. V *is spanned by* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$.
4. V *is the linear span of* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$.
5. $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ *is a spanning set for* V.

Definition 6 (Linear Independence and Dependence)**.**
A set of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_k \in V$ *is* ***linearly independent*** *if* $a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k = \mathbf{0}$ *implies that* $a_1 = a_2 = \cdots = a_k = 0$. *The set is* ***linearly dependent*** *if* $a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k = \mathbf{0}$ *for some nonzero scalar* a_i $(1 \leq i \leq k)$.

Figure 2.1 shows two linearly dependent vectors in $\mathbb{R}^2$, namely $\begin{bmatrix}1\\1\end{bmatrix}$ and $\begin{bmatrix}3\\3\end{bmatrix}$.

$$a_1\begin{bmatrix}1\\1\end{bmatrix} + a_2\begin{bmatrix}3\\3\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$
$$\begin{bmatrix}a_1 + 3a_2\\a_1 + 3a_2\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

This gives the system of equations and solution

$$\begin{cases} a_1 + 3a_2 &= 0 \\ a_1 + 3a_2 &= 0 \end{cases} \Longrightarrow a_1 = -3a_2$$

So, there exists nonzero scalars a_1 and a_2 such that $a_1\begin{bmatrix}1\\1\end{bmatrix} + a_2\begin{bmatrix}3\\3\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$. This means that the two vectors are linearly dependent. In other words, these two vectors are linearly dependent because the second is a scalar multiple of the first.

Figure 2.2 shows two linearly independent vectors in $\mathbb{R}^2$, namely $\begin{bmatrix}2\\1\end{bmatrix}$ and $\begin{bmatrix}-1\\3\end{bmatrix}$.

$$a_1\begin{bmatrix}2\\1\end{bmatrix}+a_2\begin{bmatrix}-1\\3\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}$$
$$\begin{bmatrix}2a_1-a_2\\a_1+3a_2\end{bmatrix}$$

This gives the system of equations and solution

$$\begin{cases}2a_1-a_2 &=0\\ a_1+3a_2 &=0\end{cases}\implies 2a_1=a_2 \text{ and } a_1=-3a_2 \implies a_1=a_2=0$$

Since the only solution to the system of equations is $a_1=a_2=0$, these two vectors are linearly independent. Notice that one is not a scalar multiple of the other.

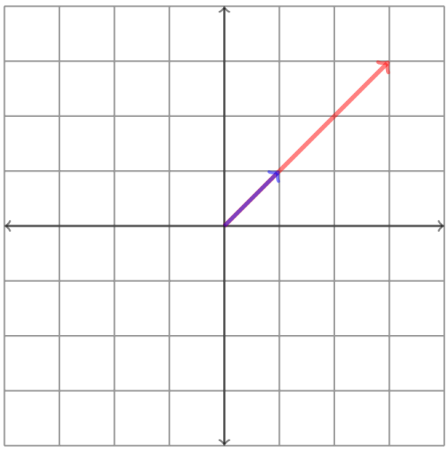


Figure 2.1 linearly dependent vectors in $\mathbb{R}^2$

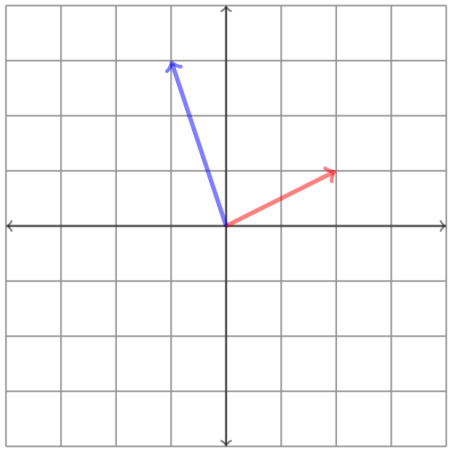


Figure 2.2 linearly independent vectors in $\mathbb{R}^2$

Remark 2.1.5. *In* $\mathbb{R}^n$,

- *The zero vector* $\mathbf{0}$ *is linearly dependent.*
- *A single vector* $\mathbf{v}$ *is linearly independent if and only if* $\mathbf{v}\neq\mathbf{0}$.
- *Two vectors are linearly independent if and only if they are not scalar multiples of one another.*
- $\{\mathbf{v},\mathbf{0}\}$ *is linearly dependent for any vector* $\mathbf{v}$.

Theorem 2.1.2. *Let* $T=\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_n\}\subseteq\mathbb{R}^n$, *and let* $S=\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_n,\mathbf{v}_{n+1}\}\subseteq\mathbb{R}^n$. *That is, let* $T\subseteq S\subseteq\mathbb{R}^n$. *Then,*

1. *If T is linearly dependent, then S is linearly dependent. That is, a superset of a linearly dependent set is linearly dependent. In other words, you cannot make a linearly dependent set independent by adding vectors to it.*

2. *If S is linearly independent, then T is linearly independent. That is, a subset of a linearly independent set is linearly independent. In other words, you cannot make a linearly independent set dependent by removing vectors.*

3. *If T is a spanning set for $\mathbb{R}^n$, then S is a spanning set for $\mathbb{R}^n$.*

Example 2.1.4. *We will show an example of statement 2. Let* $T = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$ *and* $S = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$. *Then* $T \subseteq S \subseteq \mathbb{R}^3$. *For ease of notation, denote the vectors in* S *as* $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. *We can see that* S *is linearly independent, because* $a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + a_3\mathbf{e}_3 = \mathbf{0} \iff a_1 = a_2 = a_3 = 0$. *So,* T *is also linearly independent.*

Remark 2.1.6. *$S = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\} \subseteq \mathbb{R}^n$ is linearly dependent if and only if there is at least one vector $\mathbf{v}_k \in S$ such that $\mathbf{v}_k$ is a linear combination of the other vectors for some $k = 1, 2, \ldots, n$. That is, S is linearly dependent if and only if $\mathbf{v}_k \in span\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{k-1}, \mathbf{v}_{k+1}, \ldots, \mathbf{v}_n\}$ for some $\mathbf{v}_k \in S$.*

Definition 7 (Basis)**.**
A **basis** *for a vector space V is a set of linearly independent vectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ that span V.*

Remark 2.1.7. *We call $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$, as in Example 2.1.4, the standard basis for $\mathbb{R}^n$, where $\mathbf{e}_i$ has a 1 in the ith position and a 0 elsewhere $(1 \leq i \leq n)$. Figure 2.3 shows the standard basis for $\mathbb{R}^2$ in red.*

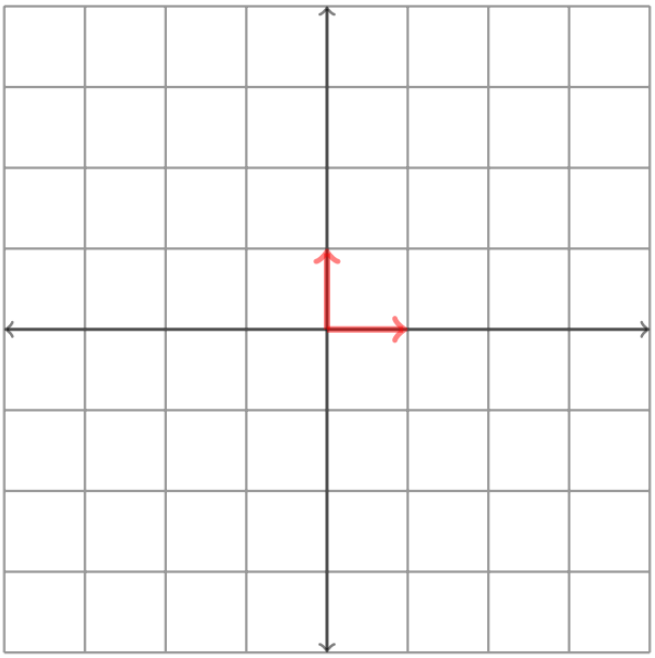


Figure 2.3 $\mathbf{e}_1$ and $\mathbf{e}_2$ form a basis for $\mathbb{R}^2$.

Every real vector space has a basis. We can represent a vector space as a linear span of a set of vectors, which can be linearly independent. In that way, we can now say that a real vector space is $\{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_n\mathbf{v}_n : a_1, a_2, \ldots, a_n \in \mathbb{R}\}$ for basis vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. This notion of a vector space is important moving forward, as we begin to discuss lattices.

Theorem 2.1.3. *Any two bases of the same vector space have the same number of vectors.*

Definition 8 (Dimension)**.**
The **dimension** *of a vector space V, denoted* $\dim(V)$*, is the number of basis vectors.*

Example 2.1.5. $\dim(\mathbb{R}^n) = n$.

Example 2.1.6. *Let* $V = span\left\{\begin{bmatrix}1\\-1\\2\end{bmatrix}, \begin{bmatrix}2\\1\\-3\end{bmatrix}, \begin{bmatrix}3\\0\\-1\end{bmatrix}\right\}$. *Then* $\dim(V) = 2$*, since the third vector is the sum of the first two.*

Proposition 2.1.1. $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ *is a basis for vector space V if and only if every* $\mathbf{w} \in V$ *has a unique representation as a linear combination of* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$.

2.2 Matrix Algebra

We begin with some basic matrix notation that we will carry throughout the course. We use capital letters to denote matrices. As you may have noticed, we use boldface lowercase letters for vectors and non-boldface lowercase letters for scalars.

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

- $A \in \mathbb{R}^{m \times n}$
- A is an $m \times n$ matrix with m rows and n columns.
- A has $m \cdot n$ elements.
- a_{ij} is in row i and column j.
- A has m row vectors and n column vectors.
- Row i of A is $\begin{bmatrix} a_{i1} & a_{i2} & \dots & a_{in} \end{bmatrix} \in \mathbb{R}^{1 \times n}$ for $1 \leq i \leq m$.
- Column j of A is $\mathbf{a}_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix} \in \mathbb{R}^{m \times 1} = \mathbb{R}^m$ for $1 \leq j \leq n$. This course will default to using vectors as columns, though other texts may use row vectors for cryptographic applications. We will be sure to specify.
- **"Column View"** of A: $A = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix}$, where $\mathbf{a}_j \in \mathbb{R}^m$ for $1 \leq j \leq n$.

Example 2.2.1 (Matrices in Python and Sage)**.**

Let $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$.

To enter A into Python,

```
import numpy as np

A = np.array([[1,2,3],[4,5,6],[7,8,9]])
```

To enter A into SageMath,

```
sage: A = Matrix([[1,2,3],[4,5,6],[7,8,9]])
```

To extract, for example, the first column or the first row of A in Sage, we can do the following:

```
sage: A[:,0]
[1]
[4]
[7]

sage: A[0,:]
[1 2 3]
```

Note that Python and Sage both begin their indexing at 0*, not at* 1*.*

Definition 9 (Equality of Matrices)**.**
Two matrices are **equal** *if they are the same size and have all corresponding equal entries. That is,* $A = B$ *if* $A, B \in \mathbb{R}^{m\times n}$ *with* $a_{ij} = b_{ij}$ *for all* $1 \le i \le m$ *and* $1 \le j \le n$*.*

Operations with Matrices: Let $A, B \in \mathbb{R}^{m\times n}$ and $\alpha \in \mathbb{R}$.

- **Addition:** $C = A + B$ has entries $c_{ij} = a_{ij} + b_{ij}$, $1 \le i \le m$, $1 \le j \le n$.
  - Column View: $C = \begin{bmatrix} \mathbf{a}_1 + \mathbf{b}_1 & \mathbf{a}_2 + \mathbf{b}_2 & \dots & \mathbf{a}_n + \mathbf{b}_n \end{bmatrix}$.
  - Zero Matrix: $\mathbf{0}_{m\times n} \in \mathbb{R}^{m\times n}$ has all entries equal to 0, and $A + \mathbf{0}_{m\times n} = \mathbf{0}_{m\times n} + A = A$.
- **Scalar Multiplication:** $M = \alpha A$ has elements $m_{ij} = \alpha a_{ij}$, $1 \le i \le m$, $1 \le j \le n$ for $\alpha \in \mathbb{R}$.
  - Column View: $M = \begin{bmatrix} \alpha\mathbf{a}_1 & \alpha\mathbf{a}_2 & \dots & \alpha\mathbf{a}_n \end{bmatrix}$.
  - Zero Matrix: $0 \cdot A = A \cdot 0 = \mathbf{0}_{m\times n}$.

Example 2.2.2 (Matrix Algebra)**.**
Let $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ *and* $B = \begin{bmatrix} 1 & -3 & 5 \\ -2 & 4 & -6 \end{bmatrix}$*. Find* $A - 2B$*.*

Column View: Denote $A=\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3\end{bmatrix}$ *and* $B=\begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3\end{bmatrix}$. *Then*

$$\begin{aligned}A-2B&=\begin{bmatrix}\mathbf{a}_1-2\mathbf{b}_1 & \mathbf{a}_2-2\mathbf{b}_2 & \mathbf{a}_3-2\mathbf{b}_3\end{bmatrix}\\&=\left[\begin{bmatrix}1\\4\end{bmatrix}-2\cdot\begin{bmatrix}1\\-2\end{bmatrix}\quad\begin{bmatrix}2\\5\end{bmatrix}-2\cdot\begin{bmatrix}-3\\4\end{bmatrix}\quad\begin{bmatrix}3\\6\end{bmatrix}-2\cdot\begin{bmatrix}5\\-6\end{bmatrix}\right]\\&=\begin{bmatrix}-1&8&-7\\8&-3&18\end{bmatrix}\end{aligned}$$

Element View: Denote $A=\begin{bmatrix}a_{11}&a_{12}&a_{13}\\a_{21}&a_{22}&a_{23}\end{bmatrix}$ *and* $B=\begin{bmatrix}b_{11}&b_{12}&b_{13}\\b_{21}&b_{22}&b_{23}\end{bmatrix}$. *Then*

$$\begin{aligned}A-2B&=\begin{bmatrix}a_{11}-2\cdot b_{11}&a_{12}-2\cdot b_{12}&a_{13}-2\cdot b_{13}\\a_{21}-2\cdot b_{21}&a_{22}-2\cdot b_{22}&a_{23}-2\cdot b_{23}\end{bmatrix}\\&=\begin{bmatrix}1-2(1)&2-2(-3)&3-2(5)\\4-2(-2)&5-2(-4)&6-2(-6)\end{bmatrix}\\&=\begin{bmatrix}-1&8&-7\\8&-3&18\end{bmatrix}\end{aligned}$$

We can compute this in Sage as follows:

```
sage: A = Matrix([[1,2,3], [4,5,6]])
sage: B = Matrix([[1,-3,5], [-2,4,-6]])
sage: A - 2*B
[-1  8 -7]
[ 8 -3 18]
```

It is done very similarly in Python:

```
import numpy as np

A = np.array([[1,2,3],[4,5,6]])
B = np.array([[1,-3,5],[-2,4,-6]])

print(A-2*B)

[[-1  8 -7]
 [ 8 -3 18]]
```

Properties of Matrix Operations: For $A, B, C \in \mathbb{R}^{m\times n}$ and $\alpha, \beta \in \mathbb{R}$, we have the following:

- *(Commutativity of Addition)* $A + B = B + A$
- *(Associativity of Addition)* $A + (B + C) = (A + B) + C$
- *(Zero Matrix or Additive Identity)* $A + \mathbf{0}_{m\times n} = \mathbf{0}_{m\times n} + A = A$
- *(Commutativity of Scalar Multiplication)* $\alpha \cdot A = A \cdot \alpha$
- *(Associativity of Scalar Multiplication)* $\alpha(\beta \cdot A) = (\alpha\beta) \cdot A$
- *(Distributivity)* $\alpha(A + B) = \alpha \cdot A + \alpha \cdot B$ and $(\alpha + \beta) \cdot A = \alpha \cdot A + \beta \cdot A$

Remark 2.2.1. $\mathbb{R}^{m\times n}$ *with element-wise addition and scalar multiplication is a vector space.*

Matrix-Vector Multiplication: If $A \in \mathbb{R}^{m\times n}, \mathbf{x} \in \mathbb{R}^n$, then $A\mathbf{x} \in \mathbb{R}^m$, and we can write $A\mathbf{x}$ as a linear combination of the columns of A with coefficients from $\mathbf{x}$:

$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix}.$$

Column View: For $A = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix}$,

$$A\mathbf{x} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_n\mathbf{a}_n.$$

Example 2.2.3 (Matrix-Vector Multiplication)**.**
Let $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ *and* $\mathbf{x} \in \mathbb{R}^3$. *Then*

$$A\mathbf{x} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + x_2 \begin{bmatrix} 2 \\ 5 \end{bmatrix} + x_3 \begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} x_1 + 2x_2 + 3x_3 \\ 4x_1 + 5x_2 + 6x_3 \end{bmatrix} \in \mathbb{R}^{2\times 1} = \mathbb{R}^2.$$

To compute this in Sage,

```
sage: var('x1','x2','x3')
sage: A = Matrix([[1,2,3],[4,5,6]])
```

```
sage: x = vector([x1,x2,x3])
sage: A*x
(x1 + 2*x2 + 3*x3, 4*x1 + 5*x2 + 6*x3)
```

Example 2.2.4 (Inner Product)**.**
Let $A=\begin{bmatrix}4 & 3 & 2 & 1\end{bmatrix}$ *and* $\mathbf{x}\in\mathbb{R}^4$. *Then*

$$A\mathbf{x}=\begin{bmatrix}4 & 3 & 2 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}=x_1\begin{bmatrix}4\end{bmatrix}+x_2\begin{bmatrix}3\end{bmatrix}+x_3\begin{bmatrix}2\end{bmatrix}+x_4\begin{bmatrix}1\end{bmatrix}=4x_1+3x_2+2x_3+1x_4\in\mathbb{R}.$$

Example 2.2.4 is an example of what is known as an **inner product**. Specifically, this is a dot product. Note that row $\cdot$ column = scalar.
Try Example 2.2.4 in SageMath. Don't forget to define your variables.

Matrix Multiplication: If $A\in\mathbb{R}^{m\times n}$ and $B\in\mathbb{R}^{n\times k}$, then $AB\in\mathbb{R}^{m\times k}$. Note that the number of columns in A must be the same as the number of rows in B in order for the matrix multiplication to be defined.

Column View: Denote $B=\begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_k\end{bmatrix}$, $\mathbf{b}_j\in R^n, 1\le j\le k$. Then

$$AB=A\begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_k\end{bmatrix}=\begin{bmatrix}A\mathbf{b}_1 & A\mathbf{b}_2 & \dots & A\mathbf{b}_k\end{bmatrix}.$$

Column j of AB is the matrix-vector product of A with column j of B.

Element View: The (i,j) element of AB is the inner product of row i from matrix A and column j from matrix B. That is,

$$(AB)_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+\cdots+a_{in}b_{nj}=\sum_{p=1}^{n}a_{ip}b_{pj},\quad \text{for } 1\le i\le m, 1\le j\le k.$$

Example 2.2.5 (Matrix Multiplication)**.**
Let $A=\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$ *and* $B=\begin{bmatrix}1 & 0 & -1\\1 & 1 & 1\\-2 & 3 & 1\end{bmatrix}$. *Find* AB.

Column View: Denote $B = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix}$. *Then*

$$AB = \begin{bmatrix} A\mathbf{b}_1 & A\mathbf{b}_2 & A\mathbf{b}_3 \end{bmatrix}, \quad \textit{where}$$

$$A\mathbf{b}_1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix} = 1\begin{bmatrix} 1 \\ 4 \end{bmatrix} + 1\begin{bmatrix} 2 \\ 5 \end{bmatrix} + -2\begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} -3 \\ -3 \end{bmatrix},$$

$$A\mathbf{b}_2 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix} = 0\begin{bmatrix} 1 \\ 4 \end{bmatrix} + 1\begin{bmatrix} 2 \\ 5 \end{bmatrix} + 3\begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} 11 \\ 23 \end{bmatrix}, \quad \textit{and}$$

$$A\mathbf{b}_3 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix} = -1\begin{bmatrix} 1 \\ 4 \end{bmatrix} + 1\begin{bmatrix} 2 \\ 5 \end{bmatrix} + 1\begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}. \quad \textit{So,}$$

$$AB = \begin{bmatrix} -3 & 11 & 4 \\ -3 & 23 & 7 \end{bmatrix}.$$

Element View: $AB = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 1 & 1 \\ -2 & 3 & 1 \end{bmatrix}$, *with the* (i, j) *element of* AB *as the dot product of row* i *from matrix* A *with column* j *from matrix* B. *For example,*

$$(AB)_{11} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix} = 1(1) + 1(2) + -2(3) = -3, \quad \textit{and}$$

$$(AB)_{12} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix} = 0(1) + 1(2) + 3(3) = 11,$$

and so on. Thus, just as before,

$$AB = \begin{bmatrix} -3 & 11 & 4 \\ -3 & 23 & 7 \end{bmatrix}.$$

If we wanted to do this in Python, we would execute the code below:

```
import numpy as np
```

```
A = np.array([[1,2,3],[4,5,6]])
B = np.array([[1,0,-1],[1,1,1],[-2,3,1]])

print(np.dot(A,B))

[[-3 11  4]
 [-3 23  7]]
```

We can, of course, compute this in SageMath as well:

```
sage: A = Matrix([[1,2,3],[4,5,6]])
sage: B = Matrix([[1,0,-1],[1,1,1],[-2,3,1]])
sage: A*B
[-3 11  4]
[-3 23  7]
```

Example 2.2.6 (Outer Product)**.**

Let $A = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 2 \end{bmatrix}$ *and* $B = \begin{bmatrix} 2 & 3 & 4 \end{bmatrix}$. *Then*

$$AB = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 2 \end{bmatrix} \begin{bmatrix} 2 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 4 \\ -2 & -3 & -4 \\ 2 & 3 & 4 \\ -4 & -6 & -8 \end{bmatrix}.$$

To compute this in Sage:

```
sage: A = Matrix([[1],[-1],[1],[2]])
sage: B = Matrix([[2,3,4]])
sage: A*B
[ 2  3  4]
[-2 -3 -4]
[ 2  3  4]
[ 4  6  8]
```

Example 2.2.6 is an example of an **outer product**. Note that column · row = matrix. Contrast this with the inner product in Example 2.2.4.

Properties of Matrix Multiplication:

- *(Associativity)* For $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times k}$, and $C \in \mathbb{R}^{k\times l}$,

$$A = A\cdot(B\cdot C) = (A\cdot B)\cdot C \in \mathbb{R}^{m\times l}.$$

- *(Distributivity)* For $A, B \in R^{m\times n}$ and $C, D \in \mathbb{R}^{n\times k}$,

$$A\cdot(C+D) = AC + AD \in \mathbb{R}^{m\times k}$$

$$(A+B)\cdot C = AC + BC \in \mathbb{R}^{m\times k}.$$

- *(Scalar Multiplication)* For $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times k}$, and $\alpha \in \mathbb{R}$,

$$(\alpha A)\cdot B = \alpha(A\cdot B) = A\cdot(\alpha B) \in \mathbb{R}^{m\times k}.$$

Remark 2.2.2. *In general, matrix multiplication is* **not** *commutative. That is $AB \neq BA$ unless both A and B are 1×1 matrices.*

Example 2.2.7 (Matrix Multiplication is not Commutative)**.**
Let $A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ *and* $B = \begin{bmatrix} 2 & -1 \end{bmatrix}$.
Then AB is not defined, but $BA = \begin{bmatrix} 2 & -1 \end{bmatrix}\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 0 & 2 \end{bmatrix}$.

Example 2.2.8 (Matrix Multiplication is not Commutative)**.**
Let Let $A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$ *and* $B = \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}$.
Then $AB = \begin{bmatrix} -1 & 2 \\ 1 & -1 \end{bmatrix}$, *but* $BA = \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix}$.

Identity Matrices: The $n\times n$ identity matrix, denoted I_n, is a square matrix with 1s on the main diagonal and 0s elsewhere. We denote the columns of the identity matrix as $\mathbf{e}_i$ for $1 \le i \le n$, where $\mathbf{e}_i$ has a 1 in the ith position and 0s elsewhere.

Example 2.2.9.

$$I_1 = \begin{bmatrix} 1 \end{bmatrix} \in \mathbb{R}^{1\times 1}, \qquad I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \qquad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

$$I_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_n \end{bmatrix} \in \mathbb{R}^{n\times n}$$

To generate, for example, a 5×5 *identity matrix in Sage,*

```
sage: I5 = identity_matrix(5)
sage: I5
[1 0 0 0 0]
[0 1 0 0 0]
[0 0 1 0 0]
[0 0 0 1 0]
[0 0 0 0 1]
```

Remark 2.2.3. *For every* $A \in \mathbb{R}^{n\times n}$, $A \cdot I_n = I_n \cdot A = A$. I_n *is the multiplicative identity in* $\mathbb{R}^{n\times n}$.

Transpose of a Matrix: Let $A \in \mathbb{R}^{m\times n}$. Then $A^T \in \mathbb{R}^{n\times m}$ changes columns into rows and rows into columns. That is, if $(A)_{ij} = a_{ij}$, then $(A^T)_{ij} = a_{ji}$.

Example 2.2.10.

$$For\ A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \in \mathbb{R}^{2\times 3}, \quad A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \in \mathbb{R}^{3\times 2}.$$

We can also find this two ways in Sage:

```
sage: A = Matrix([[1,2,3],[4,5,6]])
sage: A.transpose()
[1 4]
[2 5]
[3 6]
```

```
sage: A.T
[1 4]
[2 5]
[3 6]
```

Or, we could use Python:

```
import numpy as np

A = np.array([[1,2,3],[4,5,6]])

print(A.T)

[[1 4]
 [2 5]
 [3 6]]
```

Properties of Transposes: Let $A, B \in \mathbb{R}^{m\times n}$. Then

- $(A^T)^T = A$
- $(A+B)^T = A^T + B^T$
- $(\alpha A)^T = \alpha A^T$ for all $\alpha \in \mathbb{R}$
- $(AB)^T = B^T A^T$

Definition 10 (Symmetric and Skew Symmetric Matrices)**.**
Let $A \in \mathbb{R}^{n\times n}$. If $A^T = A$, then A is **symmetric**. *If $A^T = -A$, then A is* **skew-symmetric**.

Example 2.2.11 (Symmetric and Skew Symmetric Matrices)**.**
$A = \begin{bmatrix} 0 & 4 & 1 \\ 4 & -1 & 5 \\ 1 & 5 & 3 \end{bmatrix}$ *is symmetric since* $A^T = \begin{bmatrix} 0 & 4 & 1 \\ 4 & -1 & 5 \\ 1 & 5 & 3 \end{bmatrix} = A$, *and*
$B = \begin{bmatrix} 0 & -2 & -1 \\ 2 & 0 & 3 \\ 1 & -3 & 0 \end{bmatrix}$ *is skew-symmetric since* $B^T = \begin{bmatrix} 0 & 2 & 1 \\ -2 & 0 & -3 \\ -1 & 3 & 0 \end{bmatrix} = -B$.

2.3 Systems of Linear Equations

Definition 11 (System of Equations)**.**

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & = b_2 \\ & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & = b_m \end{cases}$$ *is a* **system of m linear equations in n variables**.

The **coefficient matrix** *of this system is* $\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m\times n}$, *where* a_{ij} *is the coefficient of* x_j *in equation* i.

The **augmented matrix** *of this system is* $\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} & b_m \end{array}\right] \in \mathbb{R}^{m\times(n+1)}$.

Remark 2.3.1.

Solving the system of linear equations given by $\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & = b_2 \\ & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & = b_m \end{cases}$ *is equivalent to the following statement:*

Given $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$ *and* $\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$, *find* $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ *so that* $A\mathbf{x} = \mathbf{b}$.

Example 2.3.1 (Coefficient and Augmented Matrices)**.**

The system of linear equations $\begin{cases} x_1 + x_2 - 3x_3 & = 3 \\ -2x_1 - x_2 & = -4 \\ 4x_1 + 2x_2 + 3x_3 & = 7 \end{cases}$ *has coefficient matrix* $\begin{bmatrix} 1 & 1 & -3 \\ -2 & -1 & 0 \\ 4 & 2 & 3 \end{bmatrix}$

and augmented matrix $\left[\begin{array}{ccc|c} 1 & 1 & -3 & 3 \\ -2 & -1 & 0 & 4 \\ 4 & 2 & 3 & 7 \end{array}\right]$. *We will use the augmented matrix to solve for* x_1, x_2, x_3.

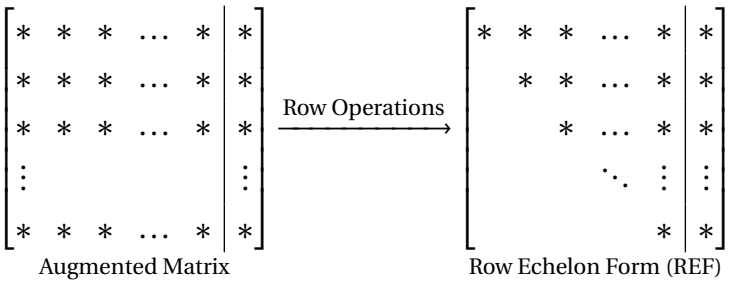
To enter an augmented matrix in Sage, simply use the augment command. Just be sure to enter **b** *as a matrix instead of a row vector.*

```
sage: A = Matrix([[1,1,-3],[-2,-1,0],[4,2,3]])
sage: b = Matrix([[3],[4],[7]])
sage: A.augment(b)
[ 1  1 -3  3]
[-2 -1  0  4]
[ 4  2  3  7]
```

Given a linear system $A\mathbf{x} = \mathbf{b}$, we want to find $\mathbf{x}$ that makes the statement true. As we have seen, a system of linear equations can be represented by an augmented matrix. We will use what is known as Gaussian Elimination on our matrices to solve systems of equations.

Solving Linear Systems To solve a system of linear equations, form the augmented matrix. Then, do the following:

1. Use row operations to reduce the augmented matrix to row echelon form.



2. Solve for unknowns from the bottom up (back substitution).

Example 2.3.2. *Solve the following system. While illustrating row operations, we will use* R_i *to denote the i th row. We will define valid row operations in a moment. This example shows how to use row operations to get a matrix into REF, or upper triangular form, and how to use back substitution to solve a system.*

$$\begin{cases} x_1 + x_2 - 3x_3 & = 3 \\ -2x_1 - x_2 & = -4 \\ 4x_1 + 2x_2 + 3x_3 & = 7 \end{cases}$$

Augmented Matrix (and row operations):

$$\left[\begin{array}{ccc|c} 1 & 1 & -3 & 3 \\ -2 & -1 & 0 & 4 \\ 4 & 2 & 3 & 7 \end{array}\right] \xrightarrow[R_3=R_3-4R_1]{R_2=R_2+2R_1} \left[\begin{array}{ccc|c} 1 & 1 & -3 & 3 \\ 0 & 1 & -6 & 2 \\ 0 & -2 & 15 & -5 \end{array}\right] \xrightarrow{R_3=R_3+2R_2} \left[\begin{array}{ccc|c} 1 & 1 & -3 & 3 \\ 0 & 1 & -6 & 2 \\ 0 & 0 & 3 & -1 \end{array}\right]$$

Back Substitute:

$$\begin{cases} x_1 + x_2 - 3x_3 &= 3 \\ x_2 - 6x_3 &= 2 \\ 3x_3 &= -1 \end{cases} \underset{eq'n\ 3}{\Longrightarrow} x_3 = \frac{-1}{3} \underset{eq'n\ 2}{\Longrightarrow} x_2 = 0 \underset{eq'n\ 1}{\Longrightarrow} x_1 = 2$$

This system has solution $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ \frac{-1}{3} \end{bmatrix}$.

Definition 12 (Consistent System)**.**
A system that has a solution is called a **consistent system**. *A consistent system of linear equations has either one or infinitely many solutions.*

What does the solution to the previous example problem tell us?
"Row View:" All three linear equations can be satisfied at the same time, namely when $x_1 = 2$, $x_2 = 0$, and $x_3 = \frac{-1}{3}$.
"Column View:" $x_1 \begin{bmatrix} 1 \\ -2 \\ 4 \end{bmatrix} + x_2 \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} + x_3 \begin{bmatrix} -3 \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \\ -4 \\ 7 \end{bmatrix}$ has a solution for x_1, x_2, x_3, so $\begin{bmatrix} 3 \\ -4 \\ 7 \end{bmatrix} \in$ span $\left\{ \begin{bmatrix} 1 \\ -2 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}, \begin{bmatrix} -3 \\ 0 \\ 3 \end{bmatrix} \right\}$.

Valid Row Operations ("Elementary Row Operations")
The following row operations, as seen in Example 2.3.2, on the augmented matrix of a linear system will not change the solution of the system:

1. Exchanging two rows ($E_i \longleftrightarrow E_j$)
2. Scaling a row ($E_i = \alpha E_i$ for $0 \neq \alpha \in \mathbb{R}$)
3. Adding a multiple of one row to another ($E_i = E_i + \alpha E_j$ for $\alpha \in \mathbb{R}$)

Definition 13 (REF)**.**
A matrix is in **row echelon form** *(REF) if*

1. *the leading nonzero element of a row is to the left of all other lower leading nonzeros,*
2. *elements below a leading nonzero entry are zero, and*
3. *zero rows are at the bottom*

Example 2.3.3 (REF Examples and Non-examples)**.**
The following coefficient matrices are in REF:

$$\begin{bmatrix} 3 & 1 & 1 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 3 & 0 & 1 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The following coefficient matrices are not in REF:

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 3 & 0 & 1 \\ 2 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The first matrix has elements below the leading nonzero entry in row 1*. The second matrix has a leading nonzero element in row* 2 *that is to the left of the leading nonzero element in row* 1*. The third matrix has a row of all zeros above a nonzero row.*

To solve the system in Example 2.3.2 in Sage, simply create the augmented matrix, and use the rref() command. This puts the matrix in **Row Reduced Echelon Form**, which is REF plus two additional conditions: 1s on the diagonal and 0s above and below any leading 1s. Then, just read off the solution as we did with the equations above.

```
sage: A = Matrix([[1,1,-3,3],[-2,-1,0,-4],[4,2,3,7]])
sage: A.rref()
[   1    0    0    2]
[   0    1    0    0]
[   0    0    1 -1/3]
```

This tells us that $x_1 = 2$, $x_2 = 0$, and $x_3 = \frac{-1}{3}$.

Alternatively, you can define two matrices and use the augment command and then use rref():

```
sage: A = Matrix([[1,1,-3],[-2,-1,0],[4,2,3]])
sage: B = Matrix([[3],[-4],[7]])
sage: (A.augment(B)).rref()
[   1    0    0    2]
[   0    1    0    0]
[   0    0    1 -1/3]
```

Example 2.3.4 (System with No Solution)**.**
Solve the following system of linear equations.

$$\begin{cases} x+y+z & =4 \\ 3x-y-z & =2 \\ x+3y+3z & =8 \end{cases}$$

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 4 \\ 3 & -1 & -1 & 2 \\ 1 & 3 & 3 & 8 \end{array}\right] \xrightarrow[R_3=R_3-R_1]{R_2=R_2-3R_1} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 4 \\ 0 & -4 & -4 & -10 \\ 0 & 2 & 2 & 4 \end{array}\right] \xrightarrow{R_3=R_3+\frac{1}{2}R_2} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 4 \\ 0 & -4 & -4 & -10 \\ 0 & 0 & 0 & -1 \end{array}\right]$$

$$\begin{cases} x+ \ y+ \ z & =4 \\ -4y-4z & =10 \\ 0 & =-1 \end{cases}$$

The third equation is a false statement. So, this system has no solution.

Definition 14 (Inconsistent System)**.**
A system that has no solution is called an **inconsistent system**.

What does the solution to the previous example problem tell us?
"Row View:" Not all equations can be satisfied at the same time.
"Column View:" $\mathbf{b} \notin \text{span}\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$, where $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are the columns of the coefficient matrix and $\mathbf{b}$ is the right hand side of the augmented matrix.

Example 2.3.5 (System with Infinitely Many Solutions)**.**
Solve the following system of linear equations using Gaussian Elimination.

$$\begin{cases} x_1-x_2+x_3 & =5 \\ x_1-x_2 & =2 \\ 2x_1-2x_2+x_3 & =7 \end{cases}$$

$$\left[\begin{array}{ccc|c} 1 & -1 & 1 & 5 \\ 1 & -1 & 0 & 2 \\ 2 & -2 & 1 & 7 \end{array}\right] \xrightarrow{REF} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 5 \\ 0 & 0 & -1 & 3 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$\begin{cases} x_1 - x_2 + x_3 = 5 \\ -x_3 = -3 \end{cases}$$

This gives that $x_3 = 3$. Back substituting $x_3 = 3$ into the first equation and solving for x_1 gives $x_1 = x_2 + 2$. So, there are infinitely many solutions (i.e. this system is consistent), each dependent on our free choice of x_2. For example, if we choose $x_2 = 1$, then $x_1 = 3$ and $x_3 = 3$.

Definition 15 (Free and Leading Variables)**.**
In our previous example (Example 2.3.5), x_2 is a **free variable***; there is no leading entry in the x_2 column of the REF of a consistent system.*
x_1 and x_3 are **leading variables***; there is a leading nonzero entry in the x_1 and x_3 columns of the REF.*
If there is a free variable, we assign it a **parameter** *when we write the solution. In Example 2.3.5, let $x_2 = t \in \mathbb{R}$. Then, writing our solution as a vector gives*

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} t+2 \\ t \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} t + \begin{bmatrix} 2 \\ 0 \\ 3 \end{bmatrix} \text{ for all } t \in \mathbb{R}$$

.

Remark 2.3.2. *Number of free variables + Number of leading variables = Number of variables.*

Remark 2.3.3. *The only possible outcomes of solving a linear system are*

- *one unique solution (example 2.3.2),*
- *no solution (example 2.3.4), or*
- *infinitely many solutions (example 2.3.5).*

Definition 16 (Rank)**.**
The number of nonzero rows (i.e. the number of leading variables) of the REF of a matrix is called the **rank** *of the matrix.*

Example 2.3.6 (Rank)**.**
The matrix $\begin{bmatrix} 1 & 0 & 3 \\ 0 & 2 & 4 \\ 0 & 0 & 0 \end{bmatrix}$ *has rank* 2.

To find the rank of this matrix with Sage, simply use the rank() command.

```
sage: A = Matrix([[1,0,3],[0,2,4],[0,0,0]])
sage: rank(A)
2
```

While using Sage to give us the rank in this example is very simple, doing so is a beneficial tool when you have something like a 300×400 *matrix and need to know the rank. It would be time consuming to row reduce that and count the leading nonzero entries!*

Now, let us see how systems of equations can connect to the idea of rank. Consider $\left[\begin{array}{ccc|c} 1 & 0 & 5 & 2 \\ 0 & 1 & 7 & -1 \\ 0 & 0 & 0 & 1 \end{array}\right]$.
Let A be the coefficient matrix and let $\overline{A}$ *be the augmented matrix. Then* $rank(A) = 2$ *and* $rank(\overline{A}) = 3$. *Note that this system is inconsistent.*

Consider $\left[\begin{array}{ccc|c} 1 & 0 & 3 & 1 \\ 0 & 1 & 2 & -5 \\ 0 & 0 & 0 & 0 \end{array}\right]$ *with the same description for A and* $\overline{A}$. *Then* $rank(A) = rank(\overline{A}) = 2$.
Note that this system is consistent. It has infinitely many solutions since x_3 *is a free variable.*

Theorem 2.3.1. *Let* $\overline{A}$ *be the augmented matrix of a linear system, and let A be its corresponding coefficient matrix, as above. Then*

1. *the system is consistent if and only if* $rank(A) = rank(\overline{A})$ *with*
   - *unique solution if and only if* $rank(A) =$ *number of variables, and*
   - *infinitely many solutions if and only if* $rank(A) <$ *number of variables.*
2. *the system is inconsistent if and only if* $rank(A) \neq rank(\overline{A})$.

2.4 Matrix Inverses

Definition 17 (Invertible Matrix)**.**
$A \in \mathbb{R}^{n \times n}$ *is said to be* **invertible** *(or* **nonsingular***) if there exists* $B \in \mathbb{R}^{n \times n}$ *such that* $AB = BA = I_n$.

If $A \in \mathbb{R}^{n\times n}$ is not invertible, A is called **singular**.

Theorem 2.4.1. *Let $A \in \mathbb{R}^{n\times n}$. If A^{-1} exists, it is unique.*

Proof. Let $A \in \mathbb{R}^{n\times n}$, and let $B, C \in \mathbb{R}^{n\times n}$ be such that $AB = BA = I_n$ and $AC = CA = I_n$. Then

$$\begin{aligned} B &= B \cdot I_n \\ &= B(AC) && \text{(by assumption that } I_n = AC) \\ &= (BA)C && \text{(by associativity of matrix multiplication)} \\ &= I_n \cdot C && \text{(by assumption that } BA = I_n) \\ &= C \end{aligned}$$

We have shown that, if two matrices satisfy the definition of matrix inverse, then they must be equal. So, *the* inverse of $A \in \mathbb{R}^{n\times n}$is unique if it exists. □

In the case where $AB = BA + I_n$, B is called **the inverse** of A, denoted A^{-1}. That is, $AA^{-1} = A^{-1}A = I_n$.

Note that only a square matrix can have an inverse, as a non-square matrix cannot be row reduced to the identity matrix.

So, how do we find the inverse of an $n \times n$ matrix, if it exists?

Recall: Every linear system can be written as a matrix-vector equation $A\mathbf{x} = \mathbf{b}$, where $A \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^m$. Take $m = n$ (square) here.

Example 2.4.1. *($n = 3$)*
For $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$, we want to find $B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$ such that $AB = BA = I_3$. We will find B such that $AB = I_3$, and leave the checking that $BA = I_3$ up to the reader. So, let us find B such that

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Taking the column view, we can write $B = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix}$ and $I_3 = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{bmatrix}$. Then, finding B amounts to finding $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \in \mathbb{R}^3$ such that

$$A\mathbf{b}_1 = \mathbf{e}_1, \qquad A\mathbf{b}_2 = \mathbf{e}_2, \qquad A\mathbf{b}_3 = \mathbf{e}_3.$$

Setting these up as systems of equations and writing them as augmented matrices, we have that

$$\left[A \;\middle|\; \mathbf{e}_1\right], \qquad \left[A \;\middle|\; \mathbf{e}_2\right], \qquad \left[A \;\middle|\; \mathbf{e}_3\right],$$

which gives us the solutions

$$\mathbf{b}_1, \qquad \mathbf{b}_2, \qquad \mathbf{b}_3,$$

respectively. This tells us that $A^{-1} = B = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3\end{bmatrix}$. *For ease, we can write the three systems above in one augmented matrix*

$$\left[A \;\middle|\; I_3\right]$$

and row reduce to row reduced echelon form until we have

$$\left[I_3 \;\middle|\; B\right]$$

In general, to find A^{-1}, *if it exists, of* $A \in \mathbb{R}^{n\times n}$, *we do the following:*

$$\left[A \;\middle|\; I_n\right] \xrightarrow{Row\ Reduce} \left[I_n \;\middle|\; A^{-1}\right]$$

Example 2.4.2 (Finding a Matrix Inverse)**.**

Let $A = \begin{bmatrix} 1 & 1 & -2 \\ -1 & 2 & 0 \\ 0 & -1 & 1 \end{bmatrix}$. *Find* A^{-1}, *if it exists.*

$$\left[\begin{array}{ccc|ccc} 1 & 1 & -2 & 1 & 0 & 0 \\ -1 & 2 & 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_2=R_2+R_1} \left[\begin{array}{ccc|ccc} 1 & 1 & -2 & 1 & 0 & 0 \\ 0 & 3 & -2 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_2\leftrightarrow R_3}$$

$$\left[\begin{array}{ccc|ccc} 1 & 1 & -2 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 \\ 0 & 3 & -2 & 1 & 1 & 0 \end{array}\right] \xrightarrow{R_3=R_3+3R_2} \left[\begin{array}{ccc|ccc} 1 & 1 & -2 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 3 \end{array}\right] \xrightarrow{R_2=-R_2} \left[\begin{array}{ccc|ccc} 1 & 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & 1 & 3 \end{array}\right]$$

$$\xrightarrow{R_1=R_1-R_2} \left[\begin{array}{ccc|ccc} 1 & 0 & -1 & 1 & 0 & 1 \\ 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & 1 & 3 \end{array}\right] \xrightarrow[R_2=R_2+R_3]{R_1=R_1+R_3} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 2 & 1 & 4 \\ 0 & 1 & 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 & 1 & 3 \end{array}\right]$$

This gives us that $A^{-1} = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 1 & 2 \\ 1 & 1 & 3 \end{bmatrix}$. *You should check that* $A \cdot A^{-1} = I_3$ *and* $A^{-1} \cdot A = I_3$.

Example 2.4.3 (Matrix Inverses in Python and Sage)**.**
Of course, we can use a computer to find the inverse of a matrix. In SageMath, one of the ways to do this is as follows:

```
sage: A = Matrix([[1,1,-2],[-1,2,0],[0,-1,1]])
sage: A^(-1)
[2 1 4]
[1 1 2]
[1 1 3]
```

In Python, we have to input a matrix as an array, and then we can use the linear algebra package.

```
import numpy as np

A = np.array([[1,1,-2],[-1,2,0],[0,-1,1]])

print(np.linalg.inv(A))

[[2. 1. 4.]
 [1. 1. 2.]
 [1. 1. 3.]]
```

Remark 2.4.1. $A \in \mathbb{R}^{n\times n}$ *is invertible if and only if A row reduces to* I_n.

If a matrix does not row reduce to the identity, then it does not have an inverse. So, if we try to row reduce $\left[\begin{array}{c|c} A & I_n \end{array}\right]$ to have the identity on the left, then we cannot find an inverse. We have already seen that nonquare matrices cannot be invertible, In fact, not even all square matrices are invertible (see Example 2.4.4).

Example 2.4.4 (Singular Square Matrix)**.**
Let $A=\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$. *We will attempt the procedure to find the inverse of A.*

$$\left[\begin{array}{cc|cc}1 & 1 & 1 & 0\\ 1 & 1 & 0 & 1\end{array}\right] \xrightarrow{R_2=R_2-R_1} \left[\begin{array}{cc|cc}1 & 1 & 1 & 0\\ 0 & 0 & -1 & 1\end{array}\right]$$

We see that the left-hand side is not the identity matrix, so A is not invertible.
When we try to put this in Sage, we get an error message:

```
sage: A = Matrix([[1,1],[1,1]])
sage: A^(-1)
ZeroDivisionError: matrix must be nonsingular
```

Properties of Inverses Let $A, B \in \mathbb{R}^{n\times n}$ be invertible. Then

- $(A^{-1})^{-1} = A$
  Check: $A^{-1}A = I_n$
- $(kA)^{-1} = k^{-1}A^{-1}$ for all $k \in \mathbb{R} - \{0\}$.
  Check: $(kA)\cdot(k^{-1}A^{-1}) = kAk^{-1}A^{-1} = kk^{-1}AA^{-1} = 1\cdot I_n = I_n$
- AB is also invertible with $(AB)^{-1} = B^{-1}A^{-1}$.
  Check: $(AB)(B^{-1}A^{-1}) = ABB^{-1}A^{-1} = AI_nA^{-1} = AA^{-1} = I_n$
- $(A^T)^{-1} = (A^{-1})^T = A^{-T}$
  Check: $(A^T)(A^{-1})^T = (A^{-1}A)^T = (I_n)^T = I_n$

Proposition 2.4.1. *$A \in R^{n\times n}$ is nonsingular (invertible) if and only if $A\mathbf{x} = \mathbf{b}$ has a unique solution $\mathbf{x} = A^{-1}\mathbf{b}$ for all $\mathbf{b} \in \mathbb{R}^n$.*

Proof.

$$\begin{aligned} \mathbf{b} &= A\mathbf{x} \\ A^{-1}\mathbf{b} &= A^{-1}A\mathbf{x} && \text{(left multiply by } A^{-1}\text{)} \\ A^{-1}\mathbf{b} &= \mathbf{x} && \text{(since } A^{-1}A = I_n\text{)} \end{aligned}$$

□

Example 2.4.5 (Solving Systems Using Matrix Inverses)**.**
Solve $A\mathbf{x} = \mathbf{b}$ *for* $A = \begin{bmatrix}1 & 1 & -2\\ -1 & 2 & 0\\ 0 & -1 & 1\end{bmatrix}$, *when* $\mathbf{b} = \begin{bmatrix}1\\ -3\\ 2\end{bmatrix}$.

From our last example, we know that $A^{-1} = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 1 & 2 \\ 1 & 1 & 3 \end{bmatrix}$*, so*

$$\mathbf{x} = A^{-1}\mathbf{b} = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 1 & 2 \\ 1 & 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ -3 \\ 2 \end{bmatrix} = \begin{bmatrix} 7 \\ 2 \\ 4 \end{bmatrix}.$$

This is a unique solution since A^{-1} *exists.*

Example 2.4.6 (Solving Systems Using Matrix Inverses in Sage)**.**
There are a number of ways to do this in Sage. We will illustrate three of them that we use later on in the course.

```
sage: A = Matrix([[1,1,-2],[-1,2,0],[0,-1,1]])
sage: b = vector([1,-3,2])
sage: A^(-1)*b
(7, 2, 4)

sage: A\b
(7, 2, 4)

sage: (A.augment(b)).rref()
[1 0 0 7]
[0 1 0 2]
[0 0 1 4]
```

In Python, we use the numpy library and the linear algebra package again, in two different ways. We can use the inverse function with the dot() function or simply use the solve() function.

```
import numpy as np

A = np.array([[1,1,-2],[-1,2,0],[0,-1,1]])
b = np.array([1,-3,2])

print(np.linalg.inv(A).dot(b))

[7. 2. 4.]
```

```
print(np.linalg.solve(A,b))
```

```
[7. 2. 4.]
```

As we have seen already, if A is not invertible, Sage will give you an error.

In particullar, for $A \in \mathbb{R}^{n\times n}$, the homogeneous system $A\mathbf{x} = \mathbf{0}_n$ has the unique solution $\mathbf{x} = \mathbf{0}_n$. In other words, the columns of A span $\mathbb{R}^n$ and are linearly independent. This implies the following theorem.

Theorem 2.4.2. $A = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix}$ *is nonsingular/invertible if and only if* $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$ *is a basis for* $\mathbb{R}^n$.

In summary, we have that for $A \in \mathbb{R}^{n\times n}$, the following are equivalent:

- A is nonsingular/invertible.
- A^T is nonsingular/invertible.
- A row reduces to I_n (A is "row equivalent" to I_n).
- $A\mathbf{x} = \mathbf{b}$ has a unique solution $\mathbf{x} \in \mathbb{R}^n$ for every $\mathbf{b} \in \mathbb{R}^n$.
- The columns of A span R^n.
- $\text{rank}(A) = n$.
- $A\mathbf{x} = \mathbf{0}_n$ has only the trivial solution.
- The columns of A are linearly independent.
- The columns of A form a basis for $\mathbb{R}^n$.

2.5 Determinants

Definition 18 (Determinant)**.**
Let $A \in \mathbb{R}^{n\times n}$. The **determinant of** *A, denoted* $\det(A)$ *or* $|A|$*, is a scalar given inductively by*

$$
\begin{aligned}
&n=1: \quad A=\begin{bmatrix} a_{11} \end{bmatrix} && \det(A)=a_{11} \\
&n=2: \quad A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} && \det(A)=a_{11}a_{22}-a_{12}a_{21} \\
&n=3: \quad A=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} && \det(A)=a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}-a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}+a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\
&\vdots
\end{aligned}
$$

Remark 2.5.1. *When $n=2$, the determinant of a matrix represents the signed area of the parallelogram whose sides are formed by the column vectors of the matrix.*

Example 2.5.1 (Geometric Interpretation of Determinant of a 2×2 Matrix)**.**
Let $A=\begin{bmatrix} 3 & -1 \\ 2 & 4 \end{bmatrix}$. *Then* $\det(A)=(3)(4)-(2)(-1)=14$. *Figure 2.4 shows the graph of the column vectors of A in red and blue. The parallelogram formed by these vectors is shaded in gray. We calculate the area of the gray region by finding the area of the whole rectangle and subtracting the area of the yellow and green triangles. The area of the rectangle is* $(4)(6)=24$. *The area of each yellow triangle is* $\frac{1}{2}(3)(2)=3$. *The area of each green triangle is* $\frac{1}{2}(1)(4)=2$. *Thus, the area of the parallelogram is* $24-(3+3+2+2)=24-10=14$, *which is exactly the determinant of A.*

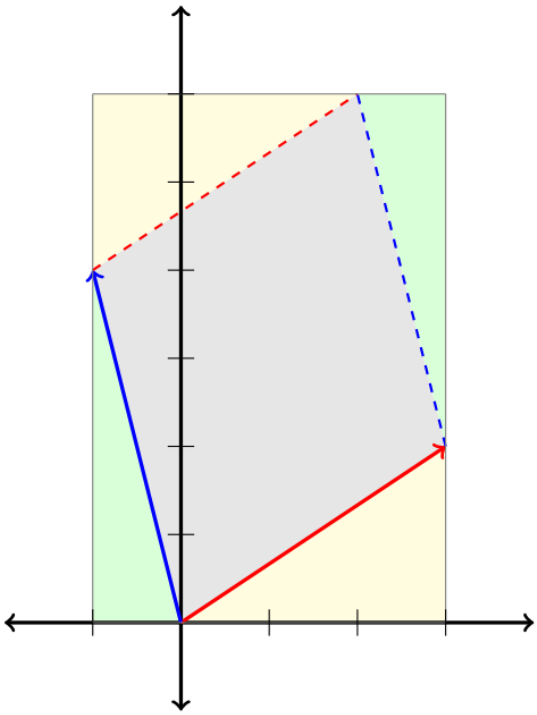


Figure 2.4 A geometric interpretation of determinant in two dimensions.

Remark 2.5.2. *When $n=3$, the determinant of a matrix represents the signed volume of the parallelepiped whose sides are formed by the column vectors of the matrix. When $n>3$, the determinant of a matrix represents the signed hypervolume of the hyperparallelepiped (or $n-$parallelotope) whose sides are formed by the column vectors of the matrix.*

Example 2.5.2 (Determinant of a 3×3 Matrix)**.**
Let $A=\begin{bmatrix}1&2&4\\3&4&2\\2&2&1\end{bmatrix}$. *Then,*

$$\begin{aligned}\det(A) &= 1\cdot\begin{vmatrix}4&2\\2&1\end{vmatrix}-2\cdot\begin{vmatrix}3&2\\2&1\end{vmatrix}+4\cdot\begin{vmatrix}3&4\\2&2\end{vmatrix}\\ &= 1(4\cdot 1-2\cdot 2)-2(3\cdot 1-2\cdot 2)+4(3\cdot 2-2\cdot 4)\\ &= 0+2-8\\ &= -6\end{aligned}$$

So, the signed volume of the parallelepiped formed by $\begin{bmatrix}1\\3\\2\end{bmatrix}$, $\begin{bmatrix}2\\4\\2\end{bmatrix}$, *and* $\begin{bmatrix}4\\2\\1\end{bmatrix}$ *is* -6.

Example 2.5.3 (Determinants in Python and Sage)**.**
Of course, we can ask Python or Sage to calculate the determinant for us. In Python, we can do the following:

```
import numpy as np
A = np.array([[1,2,4],[3,4,2],[2,2,1]])
print(np.linalg.det(A))


-6
```

To do the same in Sage, we can do one of the following:

```
sage: A = Matrix([[1,2,4],[3,4,2],[2,2,1]])
sage: A.det()
-6

sage: det(A)
-6

sage: A.determinant()
-6
```

Properties of Determinants: Let $A \in \mathbb{R}^{n\times n}$. Then,

1. $\det(A^T) = \det(A)$.
2. If A has two identical rows or two identical columns, then $\det(A) = 0$.
3. If A has an entire row or column of 0s, then $\det(A) = 0$.
4. If A is triangular, then $\det(A) = \prod_{i=1}^{n} a_{ii} = a_{11}a_{22}\cdots a_{nn}$. (Note that all diagonal matrices are triangular.)
5. If a row or column of A is multiplied by k to obtain B, then $\det(B) = k \cdot \det(A)$. In particular $\det(k \cdot A) = k^n \det(A)$.
6. If B is obtained by swapping two rows of A, then $\det(B) = -\det(A)$.

7. If B is obtained by adding a nonzero scalar multiple of one row of A to another, then $\det(B) = \det(A)$.

8. If A has two linearly dependent rows or two linearly dependent columns, then $\det(A) = 0$.

9. $\det(AB) = \det(A)\det(B)$.

10. $\det(A^m) = (\det(A))^m$ for $m \in \mathbb{N}$.

11. If A is invertible, then $\det(A^{-1}) = \frac{1}{\det(A)}$.

We encourage the reader to come up with examples to verify each of the 11 properties.

2.6 Inner Product Spaces

Definition 19 (Inner Product Space)**.**
An **inner product space** *is a vector space V over $\mathbb{R}$ together with a map $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ called an* **inner product** *such that for all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$, $c \in \mathbb{R}$, the following properties hold:*

1. *(positivity)* $\langle \mathbf{v}, \mathbf{v} \rangle \geq 0$
2. *(definiteness)* $\langle \mathbf{v}, \mathbf{v} \rangle = 0$ *if and only if* $\mathbf{v} = \mathbf{0}$
3. *(additivity)* $\langle \mathbf{u} + \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{u}, \mathbf{w} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle$
4. *(homogeneity)* $\langle c \cdot \mathbf{u}, \mathbf{v} \rangle = c \cdot \langle \mathbf{u}, \mathbf{v} \rangle$
5. *(symmetry)* $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$

The inner product that we will deal with throughout the remainder of the course is the dot product in Euclidean space, so we define that now.

Definition 20 (Dot Product)**.**
Let $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ *such that* $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$ *and* $\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$. *The* ***dot product****, also called the* ***scalar product****, of* $\mathbf{v}$ *and* $\mathbf{w}$ *is given by*

$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{n} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_n w_n$$

Example 2.6.1 (Dot Product)**.**

$$\begin{bmatrix} 1 \\ -2 \\ 5 \end{bmatrix} \cdot \begin{bmatrix} -3 \\ 1 \\ 2 \end{bmatrix} = (1)(-3)+(-2)(1)+(5)(2) = 5.$$

To do this in Sage, we first define our vectors. Then, we use dot_product() to calculate the dot product.

```
sage: u = vector([1,-2,5])
sage: v = vector([-3,1,2])
sage: u.dot_product(v)
5
```

The dot product of $\mathbf{v}$ with itself is a special case.

$$\mathbf{v}\cdot\mathbf{v} = v_1 v_1 + v_2 v_2 + \cdots + v_n v_n = v_1^2 + v_2^2 + \cdots + v_n^2.$$

Since $(v_i)^2 \geq 0$ for all v_i, $1 \leq i \leq n$, we have that

$$\mathbf{v}\cdot\mathbf{v} \geq 0 \qquad \text{and} \qquad \mathbf{v}\cdot\mathbf{v} = 0 \text{ if and only if } \mathbf{v} = \mathbf{0}. \tag{2.1}$$

This gives rise to a description of the length of a vector.

Definition 21 (Euclidean norm)**.**
*The **Euclidean norm**, also called the **length**, of $\mathbf{v} \in \mathbb{R}^n$ is given by*

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} = \left(\sum_{i=1}^{n} v_i^2 \right)^{1/2}.$$

Example 2.6.2 (Euclidean Norm)**.**

Let $\mathbf{v} = \begin{bmatrix} 1 \\ -2 \\ 0 \\ 3 \end{bmatrix}$. *Then* $\|\mathbf{v}\| = \sqrt{(1)^2+(-2)^2+(0)^2+(3)^2} = \sqrt{14}$.

To compute this with Sage, we use v.norm() after defining $\mathbf{v}$*:*

```
sage: v = vector([1,-2,0,3])
sage: v.norm()
sqrt(14)
```

Note that $\mathbf{v}\cdot\mathbf{v} = v_1^2 + v_2^2 + \cdots + v_n^2 = \|\mathbf{v}\|^2$.

Definition 22 (Angle Between Vectors)**.**
*The **angle** θ **between** any two nonzero vectors* $\mathbf{v}$ *and* $\mathbf{w}$ *in* $\mathbb{R}^n$ *satisfies*

$$\mathbf{v}\cdot\mathbf{w} = \|\mathbf{v}\|\|\mathbf{w}\|\cos(\theta),$$

and is therefore given by

$$\theta = \cos^{-1}\left(\frac{\mathbf{v}\cdot\mathbf{w}}{\|\mathbf{v}\|\|\mathbf{w}\|}\right).$$

Example 2.6.3 (Angle Between Vectors)**.**
Suppose we want to find the angle between $\mathbf{u} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}$ *and* $\mathbf{v} = \begin{bmatrix} -4 \\ 5 \\ 6 \end{bmatrix}$*. We can do this with the help of Sage.*

```
sage: u = vector([1, -2, 3])
sage: v = vector([-4, 5, 6])
sage: arccos(u.dot_product(v)/(norm(u)*norm(v))).n()
1.44866394480060
```

*This is in radians. We use .n() here to get a numerical approximation, because Sage would otherwise return arccos(2/539*sqrt(77)*sqrt(14)), which is not super helpful.*

In lattice-based cryptography, we will often be concerned about the orthogonality of our basis vectors. For this reason, we now define orthogonal vectors and discuss properties and results that can, again, be found in any introductory linear algebra course. For example, see [2] or [12].

Definition 23 (Orthogonal Vectors)**.**
$\mathbf{v}$ *and* $\mathbf{w}$ *are **orthogonal** if they are perpendicular at their point of intersection. We denote orthogonal with the symbol* $\perp$*. That is,* $\mathbf{v} \perp \mathbf{w}$ *if* $\mathbf{v}$ *and* $\mathbf{w}$ *meet at a right angle.*

Lemma 2.6.1 (Dot Product of Orthogonal Vectors)**.**
Two nonzero vectors $\mathbf{v}$ *and* $\mathbf{w}$ *in* $\mathbb{R}^n$ *are orthogonal if and only if* $\mathbf{v}\cdot\mathbf{w} = 0$*.*

Proof. $\cos\theta = 0$ if and only if $\theta = \frac{\pi}{2} + \pi k$ for $k \in \mathbb{Z}$. □

Properties of the Dot Product
Let $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Dot product has the following properties:

1. (Commutativity) $\mathbf{v}\cdot\mathbf{w} = \mathbf{w}\cdot\mathbf{v}$

2. (Distributivity) $\mathbf{u}\cdot(\mathbf{v}+\mathbf{w})=\mathbf{u}\cdot\mathbf{v}+\mathbf{u}\cdot\mathbf{w}$ and $(\mathbf{u}+\mathbf{v})\cdot\mathbf{w}=\mathbf{u}\cdot\mathbf{w}+\mathbf{v}\cdot\mathbf{w}$

3. (Scalar Multiplication) $(c\mathbf{v})\cdot\mathbf{w}=c(\mathbf{v}\cdot\mathbf{w})$

Proposition 2.6.1 (Cauchy-Schwarz Inequality)**.**
For $\mathbf{v},\mathbf{w}\in\mathbb{R}^n$,

$$|\mathbf{v}\cdot\mathbf{w}|\leq\|\mathbf{v}\|\|\mathbf{w}\|.$$

Proof. Let $\mathbf{v},\mathbf{w}\in\mathbb{R}^n$ and $t\in\mathbb{R}$.

$$\begin{aligned}(\mathbf{v}+t\mathbf{w})\cdot(\mathbf{v}+t\mathbf{w})&=\mathbf{v}\cdot\mathbf{v}+\mathbf{v}\cdot(t\mathbf{w})+(t\mathbf{w})\cdot\mathbf{v}+(t\mathbf{w})\cdot(t\mathbf{w})\\&=\|\mathbf{v}\|^2+t(\mathbf{v}\cdot\mathbf{w})+t(\mathbf{v}\cdot\mathbf{w})+t^2\mathbf{w}\cdot\mathbf{w}\\&=\|\mathbf{v}\|^2+2t(\mathbf{v}\cdot\mathbf{w})+t^2\|\mathbf{w}\|^2\end{aligned}$$

By (2.1) above, this expression is non-negative. We let $a=\|\mathbf{w}\|^2,\ b=2\mathbf{v}\cdot\mathbf{w},$ and $c=\|\mathbf{v}\|^2$. Recall from a basic algebra class that if $a>0$, then $at^2+bt+c\geq 0$ for all $t\in\mathbb{R}$ exactly when $b^2-4ac\leq 0$, which gives $b^2\leq 4ac$. Substitution yields

$$\begin{aligned}(2(\mathbf{v}\cdot\mathbf{w}))^2&\leq 4\|\mathbf{v}\|^2\|\mathbf{w}\|^2\\(\mathbf{v}\cdot\mathbf{w})^2&\leq\|\mathbf{v}\|^2\|\mathbf{w}\|^2\end{aligned}$$

Taking square roots gives

$$|\mathbf{v}\cdot\mathbf{w}|\leq\|\mathbf{v}\|\|\mathbf{w}\|.$$

□

Theorem 2.6.2. *Nonzero orthogonal vectors are linearly independent.*

Proof. Let $\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k$ be mutually orthogonal. That is, $\mathbf{v}_i\cdot\mathbf{v}_j=0$ for all $i\neq j$. We want to show that for $a_1\mathbf{v}_1+a_2\mathbf{v}_2+\cdots+a_k\mathbf{v}_k=\mathbf{0}$, we have $a_1=a_2=\cdots=a_k=0$. We compute the dot product of $\mathbf{v}_i$ with both sides: $\mathbf{v}_i\cdot(a_1\mathbf{v}_1+a_2\mathbf{v}_2+\cdots+a_k\mathbf{v}_k)=\mathbf{v}_i\cdot\mathbf{0}$. This gives

$$a_1(\mathbf{v}_i\cdot\mathbf{v}_1)+a_2(\mathbf{v}_i\cdot\mathbf{v}_2)+\cdots+a_i(\mathbf{v}_i\cdot\mathbf{v}_i)+\cdots+a_k(\mathbf{v}_i\cdot\mathbf{v}_k)=0$$

Since the vectors are mutually orthogonal, $\mathbf{v}_i\cdot\mathbf{v}_1=0,\mathbf{v}_i\cdot\mathbf{v}_2=0,\ldots,\mathbf{v}_i\cdot\mathbf{v}_k=0$, but $\mathbf{v}_i,\mathbf{v}_i\neq 0$. Thus, we have that $a_i(\mathbf{v}_i\cdot\mathbf{v}_i)=0$. But, since $\mathbf{v}_i\cdot\mathbf{v}_i=\|\mathbf{v}_1\|^2>0$, we get that $a_i=0$. We can repeat this process for every $i=1,2,\ldots,k$. So, this will give us that $a_1=a_2=\cdots=a_k=0$. Thus, $\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k$ are linearly independent. □

Remark 2.6.1. *Nonzero orthonormal vectors are linearly independent. This follows the same proof, just with* $\|\mathbf{v}_i\|^2 = 1$.

Definition 24 (Orthogonal and Orthonormal Bases)**.**
A basis $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ *of vector space* V *is an* ***orthogonal basis*** *of* V *if* $\mathbf{v}_i \cdot \mathbf{v}_j$ *for all* $i \neq j$, $1 \leq i, j \leq n$.
If, in addition, $\|\mathbf{v}_i\| = 1$ *for all* $1 \leq i \leq n$, *then* $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ *is an* ***orthonormal basis*** *for* V.

Example 2.6.4 (Normalizing a Vector)**.**
Suppose $\mathbf{v} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$. *To normalize* $\mathbf{v}$, *we simply divide by the norm.*

$$\mathbf{u} = \frac{1}{\|\mathbf{v}\|}\mathbf{v} = \frac{1}{\sqrt{14}} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{14}} \\ \frac{2}{\sqrt{14}} \\ \frac{3}{\sqrt{14}} \end{bmatrix}$$

We can normalize a vector in Python as follows:

```
import numpy as np

v1 = array([1,2,3])
v1/np.linalg.norm(v1)
```

This will output the following:

```
array([0.26726124, 0.53452248, 0.80178373])
```

We can normalize a vector in Sage in a similar way:

```
sage: v1 = vector([1,2,3])
sage: v1/v1.norm()
(1/14*sqrt(14), 1/7*sqrt(14), 3/14*sqrt(14))

sage: v1/v1.norm().n()
(0.267261241912424, 0.534522483824849, 0.801783725737273)
```

Remark 2.6.2. *If* $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ *is an orthogonal basis for vector space* V*, and if* $\mathbf{v} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_n\mathbf{v}_n \in V$ *for* $a_i \in \mathbb{R}$, $1 \leq i \leq n$*, then*

$$\begin{aligned}
\|\mathbf{v}\|^2 &= \|a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_n\mathbf{v}_n\|^2 \\
&= (a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_n\mathbf{v}_n) \cdot (a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_n\mathbf{v}_n) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j (\mathbf{v}_i \cdot \mathbf{v}_j) \\
&= \sum_{i=1}^{n} a_i^2 \|\mathbf{v}_i\|^2 \qquad (\textit{since } \mathbf{v}_i \cdot \mathbf{v}_j = 0 \textit{ when } i \neq j)
\end{aligned}$$

If $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ *is an orthonormal basis for* V*, then*

$$\begin{aligned}
\|\mathbf{v}\|^2 &= \sum_{i=1}^{n} a_i^2 \|\mathbf{v}_i\|^2 \\
&= \sum_{i=1}^{n} a_i^2 \qquad (\textit{since } \|\mathbf{v}_i\| = 1 \textit{ for all } i)
\end{aligned}$$

Example 2.6.5 (Orthogonal Basis)**.**
Determine if $\left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \right\}$ *is an orthogonal basis for* $\mathbb{R}^3$.

We have three vectors in $\mathbb{R}^3$*, so it suffices to check that they are mutually orthogonal. We will do this with Sage.*

```
sage: v1 = vector([1,1,1])
      v2 = vector([-2,1,1])
      v3 = vector([0,1,-1])
sage: v1.dot_product(v2)
0
sage: v2.dot_product(v3)
0
sage: v1.dot_product(v3)
0
```

Since all dot products are 0*, the vectors are mutually orthogonal, and so the given vectors form an orthogonal basis for* $\mathbb{R}^3$.

Example 2.6.6 (Orthonormal Basis)**.**
Using the vectors from the last example, create an orthonormal basis for $\mathbb{R}^3$.

All we need to do here is normalize the vectors. We will again use Sage to do this.

```
sage: v1/norm(v1)
(1/3*sqrt(3), 1/3*sqrt(3), 1/3*sqrt(3))

sage: v2/norm(v2)
(-1/3*sqrt(6), 1/6*sqrt(6), 1/6*sqrt(6))

sage: v3/norm(v3)
(0, 1/2*sqrt(2), -1/2*sqrt(2))
```

So, we have that an orthonormal basis for $\mathbb{R}^3$ *is* $\left\{ \begin{bmatrix} \frac{1}{3}\sqrt{3} \\ \frac{1}{3}\sqrt{3} \\ \frac{1}{3}\sqrt{3} \end{bmatrix}, \begin{bmatrix} \frac{-1}{3}\sqrt{6} \\ \frac{1}{6}\sqrt{6} \\ \frac{1}{6}\sqrt{6} \end{bmatrix}, \begin{bmatrix} 0 \\ \frac{1}{2}\sqrt{2} \\ \frac{-1}{2}\sqrt{2} \end{bmatrix} \right\}$.

2.7 The Gram Schmidt Algorithm

The Gram-Schmidt Algorithm inputs a basis for a vector space $V \subseteq \mathbb{R}^n$ and outputs an orthogonal basis for the same vector space. This concept is important for lattice reduction algorithms later in the text. In order to perform the standard Gram-Schmidt Algorithm, we first need a few more definitions.

Definition 25 (Orthogonal Complement)**.**
For vector space $V \subseteq \mathbb{R}^n$*, the set* $V^\perp$ *(pronounced "V-perp"), called the* ***orthogonal complement of*** V*, is the set of vectors in* $\mathbb{R}^n$ *that are orthogonal to every vector in* V*. In other words,*

$$V^\perp = \{\mathbf{u} \in \mathbb{R}^n : \mathbf{u} \cdot \mathbf{v} = 0 \text{ for all } \mathbf{v} \in V\}.$$

Remark 2.7.1. *For any vector space* $V \subset \mathbb{R}^n$*,* $dim(V) + dim(V^\perp) = n$.

Definition 26 (Orthogonal Projection)**.**
Let V *be a vector space in* $\mathbb{R}^n$*. Every vector* $\mathbf{y} \in \mathbb{R}^n$ *can be expressed uniquely as* $\mathbf{y} = \mathbf{v} + \hat{\mathbf{v}}$*, where* $\mathbf{v} \in V$ *and* $\hat{\mathbf{v}} \in V^\perp$*. The vector* $\mathbf{v}$ *is called the* ***orthogonal projection of*** $\mathbf{y}$ ***onto*** V
Let $\mathbf{u}, \mathbf{v} \in V$*. The* **orthogonal projection of v onto u** *is given by* $proj_{\mathbf{u}}\mathbf{v} = \dfrac{\mathbf{v} \cdot \mathbf{u}}{\mathbf{u} \cdot \mathbf{u}}\mathbf{u}$.

If $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ *is an orthogonal basis for* V*, then the orthogonal projection of* $\mathbf{y}$ *onto* V *is given by*

$$\begin{aligned}\mathbf{v} = proj_V(\mathbf{y}) &= \left(\frac{\mathbf{y}\cdot\mathbf{u}_1}{\mathbf{u}_1\cdot\mathbf{u}_1}\right)\mathbf{u}_1 + \left(\frac{\mathbf{y}\cdot\mathbf{u}_2}{\mathbf{u}_2\cdot\mathbf{u}_2}\right)\mathbf{u}_2 + \cdots + \left(\frac{\mathbf{y}\cdot\mathbf{u}_n}{\mathbf{u}_1\cdot\mathbf{u}_n}\right)\mathbf{u}_n \\ &= \frac{\mathbf{y}\cdot\mathbf{u}_1}{\|\mathbf{u}_1\|^2}\mathbf{u}_1 + \frac{\mathbf{y}\cdot\mathbf{u}_2}{\|\mathbf{u}_2\|^2}\mathbf{u}_2 + \cdots + \frac{\mathbf{y}\cdot\mathbf{u}_n}{\|\mathbf{u}_n\|^2}\mathbf{u}_n\end{aligned}$$

Remark 2.7.2. *Note that* $\hat{\mathbf{v}} = \mathbf{y} - \mathbf{v} \in V^\perp$ *is always orthogonal to all vectors in* V*. The distance from a vector* $\mathbf{y} \in \mathbb{R}^n$ *to the vector space* V *is the distance between* $\mathbf{y}$ *and the orthogonal projection of* $\mathbf{y}$ *onto* V*. So, the "closest vector" in* V *to* $\mathbf{y} \in \mathbb{R}^n$ *is* $\mathbf{v}$*. In other words, we have the following theorem:*

Theorem 2.7.1. *Let* $V \subset \mathbb{R}^n$ *be a vector space, and let* $\mathbf{y} \in \mathbb{R}^n$ *be such that* $\mathbf{y} = \mathbf{v} + \hat{\mathbf{v}}$*, with* $\mathbf{v} \in V$ *and* $\hat{\mathbf{v}} \in V^\perp$*. Then for any* $\mathbf{w} \in V$*,* $\|\mathbf{y} - \mathbf{w}\|$ *is minimized when* $\mathbf{w} = \mathbf{v}$*. That is,* $\|\mathbf{y} - \mathbf{v}\| < \|\mathbf{y} - \mathbf{w}\|$ *for all* $\mathbf{v} \neq \mathbf{w} \in V$*.*

Proof. Let $\mathbf{w} \in V$ be distinct from $\mathbf{v} \in V$. Since V is a vector space, $\mathbf{w} - \mathbf{v} \in V$. Since $\hat{\mathbf{v}} = \mathbf{y} - \mathbf{v} \in V^\perp$, $\hat{\mathbf{v}}$ is orthogonal to every vector in V. In particular, $\mathbf{y} - \mathbf{v}$ is orthogonal to $\mathbf{w} - \mathbf{v}$. We can write $\mathbf{y} - \mathbf{w} = (\mathbf{y} - \mathbf{v}) + (\mathbf{v} - \mathbf{w})$. This implies that $\|\mathbf{y} - \mathbf{w}\|^2 = \|\mathbf{y} - \mathbf{v}\|^2 + \|\mathbf{v} - \mathbf{w}\|^2$. Since $\|\mathbf{v} - \mathbf{w}\|^2 > 0$, we then have that $\|\mathbf{y} - \mathbf{w}\|^2 > \|\mathbf{y} - \mathbf{v}\|^2$. Taking square roots yields $\|\mathbf{y} - \mathbf{v}\| < \|\mathbf{y} - \mathbf{w}\|$, as desired. □

We henceforth denote the projection of $\mathbf{y}$ onto vector space V by $\text{proj}_V(\mathbf{y})$.

We can take any basis for a vector space and reduce it to an orthogonal or orthonormal basis using the Gram-Schmidt Algorithm.
The standard Gram-Schmidt Algorithm is as follows:

Choose a basis $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ for V. Gram-Schmidt computes an orthogonal basis $\mathscr{B}' = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ for V.

$$
\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1 \\
\mathbf{u}_2 &= \mathbf{v}_2 - \frac{\mathbf{v}_2 \cdot \mathbf{u}_1}{\mathbf{u}_1 \cdot \mathbf{u}_1}\mathbf{u}_1 \\
\mathbf{u}_3 &= \mathbf{v}_3 - \frac{\mathbf{v}_3 \cdot \mathbf{u}_1}{\mathbf{u}_1 \cdot \mathbf{u}_1}\mathbf{u}_1 - \frac{\mathbf{v}_3 \cdot \mathbf{u}_2}{\mathbf{u}_2 \cdot \mathbf{u}_2}\mathbf{u}_2 \\
&\vdots \\
\mathbf{u}_n &= \mathbf{v}_n - \sum_{i=1}^{n-1} \frac{\mathbf{v}_n \cdot \mathbf{u}_i}{\mathbf{u}_1 \cdot \mathbf{u}_i}\mathbf{u}_i
\end{aligned}
$$

Theorem 2.7.2 (Gram-Schmidt Algorithm)**.**
Let $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ *be a basis for vector space* $V \in \mathbb{R}^n$*. The following algorithm creates an orthogonal basis* $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ *for* V*.*
The Gram-Schmidt Orthogonalization Algorithm takes basis vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_n$ *for* V *as input and outputs orthogonal basis vectors* $\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_n$ *for* V*.*

Algorithm 1 Gram-Schmidt Orthogonalization Algorithm

1: Set $\mathbf{u}_1 = \mathbf{v}_1$
2: for $i = 2, 3, \ldots, n$
3: for $1 \leq j < i$
4: $\mu_{ij} = \frac{\mathbf{v}_i \cdot \mathbf{u}_j}{\|\mathbf{u}_j\|^2}$
5: Set $\mathbf{u}_i = \mathbf{v}_i - \sum_{j=1}^{i-1} \mu_{ij}\mathbf{u}_j$

Example 2.7.1. *Gram-Schmidt Orthogonalization*
Let $V = span\left\{ \begin{bmatrix} -1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}$*. Find an orthogonal basis for* V*.*

$$\mathbf{u}_1 = \mathbf{v}_1$$

$$\mathbf{u}_2 = \mathbf{v}_2 - \frac{\mathbf{v}_2 \cdot \mathbf{u}_1}{\mathbf{u}_1 \cdot \mathbf{u}_1}\mathbf{u}_1$$

$$= \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} - \frac{2}{3}\begin{bmatrix} -1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{u}_2 = \begin{bmatrix} \frac{-1}{3} \\ \frac{-2}{3} \\ \frac{1}{3} \\ 0 \end{bmatrix}$$

$$\mathbf{u}_3 = \mathbf{v}_3 - \frac{\mathbf{v}_3 \cdot \mathbf{u}_1}{\mathbf{u}_1 \cdot \mathbf{u}_1}\mathbf{u}_1 - \frac{\mathbf{v}_3 \cdot \mathbf{u}_2}{\mathbf{u}_2 \cdot \mathbf{u}_2}\mathbf{u}_2$$

$$= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} - \frac{-1}{3}\begin{bmatrix} -1 \\ 1 \\ 1 \\ 0 \end{bmatrix} - \frac{-1/3}{2/3}\begin{bmatrix} \frac{-1}{3} \\ \frac{-2}{3} \\ \frac{1}{3} \\ 0 \end{bmatrix}$$

$$\mathbf{u}_3 = \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 1 \end{bmatrix}$$

So, $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ *is an orthogonal basis for* V*.*

If we wanted to normalize this to get an orthonormal basis, we would simply divide by norms.

$$\mathbf{w}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} = \frac{1}{\sqrt{3}} \begin{bmatrix} -1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{w}_1 = \begin{bmatrix} \frac{-1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ 0 \end{bmatrix}$$

$$\mathbf{w}_2 = \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} = \frac{1}{\sqrt{2/3}} \begin{bmatrix} \frac{-1}{3} \\ \frac{-2}{3} \\ \frac{1}{3} \\ 0 \end{bmatrix}$$

$$\mathbf{w}_2 = \begin{bmatrix} \frac{-1}{\sqrt{6}} \\ \frac{-2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \\ 0 \end{bmatrix}$$

$$\mathbf{w}_3 = \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} = \frac{1}{\sqrt{3/2}} \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 1 \end{bmatrix}$$

$$\mathbf{w}_3 = \begin{bmatrix} \frac{1}{\sqrt{6}} \\ 0 \\ \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{6}} \end{bmatrix}$$

Then, $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$ *is an orthonormal basis for* V*.*

We will now present another way of thinking of a Gram-Schmidt basis, but we first need a few more definitions and notions.

Definition 27 (Orthogonal Matrix)**.**
An **orthogonal matrix**, Q, *is an* $n \times n$ *matrix with orthonormal row and column vectors such that* $Q^T Q = I_n$*.*

Definition 28 (Upper Triangular Matrix)**.**
A matrix is **upper triangular** *if all matrix entries below the main diagonal are zero. We saw these with row echelon forms of matrices.*

QR Decomposition

One of the ways that we can decompose a matrix into two matrix factors is called the QR decomposition. The QR decomposition decomposes a matrix A into $A = QR$, where Q is orthogonal and R is upper triangular.

The Gram-Schmidt process produces the columns of Q. We can get the columns of R by either keeping track of the column operations or by computing $R = Q^T A$. We will briefly describe the QR process here so that we can use it to code Gram-Schmidt in Python and Sage more efficiently.

Let $A = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}$.

We begin with our same basis vectors (the columns of our matrix A) $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$, and we compute our Gram-Schmidt orthogonal vectors as before. At each step of the process, we denote $\mathbf{e}_i = \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|}$. We can then express the columns of A as linear combinations of the orthonormal basis:

$$\begin{aligned}
\mathbf{v}_1 &= (\mathbf{e}_1 \cdot \mathbf{v}_1)\mathbf{e}_1 \\
\mathbf{v}_2 &= (\mathbf{e}_1 \cdot \mathbf{v}_2)\mathbf{e}_1 + (\mathbf{e}_2 \cdot \mathbf{v}_2)\mathbf{e}_2 \\
\mathbf{v}_3 &= (\mathbf{e}_1 \cdot \mathbf{v}_3)\mathbf{e}_1 + (\mathbf{e}_2 \cdot \mathbf{v}_3)\mathbf{e}_2 + (\mathbf{e}_3 \cdot \mathbf{v}_3)\mathbf{e}_3 \\
&\vdots \\
\mathbf{v}_n &= \sum_{i=1}^{n} (\mathbf{e}_i \cdot \mathbf{v}_n)\mathbf{e}_i
\end{aligned}$$

where $\mathbf{e}_i \cdot \mathbf{v}_i = \|\mathbf{u}_i\|$. This can be rewritten in the form $A = QR$, where

$$Q = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_n \end{bmatrix} \qquad R = \begin{bmatrix} \mathbf{e} \cdot \mathbf{v}_1 & \mathbf{e}_1 \cdot \mathbf{v}_2 & \mathbf{e}_1 \cdot \mathbf{v}_3 & \dots \\ 0 & \mathbf{e}_2 \cdot \mathbf{v}_2 & \mathbf{e}_2 \cdot \mathbf{v}_2 & \dots \\ 0 & 0 & \mathbf{e}_e \cdot \mathbf{v}_3 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Example 2.7.2 (QR Decomposition)**.**
We will follow an example from [23] to illustrate this.
Let $A = \begin{bmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{bmatrix}$. *Use Gram-Schmidt to find* Q. *Let* $U = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix}$ *and* $Q =$

$\left[\frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \quad \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \quad \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|}\right]$. *Then U, the matrix with orthogonal columns is*

$$U = \begin{bmatrix} 12 & -69 & \frac{-58}{5} \\ 6 & 158 & \frac{6}{5} \\ -4 & 30 & -33 \end{bmatrix}$$

So, Q, an orthogonal matrix is given by

$$Q = \begin{bmatrix} \frac{6}{7} & \frac{-69}{175} & \frac{-58}{175} \\ \frac{3}{7} & \frac{158}{175} & \frac{6}{175} \\ \frac{-2}{7} & \frac{6}{35} & \frac{-33}{35} \end{bmatrix}$$

Then, we can find R by $R = Q^T A$, which gives

$$R = \begin{bmatrix} 14 & 21 & -14 \\ 0 & 175 & -70 \\ 0 & 0 & 35 \end{bmatrix}$$

So, we have

$$A = QR = \begin{bmatrix} \frac{6}{7} & \frac{-69}{175} & \frac{-58}{175} \\ \frac{3}{7} & \frac{158}{175} & \frac{6}{175} \\ \frac{-2}{7} & \frac{6}{35} & \frac{-33}{35} \end{bmatrix} \begin{bmatrix} 14 & 21 & -14 \\ 0 & 175 & -70 \\ 0 & 0 & 35 \end{bmatrix}$$

Example 2.7.3 (Gram-Schmidt with QR Decomposition in Python)**.**
In Python, we can now execute the Gram-Schmidt orthonormalization process as follows:

```
import numpy as np

def gramschmidt(A):
    Q, R = np.linalg.qr(A)
    return Q

A = np.array([[1,2,3],[4,5,6],[7,8,9]])
gramschmidt(A)
array([[-0.12309149,  0.90453403,  0.40824829],
       [-0.49236596,  0.30151134, -0.81649658],
       [-0.86164044, -0.30151134,  0.40824829]])
```

2.8 Exercises

Many of these exercises are adapted from [2].

1. Let $V = \left\{ \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3 \right\}$ with addition defined by $\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \oplus \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{bmatrix}$ and scalar multiplication defined by $\alpha \odot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \alpha x \\ \alpha y \\ \alpha z \end{bmatrix}$. Determine whether or not $(V, \oplus, \odot)$ forms a vector space.

2. For the given vector space V, prove whether or not the subset S is a subspace of V.

   (a) $S = \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \mathbb{R}^{2\times 2} : 2b - c = 0 \right\}$; $V = \mathbb{R}^{2\times 2}$, the vector space of all 2×2 matrices

   (b) $S = \{a_2 x^2 + a_1 x + a_0 : a_0, a_1, a_2 > 0\}$; $V = \mathscr{P}_2$, the vector space of polynomials of degree *at most* 2.

3. Let V_1 and V_2 be subspaces of vector space V. Show that $V_1 \cap V_2$ is also a subspace of V. Is $V_1 \cup V_2$ always a subspace of V?

4. Show that $\begin{bmatrix} -3 \\ -3 \\ 1 \end{bmatrix}$ is a linear combination of the vectors $\begin{bmatrix} -1 \\ -1 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ -2 \end{bmatrix}, \begin{bmatrix} -3 \\ -1 \\ -2 \end{bmatrix}$.

5. Let $S = \{x^2 + 1, x + 2, -x^2 + x\}$.

   (a) Does S span $\mathscr{P}_3$?

   (b) Does S span $\mathscr{P}_2$?

6. Is $\mathscr{B} = \left\{ \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$ a basis for the vector space $\mathbb{R}^3$?

7. Is $S = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} \right\}$ a basis for $\mathbb{R}^{2\times 2}$?

8. Let W be the subspace of $\mathbb{R}^{2\times 2}$ matrices with trace equal to 0. That is, $W = \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \mathbb{R}^{2\times 2} : a + d = 0 \right\}$. Show that $\mathscr{B} = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \right\}$ is a basis for W.

9. Let $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ be a basis for vector space V. Show that $\{\mathbf{v}_1, \mathbf{v}_1 + \mathbf{v}_2, \mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3\}$ is also a basis for V.

10. Let $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ be a basis for the vector space V. Let $c \in \mathbb{R}$, and let $A \in \mathbb{R}^{n \times n}$.

    (a) Show that $\mathscr{B}' = \{c\mathbf{v}_1, c\mathbf{v}_2, \ldots, c\mathbf{v}_n\}$ is also a basis for V.

    (b) Show that $\mathscr{B}'' = \{A\mathbf{v}_1, A\mathbf{v}_2, \ldots, A\mathbf{v}_n\}$ is also a basis for V.

11. Find a basis for the subspace $S = \{\mathbf{x} \in \mathbb{R}^4 : A\mathbf{x} = \mathbf{0}\}$, where $A = \begin{bmatrix} 3 & 3 & 1 & 3 \\ -1 & 0 & -1 & -1 \\ 2 & 0 & 2 & 1 \end{bmatrix}$.

12. Suppose $A \in \mathbb{R}^{3 \times 3}$, and suppose that $A\mathbf{x} = \mathbf{b}$ has a solution $\mathbf{x} \in \mathbb{R}^3$ for all $\mathbf{b} \in \mathbb{R}^3$. Explain why the columns of A span $\mathbb{R}^3$.

13. Consider the linear system

$$\begin{cases} x - y + 2z & = a \\ 2x + 4y - 3z & = b \\ 4x + 2y + z & = c \end{cases}$$

    Find the values of a, b, and c for which the system will have

    (a) no solution

    (b) exactly one solution

    (c) infinitely many solutions

14. Let $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times k}$, $C \in \mathbb{R}^{k \times n}$, and $D \in \mathbb{R}^{n \times n}$. Determine the dimensions of each of the following. If it is not defined, say so.

    (a) AB

    (b) $C^T B$

    (c) $A(D^T C)^T$

    (d) $(A^T C^T)^T B$

    (e) $(A - D^T)B$

    (f) $BC - A$

15. Let $A, B \in \mathbb{R}^{3 \times 3}$ with $\det(A) = 2$ and $\det(B) = 3$. Evaluate each of the following:

    (a) $\det(2A^{-1})$

(b) $\det(AB^T)$

(c) $\det(A^{-1}B^T)$

(d) $\det(-B^3)$

(e) $\det((AB)^{-1})$

16. Explain why $\det(A^{-1}) = \frac{1}{\det(A)}$ when A is invertible.

17. Find the determinant of each of the following:

(a) $A = \begin{bmatrix} 6 & 9 \\ 1 & 3 \end{bmatrix}$

(b) $B = \begin{bmatrix} 2 & 7 & 11 \\ 3 & 4 & 10 \\ 5 & 6 & 16 \end{bmatrix}$

18. Find all vectors in $\mathbb{R}^2$ that are orthogonal to $\mathbf{v} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ with respect to the dot product.

19. Verify that $\left\{ \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}, \begin{bmatrix} \frac{-1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix} \right\}$ is orthogonal. Then, normalize it to produce an orthonormal set.

20. Is $\left\{ \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \right\}$ an orthonormal basis for $\mathbb{R}^3$?

21. Let $\mathbf{u} = \begin{bmatrix} -3 \\ 5 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$.

(a) Verify that $\{\mathbf{u}, \mathbf{v}\}$ is a basis for $\mathbb{R}^2$.

(b) Use Gram-Schmidt to find an orthogonal basis for $\mathbb{R}^2$.

(c) Find an orthonormal basis for $\mathbb{R}^2$.

22. Let $\mathbf{v}_1 = \begin{bmatrix} 4 \\ 2 \\ -3 \end{bmatrix}$, $\mathbf{v}_2 = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$, and $\mathbf{v}_3 = \begin{bmatrix} 0 \\ -2 \\ 5 \end{bmatrix}$.

(a) Verify that $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ is a basis for $\mathbb{R}^3$.

(b) Use Gram-Schmidt to find an orthonormal basis for $\mathbb{R}^3$.

2.9 Computer Exercises

1. Determine whether or not $\mathscr{B} = \left\{ \begin{bmatrix} 17 \\ -31 \\ 12 \end{bmatrix}, \begin{bmatrix} -11 \\ 14 \\ -2 \end{bmatrix}, \begin{bmatrix} 6 \\ -16 \\ 10 \end{bmatrix} \right\}$ forms a basis for $\mathbb{R}^3$.

2. Let $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 7 & 9 \\ 3 & 6 & 9 \end{bmatrix}$. Find a basis for the subspace spanned by the columns of the matrix M.

3. Let $A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 6 \\ 2 & 6 & 13 \end{bmatrix}$.

   (a) Find A^{-1}.

   (b) Solve the linear system $A\mathbf{x} = \mathbf{b}$, where $\mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$.

4. Find $\det(A)$ for $A = \begin{bmatrix} 0 & 1 & 3 \\ 2 & 1 & 2 \\ 1 & 2 & 1 \end{bmatrix}$.

5. Determine if $M = \begin{bmatrix} 4 & 7 & -3 & 2 \\ -1 & 0 & 11 & 3 \\ -6 & 8 & 1 & 4 \\ 2 & 9 & 10 & -7 \end{bmatrix}$ is invertible.

6. Write code that verifies mutual orthogonality of vectors in $\mathbb{R}^n$ for $n \leq 4$.

7. Use the given algorithm to write a Python code to execute the Gram-Schmidt Orthogonalization algorithm. Use that code to verify your answers to problems 21 and 22 above.

8. Find an orthogonal basis for $\mathbb{R}^3$ if $\mathscr{B} = \left\{ \begin{bmatrix} 13 \\ 21 \\ 4 \end{bmatrix}, \begin{bmatrix} -8 \\ 11 \\ 9 \end{bmatrix}, \begin{bmatrix} 2 \\ -13 \\ 10 \end{bmatrix} \right\}$ is a basis for $\mathbb{R}^3$.

9. Using the provided Python code, find an orthonormal basis for $\mathbb{R}^5$ with basis
$$\left\{\begin{bmatrix}4\\-3\\0\\1\\2\end{bmatrix}, \begin{bmatrix}1\\2\\0\\1\\1\end{bmatrix}, \begin{bmatrix}-1\\0\\4\\-2\\3\end{bmatrix}, \begin{bmatrix}5\\1\\-1\\2\\0\end{bmatrix}, \begin{bmatrix}0\\3\\-2\\1\\4\end{bmatrix}\right\}.$$

CHAPTER

3

A REVIEW OF ABSTRACT ALGEBRA

This review of abstract algebra contains well-known definitions and results. For a complete review, see for example, [10] and [15], which we have used here.

3.1 Basic Arithmetic

Definition 29 (Divides)**.**
Let $a, b \in \mathbb{Z}$ with $b \neq 0$. We say that b **divides** *a if $a = bc$ for some $c \in \mathbb{Z}$. We write $b|a$ for "b divides a" and $b \nmid a$ for "b does not divide a." We may also say that b* **is a divisor of** *a or that b* **is a factor of** *a.*
Common divisors *of a and b are numbers that divide both a and b.*

Definition 30 (GCD)**.**
Let $a, b \in \mathbb{Z}$ with $a \neq 0$ and $b \neq 0$. The **greatest common divisor (gcd)** *of a and b is the largest integer d that divides both a and b. In other words,* $\gcd(a, b) = d$ *if*

1. *$d|a$ and $d|b$, and*
2. *if $c|a$ and $c|b$, then $c \leq d$.*

Example 3.1.1 (GCD)**.**
Let's find the gcd of 36 *and* 16.

The divisors of 36 *(up to sign) are* $1, 2, 3, 4, 6, 9, 12, 18$, *and* 36. *The divisors of* 16 *(again, up to sign) are* $1, 2, 4, 8$, *and* 16. *We see that the greatest common divisor is* 4.

We can find the gcd of two integers in Python by first importing the math library.

```
import math
print(math.gcd(36, 16))
4
```

If we want to calculate this in SageMath, we simply type

```
sage: gcd(36,16)
4
```

Definition 31 (Prime and Composite Numbers)**.**
A positive integer $p \geq 2$ *is called* **prime** *if* p *is divisible only by* ± 1 *and* $\pm p$. *A positive integer that is not prime (i.e., one that has at least one divisor other than* 1 *and itself) is called* **composite**.

Example 3.1.2 (GCD with Python)**.**
In Python, $a\%b$ *gives the remainder when* a *is divided by* b. *If the remainder is* 0, *then* $b|a$. *So, we can check if a number is prime by checking remainders in Python:*

```
n = 13479

composite = False

if n > 1:
    for i in range(2,n):
        if (n % i) == 0:
            composite = True
            break

if composite:
    print(n, "is not a prime number.")
```

```
else:
    print(n, "is a prime number.")

13479 is not a prime number.
```

Example 3.1.3 (GCD with Sage)**.**
To determine if a number is prime in Sage, we use the .is_prime() command. If the code returns True, then the number is prime. If it returns False, then the number is composite.

```
sage: a = 13479
sage: a.is_prime()
False

sage: b = 7919
sage: b.is_prime()
True
```

Definition 32 (Relatively Prime)**.**
$a, b \in \mathbb{Z}$ *are called* **relatively prime** *if* $\gcd(a,b) = 1$.

Example 3.1.4 (Relatively Prime)**.**

```
sage: gcd(437895484,3289547)
1
```

Since $\gcd(437895484, 3289547) = 1$*, we have that* $437,895,484$ *and* $3,289,547$ *are relatively prime.*

Note that neither $437,895,484$ *nor* $3,289,547$ *is prime (you should check this with Sage), but they are relatively prime to one another because they share no factors greater than* 1.

Remark 3.1.1. *If p is prime, then all positive integers less than p are relatively prime to p.*

Theorem 3.1.1 (Fundamental Theorem of Arithmetic)**.**
Every integer greater than 1 *can be decomposed into a unique (up to ordering) product of prime numbers.*

This is also known as the Unique Factorization Theorem or the Prime Factorization Theorem. That is, every integer a can be written as $a = p_1^{e_1} \cdot p_2^{e_2} \cdot p_3^{e_3} \cdots p_k^{e_k}$, where p_i prime and $e_i \in \mathbb{Z}_{>0}$ $(1 \leq i \leq k)$.

Example 3.1.5 (Factoring)**.**
$600 = 2^3 \cdot 3 \cdot 5^2$

Of course, we can just ask Sage to factor it for us.

```
sage: factor(600)
2^3 * 3 * 5^2
```

3.2 Euclidean Algorithm

Proposition 3.2.1 (The Division Algorithm)**.**
For nonzero integers a and b, there exist unique integers q and r such that

$$a = qb + r,$$

with $0 \leq r < |b|$. We call q the **quotient** *and r the* **remainder**.

Example 3.2.1 (Division Algorithm)**.**
In Python, we can define a function divisionalgorithm(a,b) that takes in integers (a,b) with $a > b$ and outputs the pair (q,r).

```
def divisionalgorithm(a,b):
    q = 0
    r = a
    while r >= b:
        r -= b
        q += 1
    return q,r

a = 44
b = 3
print(divisionalgorithm(a,b))
(14,2)
```

This says that $44 = 14 \cdot 3 + 2$.

Example 3.2.2 (Division Algorithm in Python and Sage)**.**
Another way to implement the division algorithm in Python or in Sage is to simply use the //

command and the % command. Note that // gives the integer obtained by performing the division, and % gives the remainder of the division.

```
print(44//3, 44%3)
14, 2
```

The Euclidean Algorithm

We define the Euclidean Algorithm to find the $\gcd(a,b)$ as follows:

$$\begin{aligned} a &= q_0 b + r_0 \\ b &= q_1 r_0 + r_1 \\ r_0 &= q_2 r_1 + r_2 \\ r_1 &= q_3 r_2 + r_3 \\ &\vdots \\ r_{n-2} &= q_n r_{n-1} + r_n \\ r_{n-1} &= q_{n+1} r_n, \end{aligned}$$

where $r_n = \gcd(a,b)$ is the last nonzero remainder.

Example 3.2.3. *Let us find* $\gcd(1476, 684)$.

$$\begin{aligned} 1476 &= 684 \cdot 2 + 108 \\ 684 &= 108 \cdot 6 + 36 \\ 108 &= 36 \cdot 3 + 0 \end{aligned}$$

Since 36 *is the last nonzero remainder,* $\gcd(1476, 684) = 36$.

Example 3.2.4. *Now that we know the Euclidean Algorithm, we have a new way to calculate the gcd of two integers with Python, using recursion.*

```
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, (a % b))

a = 36
b = 16
```

```
print(gcd(a,b))
4
```

Extended Euclidean Algorithm

Suppose that a and b are both positive integers. The Extended Euclidean Algorithm says that the equation $a\,u + b\,v = \gcd(a,b)$ always has a solution for integers u and v. In other words, we can find $u, v \in \mathbb{Z}$ such that $a\,u + b\,v = \gcd(a,b)$. We will illustrate this by way of an example.

In Example 3.2.3, we found that $\gcd(1476, 684) = 36$ using the Euclidean Algorithm. To find integers u, v such that $1476 \cdot u + 684 \cdot v = 36$, we simply "work backwards." We start at the second-to-last line of our work ($684 = 108 \cdot 6 + 36$) and write 36 as $36 = 684 - 108 \cdot 6$. We then use the line above it to write 108 as $108 = 1476 - 684 \cdot 2$ and substitute. We then simplify until we have our solution. The work is shown below:

$$\begin{aligned} 36 &= 684 - 108 \cdot 6 \\ &= 684 - (1476 - 684 \cdot 2) \cdot 6 \\ &= 684 - 6 \cdot 1476 + 12 \cdot 684 \\ 36 &= -6 \cdot 1476 + 13 \cdot 684 \end{aligned}$$

We have written $36 = (1476) \cdot (-6) + (684) \cdot (14)$.

3.3 Modular Arithmetic

Think back to when you were learning to tell time. After 12 o'clock comes 1 o'clock, not 13 o'clock. So, for example, instead of 18 o'clock, you say that it is 6 o'clock in the evening. What if the clock only had 9 numbers? Then 10 o'clock would really be 1 o'clock, 11 o'clock would be 2 o'clock, and so on. We can represent that last one by $9 + 2 \equiv 2$, and we can say that 11 is congruent to 2 on a 9—hour clock. This is exactly modular arithmetic.

Definition 33 (Congruent Modulo n)**.**
Let $n \in \mathbb{Z}_{\geq 1}$. $a, b \in \mathbb{Z}$ are **congruent modulo** *n if $a - b$ is divisible by n. n is called the* **modulus***, and we write*

$$a \equiv b (\mod n).$$

Example 3.3.1 (Modular Arithmetic)**.**
$16 \equiv 6 (\mod 5)$, *since* 5 *divides* $16 - 6 = 10$.
However, $19 \not\equiv 4 (\mod 7)$ *since* 7 *does not divide* $19 - 4 = 15$.

Example 3.3.2 (Modular Arithmetic in Sage)**.**
We can evaluate modular arithmetic using Sage in two different ways:

```
sage: 14 % 4
2
sage: mod(14,4)
2
```

Properties of Modular Arithmetic

1. If $a \equiv b (\mod n)$ and $c \equiv d (\mod n)$, then

$$a \pm c \equiv b \pm d (\mod n) \quad \text{and} \quad a \cdot c \equiv b \cdot d (\mod n)$$

2. There exists $b \in \mathbb{Z}$ such that $a \cdot b \equiv 1 (\mod n)$ if and only if $\gcd(a, n) = 1$

Remark 3.3.1. *In (2) above, we say that b is a multiplicative inverse of $a (\mod n)$. Moreover, if $a \cdot b_1 \equiv a \cdot b_2 \equiv 1 (\mod n)$, then $b_1 \equiv b_2 (\mod n)$.*

3.4 Groups

Definition 34 (Binary Operation)**.**
A **binary operation** $\star$ *on set G is a function $\star(\cdot, \cdot)$ that takes two elements of G to another element of G. Instead of writing $\star(f, g)$, we will write $f \star g$ for a binary operation on $f, g \in G$.*

Definition 35 (Group)**.**
A **group** *is a set G, together with a binary operation $\star$, such that for all $f, g, h \in G$,*

1. *(Closure under $\star$) $f \star g \in G$.*
2. *(Existence of an identity) there exits an element $e \in G$ such that $e \star f = f = f \star e$ for all $f \in G$.*
3. *(Existence of inverses) there exists an element $p \in G$ such that $f \star p = e = p \star f$. We call p the* **inverse** *of f in G, and we often write f^{-1}.*
4. *(Associativity) $(f \star g) \star h = f \star (g \star h)$.*

Remark 3.4.1. *We can show that an identity element is unique, so we can refer to it as <u>the</u> identity element in a group. We will write fg for $f \star g$.*

Proof. Suppose that e and e' are both identities in group G. Then, we have that $ge = g$ and $e'g = g$ for all $g \in G$. In particular, these statements hold for $g = e'$ and $g = e$. Choose $g = e'$ in the first one to get that $e'e = e'$. Choose $g = e$ in the second one to get $e'e = e$. Since we have two expressions for $e'e$, we get that $e' = e$. □

Remark 3.4.2. *We use e and 1 interchangeably to denote the identity element.*

Definition 36 (Abelian Group)**.**
A group G with operation $\star$ is **abelian** *(or* **commutative***) if, in addition to Definition 35, for all $f, g \in G$, $f \star g = g \star f$.*

Remark 3.4.3. *We will mostly deal with finite, abelian groups. We will be careful to specify in the case that we are not.*

In general, we will not use the notation $\star$ for an operation. Most frequently, we will use addition or multiplication as our operations.

Example 3.4.1 (Abelian Group Under Addition)**.**
The integers, $\mathbb{Z}$, is an abelian group under addition. In this group, the identity element is 0 and we denote the inverse of $g \in \mathbb{Z}$ as $-g$. However, the integers is not a group under multiplication, because the integer 2, for example, does not have a multiplicative inverse in the integers.

Example 3.4.2 (Group Under Multiplication)**.**
$\mathbb{R} - \{0\}$ is a group under multiplication with $e = 1$ and $g^{-1} = \frac{1}{g}$.

Example 3.4.3 (Group Under Matrix Multiplication)**.**
$GL(2,\mathbb{R}) = \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \mathbb{R}^{2\times 2} : ad - bc \neq 0 \right\}$ is the set of all 2×2 matrices with real entries whose determinant is nonzero. $GL(2,\mathbb{R})$ is a group under matrix multiplication. The identity element is the identity matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. The inverse of an element in $GL(2,\mathbb{R})$ is $\frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$.

However, the set of all 2×2 matrices is not a group under matrix multiplication, because the inverse of a matrix does not exist when it has a zero determinant.

Theorem 3.4.1. *Left and right cancellation hold in a group. That is, for all $a, b, c \in G$, $ba = ca$ implies that $b = c$ and $ab = ac$ implies that $b = c$.*

Proof. Suppose that $ba = ca$. Since inverses exist for all group elements, we have that $baa^{-1} = caa^{-1}$, or $be = ce$, so $b = c$. A similar argument gives us the other part of this claim. □

Theorem 3.4.2. *Let G be a group. For all $g \in G$, the inverse of g is unique. That is, for each $g \in G$, there exists a unique $f \in G$ such that $fg = gf = e$.*

Proof. Suppose, for the sake of contradiction, that f and h are both inverses of $g \in G$. Then $fg = e$ and $hg = e$. Since we have two expressions for e, we can equate them, getting $hg = fg$. By our cancellation theorem, we get that $h = f$. □

Theorem 3.4.3. *For f, g in group G, $(fg)^{-1} = g^{-1}f^{-1}$.*

Proof. We know that $(fg)^{-1}$ is the inverse of (fg), so we have that $(fg)(fg)^{-1} = e$. We can also see that $(fg)(g^{-1}f^{-1}) = fgg^{-1}f^{-1} = fef^{-1} = ff^{-1} = e$. Now that we have two expressions for e, we can equate them to see that $(fg)(fg)^{-1} = (fg)(g^{-1}f^{-1})$. By cancellation, we have that $(fg)^{-1} = g^{-1}f^{-1}$. □

Definition 37 (Group Order)**.**
The **order** *of group G is the number of elements in G. We denote the order of G by $|G|$.*

If $|G|$ is finite, we say that G is finite. Otherwise, we say that G is infinite.

Example 3.4.4 (Group Order)**.**
Consider the group $G = \{1,3,7,9\}$ under multiplication module 10. This has order 4, since there are 4 elements in the group.

We often care about repeated addition or repeated multiplication. We define those as follows:

$$mg = m \cdot g \overset{\text{def}}{=} \underbrace{g + g + \cdots + g}_{m \text{ times}} \qquad \text{and} \qquad g^m \overset{\text{def}}{=} \underbrace{g \cdot g \cdots g}_{m \text{ times}}$$

Definition 38 (Subgroup)**.**
H is a subgroup of group G if $H \subseteq G$ and H is a group. We write $H \leq G$.

Theorem 3.4.4. *Let G be a group and $H \subseteq G$ be nonempty. Then $H \leq G$ if H is closed under the group operation and closed under taking inverses. That is, $H \leq G$ if for all $a, b \in H$, $ab^{-1} \in H$.*

Proof. Since H and G have the same operation, and it is associative in G, it must also be associative in H. Since we know, by assumption, that H is nonempty, we have that there exists some $h \in H$. We know that all elements have an inverse, so h^{-1} exists as well, giving us $hh^{-1} = e$. So, the identity is in H. We know that $h^{-1} \in H$ when $h \in H$ because $eh^{-1} \in H$ by assumption. And, since we have shown that $h^{-1} \in H$ whenever $h \in H$, we have that $g(h^{-1})^{-1} = gh \in H$, so H is closed. Thus, $H \leq G$. □

So, in order to prove that H is a subgroup of G, it suffices to show that

1. H is nonempty
2. $f, h \in h \implies fh \in H$
3. $h^{-1} \in H$ for all $h \in H$

You can combine statements 2 and 3 into one statement and show that $fh^{-1} \in H$.

Theorem 3.4.5. *Let $\langle a \rangle = \{a^n : n \in \mathbb{Z}\}$. Then $\langle a \rangle$ is a subgroup of G for all $a \in G$.*

Proof. Since $a \in \langle a \rangle$, we have that $\langle a \rangle$ is nonempty. Let $a^n, a^m \in \langle a \rangle$. Then $a^n(a^m)^{-1} = a^n a^{-m} = a^{n-m}$, which is a power of a. So, $a^n(a^m)^{-1} \in \langle a \rangle$. Thus, $\langle a \rangle$ is a subgroup of G. □

Remark 3.4.4. *$\langle a \rangle$ is called the* **cyclic** *subgroup of G generated by a. If we have that $G = \langle a \rangle$, then we say that G is a* **cyclic group** *and that a is a* **generator** *of G.*

Example 3.4.5 (Cyclic Group Under Multiplication Modulo 10)**.**
Consider again the group $G = \{1,3,7,9\}$ under multiplication modulo 10. Note that $3^1 = 1$, $3^2 = 9$, $3^3 = 7$, $3^4 = 1$, $3^5 = 3$, $3^6 = 9, \ldots$. So, $G = \langle 3 \rangle$, and G is a cyclic group with generator 3.

Example 3.4.6 (Cyclic Group Under Addition Modulo n)**.**
The set $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ for $n \geq 1$ is a cyclic group under addition modulo n. 1 and $-1 = n-1$ are generators.

Example 3.4.7 (Generators of $\mathbb{Z}$ and $\mathbb{Z}_n$)**.**
$\mathbb{Z}$ has two generators, 1 and -1. $\mathbb{Z}_n$ can have multiple generators. For example, $\mathbb{Z}_8 = \langle 1 \rangle = \langle 3 \rangle = \langle 5 \rangle = \langle 7 \rangle$.

We pause to talk more about $\mathbb{Z}_n$. We note that many texts use the notation $\mathbb{Z}/(n\mathbb{Z})$ here. We use $\mathbb{Z}_n$ to denote the ring of integers modulo n. We discuss this a little more in future sections, but we feel that it is important to talk about $\mathbb{Z}_n$ now. If you have taken an Algebra course, you may recognize that $\mathbb{Z}/n\mathbb{Z}$ is the quotient ring of $\mathbb{Z}$ by principal ideal $n\mathbb{Z}$, and that $0, 1, 2, \ldots, n-1$ are the coset representatives for congruence classes. For us, it is only important right now that we understand the basics that we need for cryptosystems. Addition and multiplication are done as "usual" with integers, and the results are reduced modulo n. You can construct the addition and multiplication tables for $\mathbb{Z}_n$ using Sage.

Example 3.4.8 (Addition and Multiplication Tables)**.**
We show an example of the addition and multiplication tables for $\mathbb{Z}_8$ with SageMath.

```
sage: from sage.matrix.operation_table import OperationTable
sage: G = Integers(8)
sage: OperationTable(G, operator.add, names='digits')
+  0 1 2 3 4 5 6 7
 +---------------
0| 0 1 2 3 4 5 6 7
1| 1 2 3 4 5 6 7 0
2| 2 3 4 5 6 7 0 1
3| 3 4 5 6 7 0 1 2
4| 4 5 6 7 0 1 2 3
5| 5 6 7 0 1 2 3 4
6| 6 7 0 1 2 3 4 5
7| 7 0 1 2 3 4 5 6

sage: OperationTable(G, operator.mul, names='digits')
*  0 1 2 3 4 5 6 7
 +---------------
0| 0 0 0 0 0 0 0 0
1| 0 1 2 3 4 5 6 7
2| 0 2 4 6 0 2 4 6
3| 0 3 6 1 4 7 2 5
4| 0 4 0 4 0 4 0 4
5| 0 5 2 7 4 1 6 3
6| 0 6 4 2 0 6 4 2
7| 0 7 6 5 4 3 2 1
```

Definition 39 (Unit)**.**
$a \in \mathbb{Z}_n$ is called a **unit** *of $\mathbb{Z}_n$ if there exists some $b \in \mathbb{Z}_n$ such that $ab = 1$. In other words, a unit in $\mathbb{Z}_n$ is an element that has a multiplicative inverse in $\mathbb{Z}_n$.*

Example 3.4.9 (Unit)**.**
Consider $\mathbb{Z}_{14}$. If we take the operation to be multiplication, the set of units in $\mathbb{Z}_{14}$ is $\{1,3,5,9,11,13\}$. For example,

$$11 \cdot 9 = 99 \equiv 1 (\mod 14) \qquad \textit{so} \qquad 11^{-1} \equiv 9 (\mod 14).$$

We often call this set U_{14} or $\mathbb{Z}_{14}^*$ (see exercise 6).

Remark 3.4.5. Euler's Totient Function *(also called the Phi (ϕ) Function) counts the number of positive integers less than n that are relatively prime to n. It gives the order of $\mathbb{Z}_n^*$. $\phi(n)$ is used in Cryptography in the RSA Cryptosystem.*

Example 3.4.10 (Euler's Totient Function)**.**
For example, $\phi(7) = 6$ and $\phi(8) = 4$.

Remark 3.4.6. *When p is prime, every nonzero element of $\mathbb{Z}_p$ is a unit. This means that $\mathbb{Z}_p^* = \{1, 2, 3, \ldots, p-1\}$, and every element in $\mathbb{Z}_p^*$ has a multiplicative inverse.*

When p is prime, $\mathbb{Z}_p$ is an example of a **field**. *Other fields you might be familiar with are $\mathbb{R}, \mathbb{Q}$, and $\mathbb{C}$. A field is a commutative ring in which every nonzero element has a multiplicative inverse. Since $\mathbb{Z}_p$ has a finite number of elements, we call $\mathbb{Z}_p$ a* **finite field**, *so we sometimes denote it as $\mathbb{F}_p$. Similarly, we sometimes denote $\mathbb{Z}_P^*$ as $\mathbb{F}_p^*$. When we write $\mathbb{Z}_p$, we use equivalence ($\equiv$), and when we write $\mathbb{F}_p$, we use equality ($=$). We remark that finite fields are sometimes called Galois Fields, so $\mathbb{F}_p$ is sometimes written as $GF(p)$. We will see this in some of our Sage examples.*

Definition 40 (Zero Divisor)**.**
A nonzero element $a \in \mathbb{Z}_n$ is called a **zero divisor** *if there exists a nonzero element $c \in \mathbb{Z}_n$ such that $ac = 0$.*

Example 3.4.11 (Zero Divisor)**.**
Consider $\mathbb{Z}_{15}$. The element 3 is a zero divisor in $\mathbb{Z}_{15}$ because $3 \cdot 5 \equiv 0 (\mod 15)$. 5 is also a zero divisor for the same reason.

Definition 41 (Order)**.**
The **order** *of a modulo p is the smallest positive integer k such that $a^k \equiv 1 (\mod p)$.*

Proposition 3.4.1. *The order of the subgroup $\langle g \rangle$ is the smallest positive m for which $g^m = e$. If such an m does not exist, we say that the order is infinite.*

Proof. Let m be finite. We claim that $\langle g \rangle = \{e, g, g^2, g^3, \ldots, g^{m-1}\}$. For any $n \in \mathbb{Z}$, we can write $n = qm + r$, where $0 \leq r \leq m-1$. So,

$$\begin{aligned} g^n &= g^{qm+r} \\ &= g^{qm} g^r \\ &= (g^m)^q g^r \\ &= e g^r \\ &= g^r \end{aligned}$$

We also claim that that all of the elements $e, g, g^2, g^3, \ldots, g^{m-1}$ are distinct. Suppose, for the sake of contradiction, that $g^i = g^j$ for some $0 \leq i < j \leq m-1$. Then $g^{i-j} = e$. This contradicts that m is the smallest integer such that $g^m = e$. □

Theorem 3.4.6 (Lagrange's Theorem)**.**
If G is a finite group and H is a subgroup of G, then $|H|$ divides $|G|$.

Remark 3.4.7. *The proof of Lagrange's Theorem involves the use of cosets. We omit the proof, but encourage the reader to see any standard undergraduate Abstract Algebra textbook for a discussion.*

Corollary 3.4.6.1. *If G is a finite group and $g \in G$, then the order of g divides the order of G. That is, $|g| \Big| |G|$. In particular, $g^{|G|} = e$ for all $g \in G$.*

Proof. First, recall that $|g| = |\langle g \rangle|$. So by Lagrange's Theorem, we have $|G| = |g| \cdot k$ for some $k \in \mathbb{Z}_{>0}$. So, we have that $g^{|G|} = g^{|g| \cdot k} = e^k = e$. □

Theorem 3.4.7 (Fermat's Theorem)**.**
If $a, p \in \mathbb{Z}$ with p prime and $p \nmid a$, then $a^{p-1} \equiv 1 (\mod p)$.

Proof. Assume that p is prime. Then the group of units in $\mathbb{Z}_p$ is $\mathbb{Z}_p^* = \{1, 2, 3, \ldots, p-1\}$ by Remark 3.1.1. So, we have that $|\mathbb{Z}_p^*| = p-1$. Let $a \in \mathbb{Z}_p^*$ have order k. That is, k is the smallest positive integer such that $a^k \equiv 1 (\mod p)$. Then $\{1, a, a^2, a^3, \ldots, a^{k-1}\}$ form a subgroup of $\mathbb{Z}_p^*$ when reduced modulo p. By Lagrange's Theorem, $k \Big| |\mathbb{Z}_p^*|$. That is, $p - 1 = km$ for some $m \in \mathbb{Z}_{>0}$. Then we have $a^{p-1} \equiv a^{km} \equiv (a^k)^m \equiv 1^m \equiv 1 (\mod p)$. □

Corollary 3.4.7.1. *If $a, p \in \mathbb{Z}$ with p prime and $p \nmid a$, then $a^p \equiv a (\mod p)$.*

Proof. This follows directly from Fermat's Theorem. □

The most important topic for studying more modern cryptography is the theory of finite fields of prime order. We will introduce enough algebra to get to a working understanding of the topic.

3.5 Rings

Definition 42 (Ring)**.**
A nonempty set R equipped with two operations, addition and multiplication, is a **ring** *if it satisfies each of the following properties for all $a, b, c \in R$:*

1. *(Closure under addition) $a, b \in R \implies a + b \in R$*

2. *(Associativity of addition)* $a+(b+c)=(a+b)+c$

3. *(Commutativity of addition)* $a+b=b+a$

4. *(Existence of additive identity) There exists* $0_R \in R$ *such that* $a+0_R=a=0_R+a$

5. *(Existence of additive inverse) There exists* $a\, d \in R$ *such that* $a+d=0_R$

6. *(Closure under multiplication)* $a, b \in R \implies ab \in R$

7. *(Associativity of multiplication)* $a(bc)=(ab)c$

8. *(Left Distributivity)* $a(b+c)=ab+ac$

9. *(Right Distributivity)* $(a+b)c=ac+bc$

Remark 3.5.1. *Axioms* $1-5$ *of a ring just say that a ring is an abelian group under addition.*

Definition 43 (Commutative Ring)**.**
If, in addition to $1-9$*,* $ab=ba$ *for all* $a, b \in R$*, we call* R *a* **commutative ring**. *(This is commutativity of multiplication.)*

Definition 44 (Ring with Identity)**.**
If, in addition to $1-9$*, there exists a* $1_R \in R$ *such that* $a1_R=a=1_Ra$ *for all* $a \in R$*, then we call* R *a* **ring with identity**. *(This is the existence of a multiplicative identity.)*

Example 3.5.1 (Ring with Identity)**.**
$\mathbb{Z}$ *and* $\mathbb{R}$ *with the usual multiplication and addition are commutative rings with identity.* $\mathbb{Z}_n$ *is also a commutative ring with identity under the usual addition and multiplication.*

Example 3.5.2 (Ring)**.**
Let $\mathscr{M}_{2\times 2}(\mathbb{R})=\left\{\begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \mathbb{R}^{2\times 2}\right\}$ *be the set of all* 2×2 *matrices with real entries.* $\mathscr{M}_{2\times 2}(\mathbb{R})$ *is a ring with identity, but it is not commutative. The zero matrix is the additive identity, and the identity matrix is the multiplicative identity. However, matrix multiplication is not commutative.*

Remark 3.5.2. *We will most often deal with commutative rings with identity, and we will be careful to specify if that is not the case. From this point forward, it is safe to assume that "ring" means "commutative ring with identity" unless otherwise specified.*

Definition 45 (Divides)**.**
Let R *be a ring. Let* $a, b \in R$ *with* $b \neq 0$*. We say that "*b **divides** a*" and write* $b|a$ *if there exists* $a\, c \in R$ *such that* $a=bc$*. This is the same as in* $\mathbb{Z}$*.*

Definition 46 (Unit)**.**
Let R be a ring. An element $u \in R$ is called a **unit** *if it has a multiplicative inverse. An element $a \in R$ is called* **irreducible** *if a is not a unit and if $a = bc$, then either b or c is a unit.*

Definition 47 (Congruent Modulo n)**.**
Let R be a ring, and let $n \in R$. Then $a, b \in R$ are **congruent modulo** *n if $n|(a-b)$. As before, we write $a \equiv b(\mod n)$.*

Again as before, we have that if $a \equiv b(\mod n)$ and $c \equiv d(\mod n)$, then

$$a \pm c \equiv b \pm d(\mod n) \quad \textit{and} \quad a \cdot c \equiv b \cdot d(\mod n)$$

Definition 48 (Congruence Class)**.**
Let R be a ring, and let $n \in R$ with $n \neq 0$. Then for any $a \in R$, the set of all $a' \in R$ that are congruent modula n to a is denoted by $\overline{a}$. The set of $\overline{a}$ is called the **congruence class of** *a, and we denote the set of all congruence classes by $R/(n)$ or R/nR. We call this a quotient ring of R by n. That is,*

$$R/nR = \{\overline{a} : a \in R\} = \{a' : a' \equiv a(\mod n) \textit{ for } a \in R\}.$$

Addition an multiplication work just like we would hope that they would. That is,

$$\overline{a} + \overline{b} = \overline{a+b} \quad \textit{and} \quad \overline{a} \cdot \overline{b} = \overline{a \cdot b}.$$

Definition 49 (Subring)**.**
Let R be a ring. A nonempmty subset $S \subseteq R$ is a **subring** *of R if it is a ring itself.*

Theorem 3.5.1. *Let R be a ring and $S \subseteq R$ be nonempty. Then S is a subring of R if*

- *S is closed under addition ($a, b \in S \implies a + b \in S$),*
- *S is closed under multiplication ($a, b \in S \implies ab \in S$),*
- *S contains R's additive identity ($0_R \in S$), and*
- *S contains the additive inverse of all $a \in S$ ($(-a) \in S$ for all $a \in S$)*

Theorem 3.5.2. *The additive inverse of $a \in R$ is unique. That is, $a + x = 0_R$ has a unique solution for x in R.*

Proof. Let $a \in R$. By axiom 5, we know that an additive inverse of a exists. That is, we know that $a + x = 0_R$ has a solution in R. Suppose to the contrary that b and c both satisfy this equation. Then $a + b = 0_R$ and $a + c = 0_R$. Observe

$$c = 0_R + c = (a + b) + c = a + b + c = b + a + c = b + (a + c) = b + 0_R = b.$$

We have that $b = c$, so the additive inverse is unique. □

We often write $-a$ for the additive inverse of $a \in R$. This gives us the notion for subtraction.

Theorem 3.5.3. *For $a, b, c \in R$, if $a + b = a + c$, then $b = c$.*

Proof.

$$\begin{aligned} a + b &= a + c \\ -a + (a + b) &= -a + (a + c) \\ (-a + a) + b &= (-a + a) + c \\ 0_R + b &= 0_R + c \\ b &= c \end{aligned}$$

□

The nice thing is that all of the properties we are used to using "work" in a ring. We list them below.

Properties of Ring R: Let R be a ring, and let $a, b \in R$. Then

1. $a \cdot 0_R = 0_R = 0_R \cdot a$
2. $a(-b) = -ab$ and $(-a)b = -ab$
3. $-(-a) = a$
4. $-(a + b) = (-a) + (-b)$
5. $-(a - b) = -a + b$
6. $(-a)(-b) = ab$

If R has identity $1_R \neq 0_R$, then we also have that $(-1_R)a = -a$.

With all this in mind, we now begin to look at Polynomial Arithmetic, which will be very important in the NTRU Cryptosystem later on.

If R is any ring, we can form a ring of polynomials where the coefficients are taken from the ring R. We denote the polynomial ring by

$$R[x] = \{a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n : n \geq 0; a_0, a_1, a_2, \ldots, a_n \in R\}$$

The elements of $R[x]$ described above are called **polynomials with coefficients in** R, where the a_i are the **coefficients** from R.

Example 3.5.3 (Polynomials)**.**
You are likely already familiar with the ring $\mathbb{Z}[x]$*. An example of a polynomial in* $\mathbb{Z}[x]$ *is* $2+3x-x^2$*, because all of the coefficients are in* $\mathbb{Z}$*. The polynomials* $1-0.5x^2$ *is not, however, in* $\mathbb{Z}[x]$*, since* $0.5 \notin \mathbb{Z}$*. But,* $1-0.5x^2 \in \mathbb{R}[x]$*.*

Example 3.5.4 (Polynomial)**.**
The polynomial $x+2x^2 \in \mathbb{Z}_3[x]$*, since it has coefficients in* $\mathbb{Z}_3$*.*

To enter this into Sage, we first need to define the ring.

```
sage: R.<x> = PolynomialRing(GF(3),x)
sage: f = x + 2*x^2
```

Definition 50 (Leading Coefficient and Degree)**.**
The **leading coefficient** *of a nonzero polynomial* $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ *is* $a_n \neq 0$*, and the* **degree** *is* n*. In other words, the degree is the largest power of* x *that occurs with a nonzero coefficient, and the leading coefficient is that nonzero coefficient.*

Definition 51 (Monic)**.**
A nonzero polynomial with a leading coefficient of 1 *is called a* **monic** *polynomial.*

Example 3.5.5 (Degree and Leading Coefficient)**.**
$f(x) = 3-2x+4x^2+7x^3-x^4$ *has degree* 4 *and leading coefficient* -1*. We write* $\deg(f(x)) = 4$ *for the degree. Of course, if the polynomial is long or not in standard form, or if we need to use the degree in another part of our code, we might want to ask Sage to find these for us.*

```
sage: R.<x> = PolynomialRing(ZZ,x)
sage: f = 3 - 2*x + 4*x^2 + 7*x^3 - x^4
sage: f.degree()
```

```
4

sage: f.leading_coefficient()
-1
```

Arithmetic with polynomials in $R[x]$ is exactly what we are used to, and it follows from the fact that $R[x]$ is a ring. First, we have polynomial addition defined as follows:

$$\begin{aligned}&(a_0+a_1x+a_2x^2+\cdots+a_nx^n)+(b_0+b_1x+b_2x^2+\cdots+b_nx^n)\\&=(a_0+b_0)+(a_1+b_1)x+(a_2+b_2)x^2+\cdots+(a_n+b_n)x^n\end{aligned}$$

We also have that polynomial multiplication is defined as follows:

$$\begin{aligned}&(a_0+a_1x+a_2x^2+\cdots+a_nx^n)(b_0+b_1x+b_2x^2+\cdots+b_mx^m)\\&=a_0b_0+(a_0b_1+a_1b_0)x+(a_0b_2+a_1b_1+a_2b_0)x^2+\cdots+a_nb_mx^{n+m}\end{aligned}$$

where the coefficient of x^k for all $k\geq 0$ is $\displaystyle\sum_{i=0}^{k}a_1b_{k-i}$.

Example 3.5.6 (Polynomial Arithmetic)**.**
In $\mathbb{Z}_3[x]$*, let* $g(x)=2x-x^2+x^3$ *and* $f(x)=-1+2x-2x^2+x^3$*. Then*

$$\begin{aligned}g(x)+f(x)&=(2x-x^2+x^3)+(-1+2x-2x^2+x^3)\\&=-1+(2+2)x+(-1-2)x^2+(1+1)x^3\\&=-1+4x-3x^2+2x^3\\&=-1+x+2x^3\\&=2+x+2x^3\end{aligned}$$

Note that we computed the addition of the coefficients first, and then we reduced them modulo 3*. To do this in Sage, we do the following:*

```
sage: R.<x> = PolynomialRing(GF(3),x)
      f = 2*x - x^2 + x^3
      g = -1 + 2*x - 2*x^2 + x^3
sage: g + f
2*x^3 + x + 2
```

We now compute the product of $g(x)$ *and* $f(x)$*.*

$$\begin{aligned} g(x)\cdot f(x) &= (2x-x^2+x^3)(-1+2x-2x^2+x^3) \\ &= -2x+4x^2-4x^3+2x^4+x^2-2x^3+2x^4-x^5-x^3+2x^4-2x^5+x^6 \\ &= -2x+5x^2-7x^3+6x^4-3x^5+x^6 \\ &= x+2x^2+2x^3+x^6 \end{aligned}$$

Notice that we computed the product normally, and then we reduced the coefficients modulo 3*. We could also compute the product in Sage:*

```
sage: g*f
x^6 + 2*x^3 + 2*x^2 + x
```

Remark 3.5.3. *If R is commutative, then so is $R[x]$. Likewise, if R has multiplicative identity 1_R, then 1_R is also the multiplicative identity of $R[x]$.*

3.6 Fields

Definition 52 (Integral Domain)**.**
An **integral domain** *is a commutative ring R with identity $1_R \neq 0$ such that if $a\,b = 0_R$, then $a = 0_R$ or $b = 0_R$ for all $a, b \in R$. That is, there are no zero divisors.*

Definition 53 (Field)**.**
A **field** *is a commutative ring R with identity $1_R \neq 0$ such that every nonzero element in R has a multiplicative inverse in R.*

Theorem 3.6.1. *Let R be an integral domain, and let $f(x), g(x) \in R[x]$ be nonzero. Then* $\deg(f(x)\cdot g(x)) = \deg f(x) + \deg(g(x))$.

Proof. Let $f(x), g(x) \in R[x]$ be given by $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ and $g(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_m x^m$, where $a_n \neq 0_R$ and $b_m \neq 0_R$. That is, $\deg(f(x)) = n$ and $\deg(g(x)) = m$. By definition, the largest exponent that $f(x)g(x)$ can have is $n+m$. We know that $a_n \neq 0_R$ and $b_m \neq 0_R$. Since R is an integral domain, $a_n b_m \neq 0_R$. So, $f(x)g(x)$ is nonzero and has degree $n+m = \deg(f(x)) + \deg(g(x))$. □

Corollary 3.6.1.1. *If R is an integral domain, so is $R[x]$.*

Corollary 3.6.1.2. *Let R be an integral domain, and let $f(x) \in R[x]$. Then $f(x)$ is a unit in $R[x]$ if and only if $f(x)$ is a unit in R. If $\mathbb{F}$ is a field, the units in $\mathbb{F}[x]$ are the nonzero constants in $\mathbb{F}$.*

Recall the division algorithm you used in prior math classes to find, say, that $\frac{2x^2-3x+1}{x+1} = 2x-5+\frac{6}{x+1}$. Another way to write this is $2x^2-3x+1=(2x-5)(x+1)+6$. This process is the division algorithm with a remainder. We can perform this algorithm on any polynomial ring $\mathbb{F}[x]$ provided that $\mathbb{F}$ is a field. Rings that we are allowed to perform this algorithm on are called **Euclidean Domains**.

We say that $\mathbb{F}[x]$ is Euclidean. Let $\mathbb{F}$ be a field, and let $a(x), b(x) \in \mathbb{F}[x]$ with $b(x) \neq 0$. Then we can write $a(x) = b(x) \cdot q(x) + r(x)$ with $0 \leq \deg(r(x)) < \deg(b(x))$. We call $q(x)$ the **quotient** and $r(x)$ the **remainder**.

Definition 54 (Common Divisor)**.**
As one would expect, we can define a common divisor in $\mathbb{F}[x]$. *A* **common divisor** *of* $f(x), g(x) \in \mathbb{F}[x]$ *is* $d(x) \in \mathbb{F}[x]$ *that divides both* $f(x)$ *and* $g(x)$. $d(x)$ *is a* **greatest common divisor** *(or gcd) of* $f(x)$ *and* $g(x)$ *if it is a divisor of highest degree.*

Euclidean Algorithm
Let $\mathbb{F}$ be a field, and let $f(x)$ and $g(x)$ be in $\mathbb{F}[x]$ with $g(x) \neq 0$. Then we can perform the Euclidean Algorithm to find $\gcd(f(x), g(x))$ as follows:

$$\begin{aligned}
f(x) &= g(x) \cdot q_1(x) + r_1(x) \\
g(x) &= r_1(x) \cdot q_2(x) + r_2(x) \\
r_1(x) &= r_2(x) \cdot q_3(x) + r_3(x) \\
r_2(x) &= r_3(x) \cdot q_4(x) + r_4(x) \\
&\vdots \\
r_{n-2}(x) &= r_{n-1}(x) \cdot q_n(x) + r_n(x) \\
r_{n-1}(x) &= r_n(x) \cdot q_{n+1}(x)
\end{aligned}$$

At each step, $0 \leq \deg(r_{i-1}(x)) < \deg(r_i)$. $\gcd(f(x), g(x)) = r_n(x)$, the last nonzero remainder.

The **Extended Euclidean Algorithm** works the same way it did with integers. It says that there exists $u(x), v(x) \in \mathbb{F}[x]$ such that $f(x)u(x) + g(x)v(x) = \gcd(f(x), g(x))$.

The next proposition and example come from [14]. You can see the details of the proof there.

Proposition 3.6.1. *Let* $\mathbb{F}$ *be a field and* $p(x) \in \mathbb{F}[x]$ *such that* $p(x) \neq 0$. *Then every nonzero congruence class* $\overline{a(x)} \in \mathbb{F}[x]/(p(x))$ *has a unique representative* $r(x)$ *satisfying* $\deg(r(x)) < \deg(p(x))$ *and* $a(x) \equiv r(x) (\mod p(x))$.

Example 3.6.1 (Addition and Multiplication in a Ring)**.**
Consider the ring $\mathbb{F}[x]/(x^2+1)$*. Every element in this quotient ring is uniquely represented by a polynomial of the form* $a+b\,x$ *with* $a,b\in\mathbb{F}$*. We perform addition in the usual way:*

$$\overline{a_1+b_1x}+\overline{a_2+b_2x}=\overline{(a_1+a_2)+(b_1+b_2)x}.$$

Multiplication starts off in the usual way:

$$\overline{a_1+b_1x}\cdot\overline{a_2+b_2x}=\overline{a_1a_2+(a_1b_2+a_2b_1)x+b_1b_2x^2}$$

but we need to divide by x^2+1 *and take the remainder. This gives us*

$$\overline{a_1+b_1x}\cdot\overline{a_2+b_2x}=\overline{(a_1a_2-b_1b_2)+(a_1b_2+a_2b_2)x}.$$

Observe that dividing by x^2+1 *is the same as replacing* x^2 *with* -1*. The idea here is that we can think of making* $x^2+1=0$*, so we just replace all of the* x^2 *terms with* -1*.*

Example 3.6.2 (Polynomial Multiplication)**.**
Let's consider $\mathbb{Z}[x]/(x^3-1)$*. We will find the product of* $f(x)=2x^2+x-1$ *and* $g(x)=x^2+2$*.*

$$\begin{aligned}
f(x)g(x)&=(2x^2+x-1)(x^2+2)\\
&=2x^4+x^3-3x^2+4x^2+2x-6\\
&=2x^4+x^3+x^2+2x-6\\
&=2x^3\cdot x+x^3+x^2+2x-6\\
&=2(1)\cdot x+(1)+x^2+2x-6\\
&=2x+1+x^2+2x-6\\
&=x^2+4x-5
\end{aligned}$$

Of course, we can do this in Sage as well.

```
sage: R.<x> = PolynomialRing(ZZ,x)
sage: F.<x> = R.quotient_ring(x^3-1)
sage: f = F(2*x^2+x-3)
sage: g = F(x^2+2)
sage: f*g
x^2 + 4*x - 5
```

Example 3.6.3 (Polynomial Multiplication)**.** *Now, let's consider the same product, but this time in* $\mathbb{Z}_4[x]/(x^3-1)$*. We compute the product the exact same way that we did before. Then,*

we reduce the coefficients $\bmod 4$ *to get* $f(x)g(x) = x^2 + 1$. *In Sage, we could simply change our polynomial ring to have coefficients in* $\mathbb{Z}_4$ *instead of* $\mathbb{Z}$.

```
sage: R.<x> = PolynomialRing(GF(4),x)
sage: F.<x> = R.quotient_ring(x^3-1)
sage: f = F(2*x^2+x-3)
sage: g = F(x^2+2)
sage: f*g
x^2 + 1
```

3.7 Exercises

1. Use the Euclidean Algorithm to find gcd(42, 93) by hand.
2. Use the Euclidean Algorithm to find gcd(1533, 26187) by hand.
3. Find gcd(42823, 6409). Then, find integers x and y such that $42823x + 6409y = \gcd(42823, 6409)$.
4. Use the Euclidean algorithm to verify that 44 and 17 are relatively prime. Then, find x and y such that $44x + 17y = 1$.
5. Find a multiplicative inverse of 13 mod 35.
6. Let $\mathbb{Z}_n^*$ be the set of all positive integers less than $n > 1$ and relatively prime to n. In other words, $\mathbb{Z}_n^*$ is the set of units in $\mathbb{Z}_n$. Show that $\mathbb{Z}_n^*$ is a group under multiplication modulo n.
7. List the elements of $\mathbb{Z}_{12}^*$. Find the order of the group and the order of each element in the group.
8. Is $\mathbb{Z}_{14}^*$ cyclic? If so, what is/are its generator(s)?
9. List all (six) cyclic subgroups of $\mathbb{Z}_{15}^*$.
10. List the elements of $\langle 7 \rangle$ in $\mathbb{Z}_{18}^*$.
11. Let H and K be subgroups of group G. Show that $H \cap K$ is also a subgroup of G.
12. Show that an element and its inverse have the same order in any group.
13. Find all units in $\mathbb{Z}_7$ and $\mathbb{Z}_8$.
14. Find all zero divisors in $\mathbb{Z}_7$ and $\mathbb{Z}_8$.
15. Based on questions 13 and 14, what can you say about units and zero divisors in $\mathbb{Z}_n$? (It may be helpful to list out the units and zero divisors for a few more groups, such as $\mathbb{Z}_9$ and $\mathbb{Z}_{10}$.)

3.8 Computer Exercises

1. Write a Python code that performs the Extended Euclidean Algorithm to find x and y such that $a\,x + b\,y = 1$, where $\gcd(a, b) = 1$. Verify that your code works with your answer to numbers 3 and 4 above.

2. Verify your solutions to problems 1 and 2 above.

3. Generate an addition and multiplication table for $\mathbb{Z}_{12}$ and $\mathbb{Z}_{13}$.

4. Compute the sum and product of $f(x) = 2x^4 - x^3 + x - 2$ and $g(x) = -2x^3 + x^2 + 2x - 1$ in $\frac{\mathbb{Z}_3[x]}{x^5 - 1}$.

CHAPTER

4

PRE-QUANTUM CRYPTOSYSTEMS AND THEIR COMPUTATIONAL HARD PROBLEMS

4.1 RSA

In 1976, Whitfield Diffie and Martin Hellman famously introduced the concept of public key cryptography in [8]. Ron Rivest, Adi Shamir, and Leonard Adleman answered the call for such cryptosystems in 1978. [25] We will now discuss one of the oldest and most widely-used Public Key Cryptosystems to date, the RSA of Rivest, Shamir, and Adelman.

Key Creation

Alice chooses two distinct primes, p and q. These primes should be similar in magnitude but differ by a few digits. She then computes the public modulus, $n = pq$. Next, she computes $\phi(n) = (p-1)(q-1)$. Note that this is Euler's Totient function. Alice then chooses an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$. e is called the *encryption exponent*. Alice will keep her *private key* (p, q) a secret and publish her *public key* (n, e).

Encryption

Suppose Bob wants to send the *plaintext message* m to Alice. m should be an integer with $0 \leq m < n$. To encrypt the the message, Bob computes $c \equiv m^e (\text{mod } n)$. Notice that he uses Alice's encryption exponent, e, to do this. He then sends the *ciphertext* c to Alice.

Decryption

Alice receives c from Bob, and she wants to recover m. She first computes the *decryption exponent* $d \equiv e^{-1} (\text{mod } \phi(n))$. In order to calculate d, Alice should use the Euclidean Algorithm to find d such that $de \equiv 1 (\text{mod } \phi(n))$. To recover the plaintext, Alice raises the ciphertext to the power of the decryption exponent, modulo n. In other words, she computes $m' \equiv c^d (\text{mod } n)$. m' is exactly the plaintext message m.

It remains to show that $m' = m$. In order to do so, we first need another theorem.

Theorem 4.1.1 (Euler's Theorem)**.**
If $\gcd(a,n) = 1$*, then* $a^{\phi(n)} \equiv 1 (\mod n)$.

Here, $\phi(n)$ *is Euler's totient function, which is the number of integers in* $\{1, 2, \ldots, n-1\}$ *that are relatively prime to* n*, as we saw in the review of abstract algebra.*

Proof. The units in $\mathbb{Z}_n$ form a group under multiplication. We denote this with $\mathbb{Z}_n^*$. Since $\gcd(a,n) = 1$, we know that a is in this group. This group has $\phi(n)$ elements. The subgroup generated by a is $\{a^n : n \in \mathbb{Z}\} = \{1, a, a^2, \ldots, a^{m-1}\}$ and has order m. By Lagrange's Theorem, $m | \phi(n)$. That is, $\phi(n) = km$ for some $k \in \mathbb{Z}$. Then we have that

$$a^{\phi(n)} \equiv a^{mk} \equiv (a^m)^k \equiv 1^k \equiv 1 (\mod n).$$

□

Remark 4.1.1. *When* n *is prime, Theorem 4.1.1 is just Fermat's Little Theorem: Let* p *be a prime which does not divide the integer* a*. Then* $a^{p-1} \equiv 1 (\mod p)$.

Proposition 4.1.1. *In the RSA Cryptosystem,* $m' = m$*. That is,* $m' = c^d (mod\ n) = m$

Proof.

$$
\begin{aligned}
m' &\equiv c^d \pmod{n} \\
&= (m^e)^d \pmod{n} \\
&= m^{ed} \pmod{n} \\
&= m^{1+k\phi(n)} \pmod{n} && \text{(for some } k \in \mathbb{Z}_{\geq 0}\text{)} \\
&= m(m^{\phi(n)})^k \pmod{n} \\
&= m \pmod{n} && \text{(Euler's Theorem)} \\
m' &= m
\end{aligned}
$$

□

Example 4.1.1 (RSA)**.**
We will illustrate the RSA with a toy example. In practice, primes p and q are much larger.

Key Creation
Suppose that Alice chooses distinct primes $p = 3$ and $q = 11$. Note that these are similar in magnitude. She computes the public modulus $n = 3 \times 11 = 33$ and her secret $\phi(n) = 2 \times 10 = 20$. She then chooses an encryption exponent $e = 7$. Note that $1 < 7 < 20$ and $gcd(7,20) = 1$. Alice publishes $(n,e) = (33,7)$ and keeps $(p,q) = (3,11)$ private.

Encryption
Suppose that Bob wants to send the plaintext message $m = 18$ to Alice. Note that he has chosen m such that $1 \leq m < n$. He computes $c \equiv 18^7 (mod\, 33) = 6$ and sends ciphertext $c = 6$ to Alice.

Decryption
Alice receives $c = 6$ from Bob. She first computes her decryption exponent $d \equiv 7^{-1} (mod\, 20) = 3$. Note that $3 \cdot 7 \equiv 1 (\mod 20)$. She uses d to compute $m' \equiv 6^3 (mod\, 33) = 18$. Note that Alice recovered the plaintext message.

Example 4.1.2 (RSA with Sage)**.**
We will work through another example of RSA with SageMath.

Key Creation
Alice chooses $p = 499$ and $q = 643$. Recall that you can check that these are prime. She then

computes $n = pq = 320857$ *and* $\phi(n) = (p-1)(q-1) = 319716$ *and chooses encryption exponent* $e = 71$. *She checks that* $\gcd(71, 319716) = 1$.

```
sage: p = 499
sage: q = 643
sage: n = p*q
sage: phi = (p-1)*(q-1)
sage: e = 71
sage: gcd(e, phi)
1
```

Encryption
Bob wants to send a message $m = 327$ *to Alice. He computes* $c \equiv m^e \pmod{320857} = 142145$ *and sends it to Alice.*

```
sage: m = 327
sage: c = mod(m^e, n)
sage: c
142145
```

Decryption
Alice receives $c = 142145$. *She first computes her decryption exponent* $d \equiv e^{-1} \pmod{\phi(n)} \equiv 71^{-1} \pmod{319716} = 211643$. *She then decrypts the ciphertext by computing* $m' \equiv c^d \pmod{n} \equiv 142145^{211643} \pmod{320857} = 327$.

```
sage: d = mod(e^(-1),phi)
sage: mprime = mod(c^d,n)
sage: mprime
327
```

She has recovered the plaintext.

4.1.1 The RSA Problem

An adversary, Eve, can intercept c, the ciphertext message. She already knows e, the encryption exponent, and n, the public modulus. Her task, then, is to compute m. We can frame this as the RSA Problem.

Problem 4.1.1 (RSA Hard Problem)**.**
Given an RSA public key (n, e) *and ciphertext* $c \equiv m^e \pmod{n}$, *compute* m.

This means that Eve would need to invert the RSA function to recover the plaintext message. There are a number of ways to attack the RSA when n is small. But, when n is sufficiently large and randomly generated, it is known that the RSA problem is hard to solve. That is, the RSA function is a *trapdoor function*. It would be easy to solve if Eve just knew p and q. Then, she could just compute $\phi(n)$ and decrypt just as Alice would. This is known as the Integer Factorization Problem. So, it is clear that breaking RSA boils down to solving the Integer Factorization Problem.

Problem 4.1.2 (Integer Factorization Problem)**.**
Given $n = pq$, find primes p and q.

It is not difficult to factor n when n is small. In fact, there are a number of factoring algorithms (and pre-programmed functions in Sage) that do this for us. But, when n is sufficiently large and randomly generated, it is very hard to find p and q.

4.2 Elgamal

We now discuss the Elgamal Cryptosystem of Taher Elgamal from 1985 as presented in [14]. Before getting into he cryptosystem itself, we need to look at its underlying hard problem.

4.2.1 Discrete Logarithm Problem

Let $\mathbb{F}_p$ be a finite field. Then there exists an element $g \in \mathbb{F}_p$ such that $\mathbb{F}_p^* = \{1, g, g^2, \ldots, g^{p-2}\}$. In particular, Fermat's Little Theorem says that $g^{p-1} = 1$. g is called a **primitive root** of $\mathbb{F}_p$.

Problem 4.2.1 (Discrete Logarithm Problem)**.**
Let g be a primitive root of $\mathbb{F}_p$. Let $h \in \mathbb{F}_p$ be nonzero. Find an exponent x such that $g^x \equiv h \pmod{p}$.

The number x is called the **discrete logarithm of h with base g**. It is denoted by $\log_g(h)$.

4.2.2 Diffie-Hellman Problem

The Diffie-Hellman Problem allows Alice and Bob to share some key without Eve being able to know what it is.

The Diffie-Hellman Key Exchange has public parameters (p, g), where p is a large prime and g is a primitive root of large prime order in $\mathbb{F}_p^*$.

Alice chooses her secret key $a \in \mathbb{Z}$ and computes $A \equiv g^a \pmod{p}$. She sends A to Bob. At the same time, Bob chooses his own secret key $b \in \mathbb{Z}$ and computes $B \equiv g^b \pmod{p}$. He sends B to Alice.

Alice receives B from Bob and computes $B^a \pmod{p}$. Bob receives A from Alice and computes $A^b \pmod{p}$. Note that

$$B^d \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \pmod{p},$$

so this is their shared secret key.

Example 4.2.1 (Diffie-Hellman Problem)**.**
The Diffie-Hellman Problem has public parameters $p = 1039$ *and* $g = 593$.

```
sage: p = 1039
sage: g = 593
```

Alice chooses $a = 323$ *and computes* $A \equiv 593^{323} \pmod{1039} = 108$. *She sends* $A = 108$ *to Bob.*

```
sage: a = 323
sage: A = mod(g^a, p)
sage: A
108
```

At the same time, Bob chooses $b = 691$ *and computes* $B \equiv 593^{691} \pmod{1039} = 613$. *He sends* $B = 613$ *to Alice.*

```
sage: b = 691
sage: B = mod(g^b, p)
sage: B
613
```

Alice receives $B = 613$ *and computes* $613^{323} \pmod{1039} = 617$.

```
sage: mod(B^a, p)
617
```

Bob receives $A = 108$ *and computes* $108^{691} \pmod{1039} = 617$.

```
sage: mod(A^b, p)
617
```

They both got 617*, so this is their shared key.*

Eve can inercept A and B. Her problem then becomes to solve one of the following:

$$593^a \equiv 108 (\mod 1039) \qquad \textit{or} \qquad 593^b \equiv 613 (\mod 1039).$$

Eve knows $A = g^a$, $B = g^b$, g, and p. If she can solve the Discrete Log Problem, she can find either a or b (or both), and then find g^{ab}, which is Alice and Bob's secret shared key.

Problem 4.2.2 (Diffie-Hellman Problem)**.**
Given the values of $g^a (\mod p)$ and $g^b (\mod p)$, find the value of $g^{ab} (\mod p)$.

If Eve can solve the Discrete Log Problem, then she can clearly solve the Diffie-Hellman Problem. Note that it is unknown whether solving the Diffie-Hellman Problem solves he Discrete Log Problem.

Elgamal Cryptosystem

The Elgamal Cyrptosystem has public parameters (p, g), where p is a large prime, and g is a primitive root of large prime order in $\mathbb{F}_p$.

Key Creation
Alice chooses a private key $a \in \mathbb{Z}$, where $1 \leq a \leq p-1$ and computes $A \equiv g^a (\mod p)$. She publishes A as her public key.

Encryption
Bob wants to send a plaintext message m to Alice, where $2 \leq m \leq p-1$. He chooses a random integer k and computes $c_1 = g^k (\mod p)$ and $c_2 = mA^k (\mod p)$. He sends (c_1, c_2) as his ciphertext to Alice.

Decryption
Alice receives (c_1, c_2) and computes $(c_1^a)^{-1} \cdot c_2 (\mod p)$. This gives her back exactly m.

Proposition 4.2.1. *In the Elgamal Cryptosystem, $(c_1^a)^{-1} \cdot c_2 (\mod p) = m$.*

Proof.

$$\begin{aligned}(c_1^a)^{-1} \cdot c_2 &\equiv ((g^k)^a)^{-1} \cdot mA^k (\mod p) \\ &\equiv (g^{ak})^{-1} \cdot mA^k (\mod p) \\ &\equiv (g^{ak})^{-1} m (g^a)^k (\mod p) \\ &\equiv (g^{ak})^{-1} (g^{ak}) m (\mod p) \\ &= m\end{aligned}$$

□

Example 4.2.2 (Elgamal)**.**
The Elgamal Cryptosystem has public parameters $p = 2677$ *and* $g = 993$.

```
sage: p = 2677
sage: g = 993
```

Key Creation
Alice chooses $a = 1234$ *and computes* $A \equiv g^a (\mod p) \equiv 993^{1234} (\mod 2677) = 2661$ *and sends it to Bob.*

```
sage: a = 1234
sage: A = mod(g^a,p)
sage: A
2661
```

Encryption
Bob receives $A = 2661$. *He wants to send* $m = 2132$ *to Alice. He picks* $k = 213$ *and computes* c_1 *and* c_2. *He gets* $c_1 \equiv g^k (\mod p) \equiv 993^{213} (\mod 2677) = 2563$ *and* $c_2 \equiv mA^k (\mod p) \equiv 2132 \cdot 2661^{213} (\mod 2677) = 2549$. *He sends* $(c_1, c_2) = (2563, 2549)$ *to Alice.*

```
sage: m = 2132
sage: k = 213
sage: c1 = mod(g^k,p)
sage: c1
2563

sage: c2 = mod(m*A^k,p)
sage: c2
2549
```

Decryption
Alice receives $(c_1, c_2) = (2563, 2549)$ *and computes* $(c_1^a)^{-1} \cdot c_2 (\mod p) \equiv (2563^{1234})^{-1} \cdot 2549 (\mod 2677 = 2132$. *This is exactly* m.

```
sage: mod((c1^a)^(-1)*c2,p)
2132
```

4.3 Knapsack

We will now introduce the Knapsack Cryptosystem by first discussing its hard problem, the Subset-Sum Problem.

4.3.1 The Subset-Sum Problem

Problem 4.3.1 (The Subset-Sum Problem)**.**
Suppose you are given a list $N = (N_1, N_2, \ldots, N_k)$ *of positive integers and another integer* M. *Find a subset of the elements of* N *that sum to* M, *assuming that there exists at least one such sum.*

Equivalently, let $\mathbf{v} = (v_1, v_2, \ldots, v_k)$ *be a binary vector. Given*

$$M = \sum_{i=1}^{k} v_i N_i,$$

find $\mathbf{v}$ *or another binary vector giving* M.

Example 4.3.1 (Subset-Sum)**.**
Let $N = (4, 6, 7, 11, 15, 17, 20, 31, 41, 43, 47)$ *and* $M = 98$. *Then, a subset whose entries sum to* M *is* $\{6, 7, 11, 31, 43\}$. *It is easy to check that this is the only such subset.*

Example 4.3.2 (Subset Sum)**.**
Use the same N *as our last example. Now, say we have* $M = 76$. *A subset whose entries sum to* M *is* $\{4, 11, 20, 41\}$. *Another subset whose entries sum to* M *is* $\{4, 31, 41\}$.

Remark 4.3.1. *The solution to a subset-sum problem, if it exists, is not necessarily unique.*

Remark 4.3.2. *It turns out that solving a Subset-Sum Problem is very difficult, so it cannot be used for a cryptosystem. However, if Alice has some secret information that guarantees a solution, it can be used. This leads us to superincreasing sequences.*

Definition 55 (Superincreasing Sequence)**.**
A **superincreasing sequence** *of integers is a list of positive integers* $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ *such that*

$$r_{i+1} \geq 2r_i \qquad \textit{for all } 1 \leq i < n.$$

The following algorithm, as presented in [14], solves the Subset-Sum problem for a superincreasing sequence $\mathbf{M}$ and target integer S, assuming that a solution exist. It takes $(\mathbf{M}, S)$ as input and outputs $\mathbf{x} \in \{0,1\}^n$, a binary solution to the Subset-Sum problem.

Algorithm 2 Subset-Sum for a Superincreasing Sequence

1: Loop i from n down to 1
2: If $S \geq M_i$
3: set $x_i = 1$
4: $S = S - M_i$
5: Else set $x_i = 0$
6: End loop

Example 4.3.3 (Subset-Sum Algorithm for a Superincreasing Sequence)**.**
Let $\mathbf{r} = (4, 9, 21, 45, 91, 187, 379)$ *and* $S = 327$. *Note that* $\mathbf{r}$ *is superincreasing since* $r_{i+1} \geq 2r_i$ *for all* $1 \leq i < 7$. *We will use Algorithm 2.*

$S = 327 \ngeq 379 = r_7$, *so* $x_7 = 0$.

$S = 327 \geq 187 = r_6$, *so* $x_6 = 1$.
$S = 327 - 187 = 140$

$S = 140 \geq 91 = r_5$, *so* $x_5 = 1$
$S = 140 - 91 = 49$

$S = 49 \geq 45 = r_4$, *so* $x_4 = 1$
$S = 49 - 45 = 4$

$S = 4 \ngeq 21 = r_3$, *so* $x_3 = 0$

$S = 4 \ngeq 9 = r_2$, *so* $x_2 = 0$

$S = 4 \geq 4 = r_1$, *so* $x_1 = 1$

Thus, $\mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$. *Note that* $\mathbf{rx} = (1)(4)+(0)(9)+(0)(21)+(1)(45)+(1)(91)+(1)(187)+(0)(379) =$ 327.

We will now discuss the Knapsack Cryptosystem, as presented in [14], a cryptosystem based on the Subset-Sum Problem and superincreasing sequences.

Key Creation

Alice chooses a superincreasing sequence $\mathbf{r} = (r_1, r_2, \ldots, r_n)$. She then chooses relatively prime integers A and B with $B > 2r_n$. She computes $M_i = Ar_i \pmod{B}$ for $1 \leq i \leq n$ and publishes $\mathbf{M} = (M_1, M_2, \ldots, M_n)$ as her public key.

Encryption

Bob chooses a binary plaintext message $\mathbf{x} \in \{0,1\}^n$ and computes $S = \mathbf{Mx}$ to send to Alice.

Decryption

Alice receives S and computes $S' \equiv A^{-1}S \pmod{B}$. She then uses Algorithm 2 to solve the Subset-Sum Problem for S' using her secret superincreasing sequence $\mathbf{r}$. She gets back $\mathbf{x}$, the plaintext, which satisfies $\mathbf{r} \cdot \mathbf{x} = S'$.

Example 4.3.4 (Knapsack Cryptosystem)**.**

Key Creation

Alice chooses superincreasing sequence $\mathbf{r} = (4, 9, 21, 45, 91, 187, 379)$ *and integers* $A = 123$ *and* $B = 802$. *Note that* $\gcd(A, B) = 1$ *and* $B > 2 \cdot 379 = 758$. *She then computes* $M_i = 123r_i \pmod{802}$ *for all* i *to get* $\mathbf{M} = (492, 305, 177, 723, 767, 545, 101)$. *Notice that this sequence is not superincreasing, so we could not use the described algorithm to solve a Subset-Sum problem with it. Alice publishes* $\mathbf{M}$.

```
sage: A = 123
sage: B = 802
sage: gcd(A,B)
```

```
1

sage: R = IntegerModRing(B)
sage: r = Matrix(R, [4,9,21,45,91,187,379])
sage: M = A*r
sage: M
[492 305 177 723 767 545 101]
```

Encryption

Bob wants to encrypt $\mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$. *He computes* $S = \mathbf{Mx} = (1)(492)+(0)(305)+(0)(177)+(1)(723)+(1)(767)+(1)(545)+(0)(101) = 2527$ *to send to Alice.*

```
sage: Mb = Matrix([492, 305, 177, 723, 767, 545, 101])
sage: x = vector([1,0,0,1,1,1,0])
sage: S = Mb*x
sage: S
(2527)
```

Decryption

Alice receives $S = 2527$ *and computes* $S' \equiv 123^{-1}(2527)(\mod 802) = 327$. *She then solves the Subset-Sum problem for* S' *using* $\mathbf{r}$ *as we did in Example 4.3.3. Note that she recovers exactly* $\mathbf{x}$.

```
sage: S = 2527
sage: Sprime = mod(A^(-1)*S, B)
sage: Sprime
327
```

Attack

Eve intercepts $S = 2527$, *but she does not know the superincreasing sequence. So, she cannot decrypt the ciphertext.*

4.4 Exercises

1. Bob wants to send plaintext message $m = 8$ to Alice using the RSA. Use the public key $(33, 7)$ to calculate the ciphertext c. Then, use Alice's private key $(3, 11, 20)$ to recover the plaintext.

2. Suppose you choose $p = 19$ and $q = 17$ for part of your private key with the RSA Cryptosystem. Using $e = 283$ as your exponent, calculate d.

3. Suppose $\phi(n) = 439200$. Determine which of the following, if any, are valid public exponents: $e_1 = 3, e_2 = 5, e_3 = 7$.

4. **Collaborative** Use $p = 197$ and $q = 211$ for the RSA Cryptosystem. Compute n and $\phi(n)$, and choose e. Give (n, e) to a partner, and have a partner encrypt a numerical plaintext message of their choosing. Once your partner gives you the ciphertext, recover the plaintext message.

5. For the Knapsack Cryptosystem, take $\mathbf{r} = (4, 9, 21, 45, 91, 187, 379)$. Suppose $A = 145$ and $B = 901$.

   (a) Compute M.

   (b) Encrypt $\mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$.

   (c) Decrypt the cyphertext to recover $\mathbf{x}$.

4.5 Computer Exercises

1. The Diffie-Hellman Key Exchange has public parameters $p = 2203$ and $g = 421$. Alice sends Bob the value of $A = 794$. Bob uses the secret value $b = 245$.

   (a) What value of B should Bob send to Alice?

   (b) What is Alice and Bob's shared secret value?

   (c) Can you figure out a?

2. The public parameters of the Elgamal Cryptosystem are $p = 1951$ and $g = 213$.

   (a) Alice chooses $a = 823$ as her private key. What is her public key?

   (b) Bob wants to send Alice the message $m = 1023$. He chooses random $k = 119$. What is the ciphertext (c_1, c_2) that he should send to Alice?

   (c) Alice receives the (c_1, c_2) from part (b). Decrypt this ciphertext to recover m.

   (d) Suppose that Eve intercepts (c_1, c_2). What is the problem that she must solve in order to decrypt the ciphertext? Can you solve it?

3. Find the two prime factors of 260947.

4. Suppose that Alice receives ciphertext $c = 522885$ from Bob. Alice knows that $\phi(n) = 913440$ and $e = 179$. If $n = pq$, where $p = 1039$, find q. Then, decrypt the ciphertext.

5. Suppose that $n = pq = 667$ and the public exponent is $e = 3$ for the RSA Cyrptosystem.

   (a) Encrypt the plaintext message $m = 273$.

   (b) Determine p and q.

   (c) Use p and q to recover the plaintext message m.

6. Suppose that a public key for an RSA Cryptosystem is $(n, e) = (2331757, 23)$.

   (a) Encrypt the plaintext message $m = 732985$.

   (b) Suppose you know that $n = pq$, where $p = 1451$. Find q, and use it to calculate d.

7. The following code takes as input p, q, and e of the RSA cryptosystem and outputs decryption exponent d. Write code the executes the remaining portions of the RSA cryptosystem.

```
def generate_d(p,q,e):
    if not is_prime(p):
        print(‘‘p is not prime’’)
        return
    if not is_prime(q):
        print(‘‘q is not prime’’)
        return
    phi = (p-1)*(q-1)
```

```
if gcd(e, phi) != 1:
    print(‘‘e and phi are not relatively prime’’)
    return
return inverse_mod(e, phi)
```

8. Write a computer program that executes the Subset-Sum of a Superincreasing Sequence alogorithm in 2.

CHAPTER

5

INTRODUCTION TO LATTICES

With algorithms like Shor's Algorithm (see [26]), which can factor a composite number into the product of primes in polynomial time on a Quantum computer, and the promise for Quantum computers (see [28]) in the near future, comes the need for Quantum-resistant cryptosystems. Constructions based on lattices are proving promising. Before getting into these cyrptosystems, we must first become comfortable with lattices and their properties. We draw on our knowledge of vector spaces to work with lattices. We follow [14] for many of the definitions and properties.

5.1 Definition and Basic Properties

Definition 56 (Lattice)**.**
Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \in \mathbb{R}^n$ *be linearly independent vectors. The* **lattice** $\mathscr{L}$ *generated by these vectors is the set of all linear combinations of* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ *with integer coefficients. That is,*

$$\mathscr{L} = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_n\mathbf{v}_n : a_1, a_2, \ldots, a_n \in \mathbb{Z}\} = \left\{ \sum_{i=1}^{n} a_i\mathbf{v}_i : a_i \in \mathbb{Z}, 1 \leq i \leq n \right\}.$$

Vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ *are called* **basis** *vectors of* $\mathscr{L}$*.*

Definition 57 (Integer Lattice)**.**
If basis vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \in \mathbb{Z}^n$, $\mathscr{L}$ *is called an* **integer (or integral) lattice**.

We will deal exclusively with integer lattices.

Definition 58 (Dimension)**.**
The **dimension** *of a lattice is the number of basis vectors.*

Example 5.1.1 (Lattice)**.**
The 2*—dimensional integer lattice in Figure 5.1 generated by* $\mathbf{v}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ *and* $\mathbf{v}_2 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ *is given by*

$$\mathscr{L} = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 : a_1, a_2 \in \mathbb{Z}\}.$$

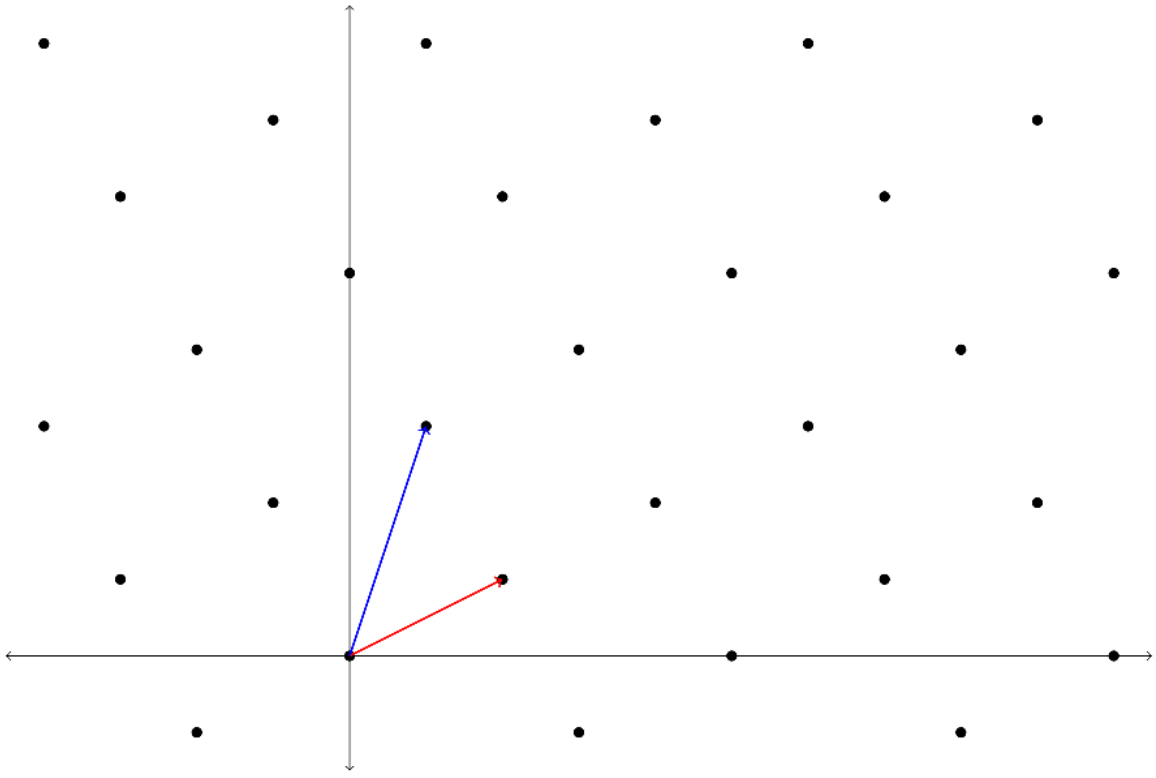


Figure 5.1 $\mathscr{L} = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 : a_1, a_2 \in \mathbb{Z}\} \subset \mathbb{R}^2$

Let us now look at how we could generate the matrix representation of the lattice from Example 5.1.1 with SageMath.

```
sage: M = Matrix(ZZ,[[2,1],[1,3]])
```

```
sage: M
[2 1]
[1 3]
```

We can also check whether or not specific vectors are in the lattice.

```
sage: vector([1,1]) in span(M)
False

sage: vector([4,2]) in span(M)
True

sage: vector([3,-1]) in span(M)
True

sage: vector([-36,12]) in span(M)
True
```

We can also find the linear combination of basis vectors using Sage. Say we want to know what specific linear combination of $\mathbf{v}_1$ and $\mathbf{v}_2$ gives us that last vector that we tested, $\begin{bmatrix} -36 \\ 12 \end{bmatrix}$. We just use the solve_left command.

```
sage: b = vector([-36,12])
sage: M.solve_left(b)
(-24, 12)
```

Then, we can check that this is, in fact, true:

```
sage: -24*M[0] + 12*M[1]
(-36, 12)
```

We can also generate a random lattice basis with SageMath:

```
sage: sage.crypto.lattice.gen_lattice(m=10, lattice=True)
Free module of degree 10 and rank 10 over Integer Ring
User basis matrix:
[ 0  1  0 -1  0 -1  0 -1  0  1]
[ 1 -1 -1  0  1  0  1  1  0  0]
```

```
[ 0  0 -1 -1 -1 -1  1  0  1  0]
[ 0  0  1  0  0 -1 -1 -1  2  0]
[ 1  1  1  1 -1  1  1  0  1  0]
[-1  0  1  0  0 -1  1  1  0 -2]
[ 1 -1  1 -1 -1  2  0  0  1  0]
[-1  0  1 -1  0 -1 -1  2 -1  1]
[ 0 -1  1  1  0 -2  2 -1 -1  1]
[-3  0  0  1  0  1  1  0  2  1]
```

Recall that Sage uses row vectors.

Remark 5.1.1. *An integer lattice is an additive subgroup of $\mathbb{Z}^n$ for some n.*

Definition 59 (Fundamental Domain)**.**
Let $\mathscr{L}$ be an $n-$ dimensional lattice with ordered basis $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$. The **fundamental domain** *of $\mathscr{L}$ corresponding to $\mathscr{B}$ is given by*

$$F(\mathscr{B}) = \left\{ \sum_{i=1}^{n} c_i \mathbf{v}_i : 0 \leq c_i < 1 \right\}.$$

A fundamental domain is also called a **fundamental region** *or a* **fundamental parallelepiped**.

Figure 5.2 shows a fundamental region $\mathscr{F}$ of the lattice $\mathscr{L}$ from Figure 5.1.

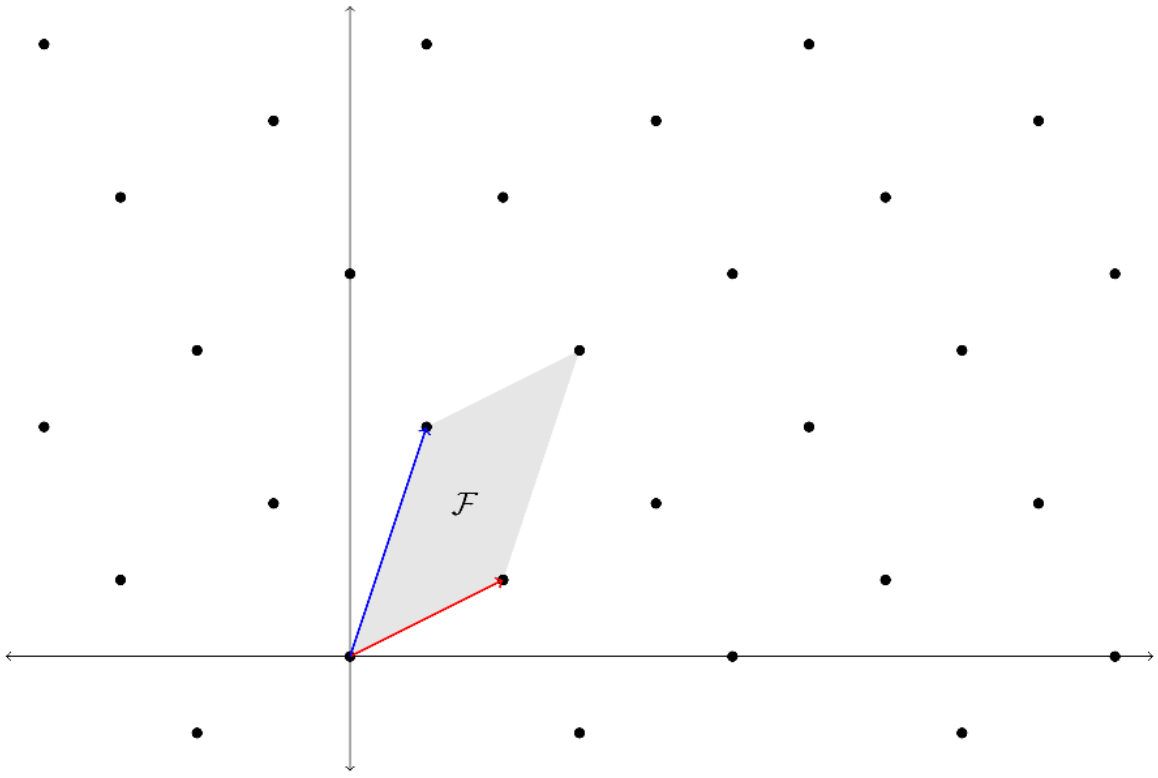


Figure 5.2 A fundamental domain $\mathscr{F} = \mathscr{F}(\mathbf{v}_1, \mathbf{v}_2)$ of $\mathscr{L}$

We now state a proposition that we will need in Section 5.3.

Proposition 5.1.1. *Let $\mathscr{L}$ be an $n-$dimensional lattice with a fundamental region $\mathscr{F}$. Then every vector $\mathbf{w} \in \mathbb{R}^n$ can be uniquely written in the form*

$$\mathbf{w} = \mathbf{x} + \mathbf{v}, \text{ where } \mathbf{x} \in \mathscr{F} \text{ and } \mathbf{v} \in \mathscr{L}.$$

Equivalently,

$$\bigcup_{\mathbf{v} \in \mathscr{L}} \mathscr{F} + \mathbf{v} = \{\mathbf{x} + \mathbf{v} : \mathbf{x} \in \mathscr{F}\}$$

exactly covers $\mathbb{R}^n$.

Figure 5.3 shows a geometric example of the same lattice $\mathscr{L}$ covering $\mathbb{R}^2$.

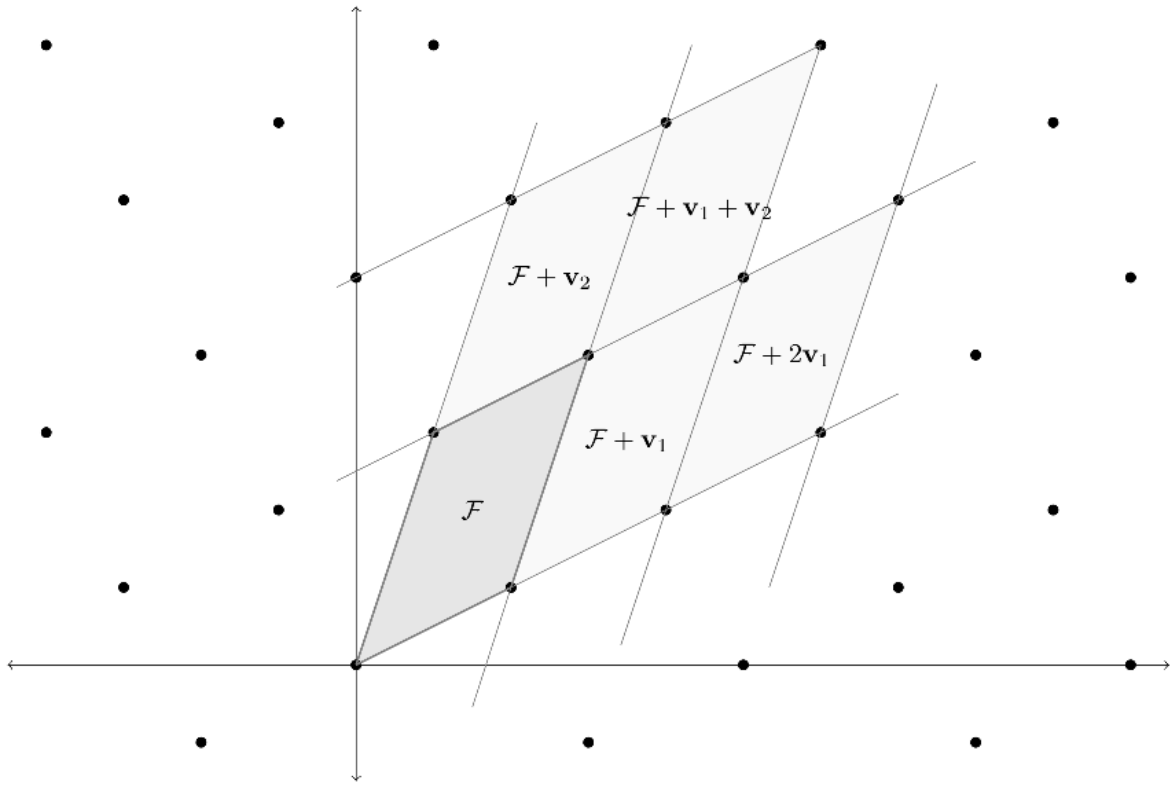


Figure 5.3 $\bigcup_{\mathbf{v}\in\mathscr{L}} \mathscr{F}+\mathbf{v}=\{\mathbf{x}+\mathbf{v}:\mathbf{x}\in\mathscr{F}\}$ covers $\mathbb{R}^2$.

We now prove the proposition.

Proof. Let $\mathscr{B}=\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_n\}$ be a basis for lattice $\mathscr{L}\subset\mathbb{R}^n$ and $\mathscr{F}=\mathscr{F}(\mathscr{B})$ a fundamental region of $\mathscr{L}$. Since $\mathscr{B}$ is n linearly independent vectors in $\mathbb{R}^n$, $\mathscr{B}$ is also a basis for $\mathbb{R}^n$. So, any $\mathbf{w}\in\mathbb{R}^n$ can be written as

$$\mathbf{w}=\sum_{i=1}^{n} a_i\mathbf{v}_i \text{ for some } a_i\in\mathbb{R}.$$

Write $a_i=c_i+k_i$ with $0\le c_i<1$ and $k_i\in\mathbb{Z}$. Then

$$\begin{aligned}\mathbf{w}&=(c_1+k_1)\mathbf{v}_1+(c_2+k_2)\mathbf{v}_2+\cdots+(c_n+k_n)\mathbf{v}_n\\&=c_1\mathbf{v}_1+k_1\mathbf{v}_1+c_2\mathbf{v}_2+k_2\mathbf{v}_2+\cdots+c_n\mathbf{v}_n+k_n\mathbf{v}_n\\&=\underbrace{(c_1\mathbf{v}_1+c_2\mathbf{v}_2+\cdots+c_n\mathbf{v}_n)}_{\mathbf{x}\in\mathscr{F}}+\underbrace{(k_1\mathbf{v}_1+k_2\mathbf{v}_2+\cdots+k_n\mathbf{v}_n)}_{\mathbf{v}\in\mathscr{L}}\end{aligned}$$

Uniqueness follows from $0\le c_i<1$. □

Definition 60 (Determinant)**.**
Let $\mathscr{L}$ be an $n-$dimensional lattice with fundamental domain $\mathscr{F}$. The **determinant** *of $\mathscr{L}$, denoted* $\det(\mathscr{L})$*, is the volume of $\mathscr{F}$.*

Suppose $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ is a basis for lattice $\mathscr{L}$ with corresponding fundamental domain $\mathscr{F}$. Let V be the $n \times n$ matrix whose ith column is $\mathbf{v}_i$. Then

$$\det(\mathscr{L}) = Vol(\mathscr{F}) = |\det(V)|.$$

Remark 5.1.2. *All fundamental domains of a single lattice have the same volume since any two bases for a lattice are related by a unimodular matrix.*

Definition 61 (Hadamard Inequality)**.**
Let lattice $\mathscr{L} \subset \mathbb{R}^n$ have basis $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, and let $\mathscr{F} = \mathscr{F}(\mathscr{B})$ be the corresponding fundamental domain. The **Hadamard Inequality** *is given by*

$$\det(\mathscr{L}) = Vol(\mathscr{F}) \leq \|\mathbf{v}_1\|\|\mathbf{v}_2\| \cdots \|\mathbf{v}_n\|.$$

The closer $\mathscr{B}$ is to orthogonal, the closer this comes to equality.

Definition 62 (Hadamard Ratio)**.**
For a basis $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ of lattice $\mathscr{L} \subset \mathbb{R}^n$, the **Hadarmard Ratio** *of $\mathscr{B}$ is given by*

$$\mathscr{H}(\mathscr{B}) = \left(\frac{\det(\mathscr{L})}{\|\mathbf{v}_1\|\|\mathbf{v}_2\| \cdots \|\mathbf{v}_n\|} \right)^{1/n},$$

where $0 < \mathscr{H}(\mathscr{B}) \leq 1$. The closer this value is to 1, the closer $\mathscr{B}$ is to orthogonal. The reciprocal of the Hadamard Ratio is often called the orthogonality defect.

5.2 Lattice Hard Problems

As we know, the security of cryptosystems relies on the hardness of their underlying problems. We now discuss two well-studied hard lattice problems. Note that there are other versions of these, but we will just discuss the basic Shortest Vector Problem and the Closest Vector Problem. We again follow [14] in defining them.

5.2.1 The Shortest Vector Problem

Problem 5.2.1 (The Shortest Vector Problem)**.**
Find a shortest nonzero vector in lattice $\mathscr{L}$. That is, find $\mathbf{0} \neq \mathbf{v} \in \mathscr{L}$ that minimizes $\|\mathbf{v}\|$.

Example 5.2.1 (Shortest Vector)**.**
As shown in Figure 5.4, the lattice $\mathscr{L} = \left\{ a_1 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + a_2 \begin{bmatrix} 1 \\ 3 \end{bmatrix} : a_a, a_2 \in \mathbb{Z} \right\}$ *from Figure 5.1 has a short vector* $\mathbf{v} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$*. That is,* $\|\mathbf{v}\| = \sqrt{2^2 + 1^2} = \sqrt{5}$ *is the smallest Euclidean norm in* $\mathscr{L}$*.*
Note that $\mathbf{v}' = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ *is also a short vector in* $\mathscr{L}$*, as* $\|\mathbf{v}'\| = \sqrt{(-1)^2 + 2^2} = \sqrt{5}$*. It is clear that a solution to The Shortest Vector Problem (SVP) is not unique. For that reason, we call* $\mathbf{v}$ *(and* $\mathbf{v}'$*) a short vector, instead of the short vector.*
We follow [14] in defining the Gaussian expected shortest length of a shortest vector, and we encourage the reader to see [14] for an in depth discussion of it.

Definition 63 (Gaussian expected shortest length)**.**
Let $\mathscr{L}$ *be a lattice of dimension* n*. The* **Gaussian expected shortest length** *is*

$$\sigma(\mathscr{L}) = \sqrt{\frac{n}{2\pi e}} (\det(\mathscr{L}))^{1/n},$$

and we expect that a shortest vector $\mathbf{v} \in \mathscr{L}$ *satisfy*

$$\|\mathbf{v}\| \approx \sigma(\mathscr{L}).$$

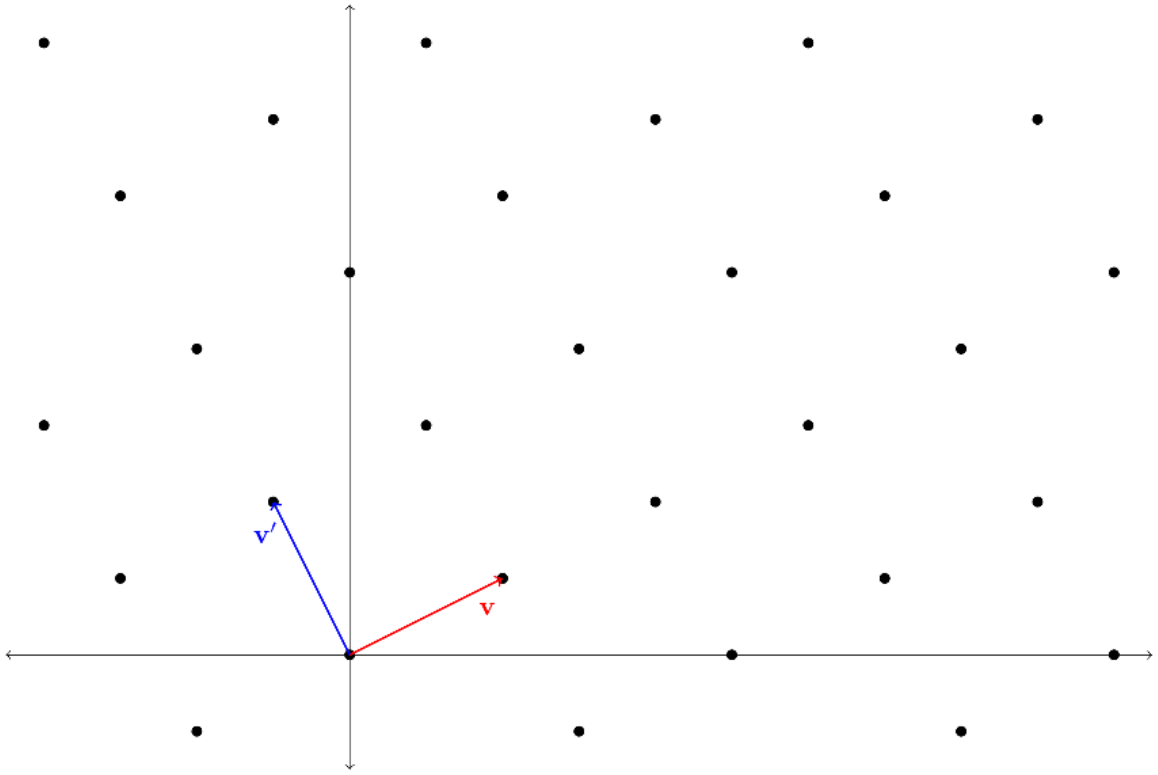


Figure 5.4 $\mathbf{v}$ and $\mathbf{v}'$ are shortest vectors in $\mathscr{L}$.

Example 5.2.2 (Gaussian Expected Length)**.**
From our previous example, we expect the shortest length to be

$$\begin{aligned}\sigma(\mathscr{L}) &= \sqrt{\frac{n}{2\pi e}}(\det(\mathscr{L}))^{1/n} \\ &= \sqrt{\frac{2}{2\pi e}}|(2)(3)-(1)(1)|^{1/2} \\ &= \sqrt{\frac{1}{\pi e}}\sqrt{5} \\ &= \sqrt{\frac{5}{\pi e}} \\ &\approx 1.35624\end{aligned}$$

Note that $\sqrt{5} \approx 2.23607$*, which is relatively close to the expected shortest length.*

5.2.2 The Closest Vector Problem

Problem 5.2.2 (The Closest Vector Problem)**.**
Given a vector $\mathbf{w} \in \mathbb{R}^n$ *with* $\mathbf{w} \notin \mathscr{L}$*, find a* $\mathbf{v} \in \mathscr{L}$ *that is closest to* $\mathbf{w}$*. That is, find* $\mathbf{v} \in \mathscr{L}$ *that minimizes* $\|\mathbf{w}-\mathbf{v}\|$*. We call* $\mathbf{w}$ *our "target vector."*

We now switch to identifying all vectors with points in space, so that our figures do not get too crowded.

Example 5.2.3 (Closest Vector)**.**
Take $\mathscr{L}$ *to be as in our previous examples, and suppose our target vector is* $\mathbf{w} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$*. Note that* $\mathbf{w} \notin \mathscr{L}$*, as there does not exist* $a_1, a_2 \in \mathbb{Z}$ *such that* $a_1\mathbf{v}_1 + a_2\mathbf{v}_2 = \mathbf{w}$*.*
The closest vector to $\mathbf{w}$ *is* $\mathbf{v} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$*, as* $\|\mathbf{w}-\mathbf{v}\| = \left\| \begin{bmatrix} 4 \\ 3 \end{bmatrix} - \begin{bmatrix} 4 \\ 2 \end{bmatrix} \right\| = 1$ *is the minimum distance between* $\mathbf{w}$ *and any other* $\mathbf{y} \in \mathscr{L}$*.*

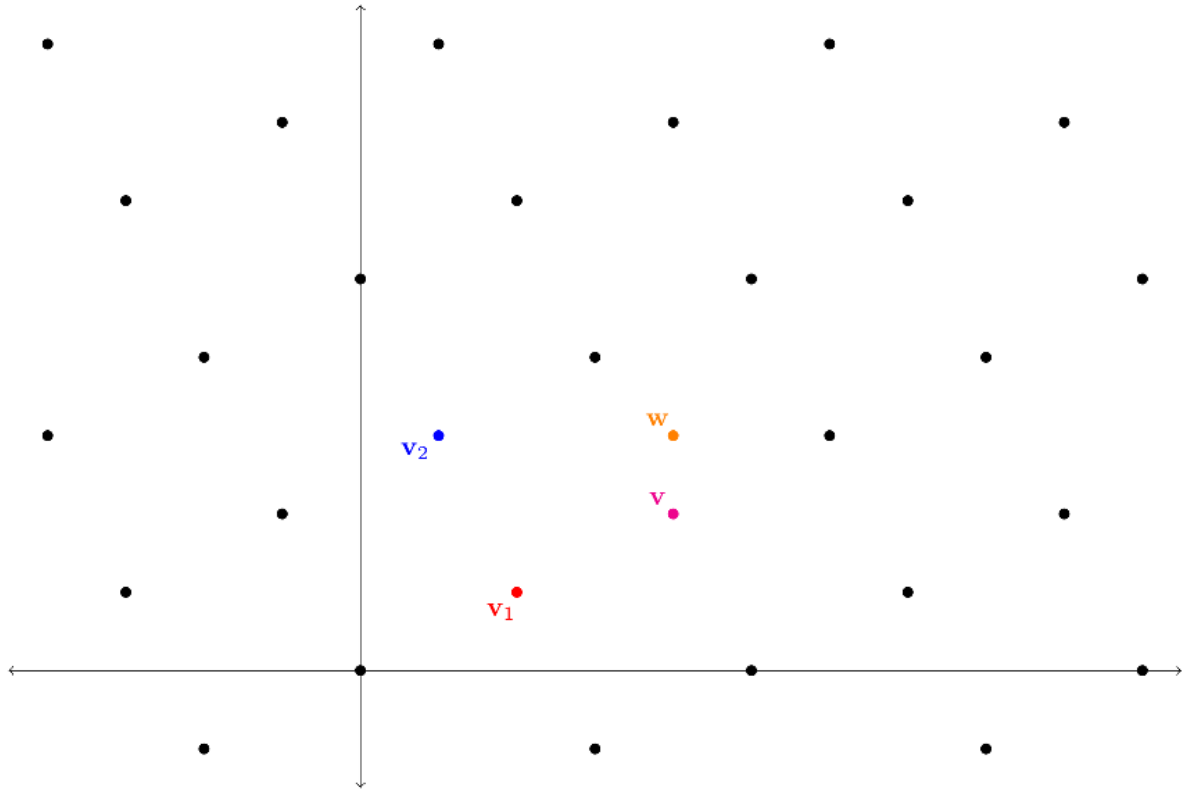


Figure 5.5 $\mathbf{v} \in \mathscr{L}$ is the closest lattice vector to the target vector $\mathbf{w} \in \mathbb{R}^2$.

As in Definition 63, we again follow [14] in saying that if $\mathscr{L}$ is a lattice of dimension n and

$\mathbf{w} \in \mathbb{R}^n$ is a random target point, then we expect $\mathbf{v} \in \mathscr{L}$ to satisfy

$$\|\mathbf{v} - \mathbf{w}\| \approx \sigma(\mathscr{L}).$$

5.3 Babai's Algorithm

We have seen that closest vectors can be relatively easy to find when we have a picture in 2—D. A natural next question would be *How can we find the closest vector to a target point for any lattice?*

Recall the definition and visual for a lattice's Fundamental Domain (Definition 59 and Figure 5.2). Recall also, as in Figure 5.3, that the union of unique representations of vectors in $\mathbb{R}^n$ as the sum of a point in the lattice and a point in the fundamental domain exactly covers $\mathbb{R}^n$.

Consider Example 5.2.3 again. Draw the "shifted" fundamental region around the target point $\mathbf{w}$, as shown in Figure 5.6. Observe that the closest lattice vector to the target point is the closest vertex of the surrounding parallelepiped.

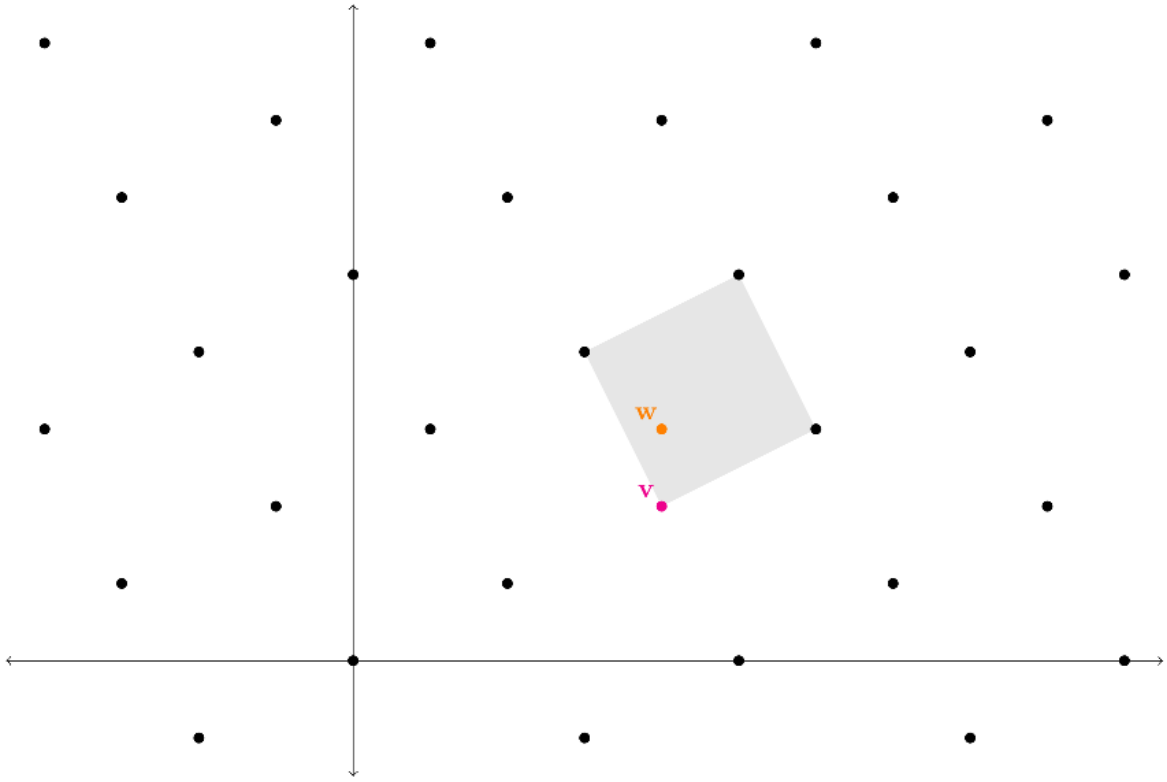


Figure 5.6 The closest lattice point **v** to target point **w** is also the closest vertex of **w**'s surrounding parallelogram.

Let us now consider a different basis for $\mathscr{L}$ with the same target vector **w**. Suppose we use the basis $\left\{\begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}7\\6\end{bmatrix}\right\}$. If we draw our fundamental region around the target point (as in Figure 5.7), we see that the closest parallelogram vertex is not the closest lattice point to the target point. Why is that?

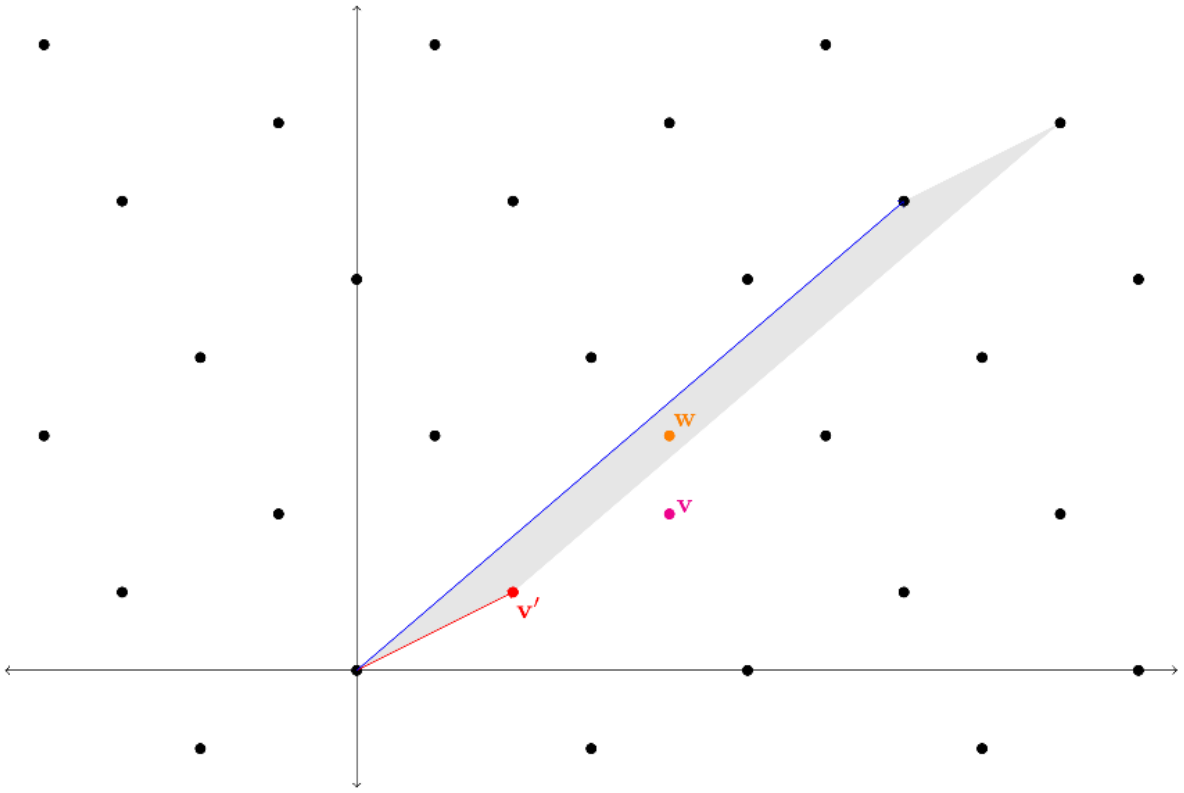


Figure 5.7 $\mathbf{v} \in \mathscr{L}$ is the closest lattice point to target point $\mathbf{w} \in \mathbb{R}^2$, but $\mathbf{v}' \in \mathscr{L}$ is the closest parallelogram vertex to $\mathbf{w}$.

As it turns out, the idea that the closest vertex is the closest lattice point is dependent on how orthogonal the basis vectors are. In Figure 5.6, the basis vectors are close to orthogonal; in Figure 5.7, the basis vectors are close to parallel. In order for this idea to work out, we need that the basis vectors are *reasonably orthogonal*.

Definition 64 (Good Basis and Bad Basis [14])**.**
A **good basis** *is a basis in which the vectors are "reasonably orthogonal," or close to pairwise orthogonal.*
A **bad basis** *is a basis in which the vectors are "reasonably parallel," or far from pairwise orthogonal.*

Orthogonality of Two Vectors

Recall from Definition 22 that the angle between two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ satisfies

$$\cos(\theta) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|\|\mathbf{v}_2\|}. \tag{5.1}$$

The closer this ratio is to 0, the less $\mathbf{v}_1$ and $\mathbf{v}_2$ are to being orthogonal. Therefore, two vectors

are reasonably orthogonal if the ratio in (5.1) is close to 0.

This definition is practical to use in very small dimensions, but we need a way to check for orthogonality in larger dimensions.

Recall from Definition 62 that the Hadamard Ratio corresponding to basis $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ for $\mathscr{L}$ is given by

$$\mathscr{H}(\mathscr{B}) = \left(\frac{\det(\mathscr{L})}{\|\mathbf{v}_1\|\|\mathbf{v}_2\| \cdots \|\mathbf{v}_n\|} \right)^{1/n}. \tag{5.2}$$

The closer this value is to 1, the closer $\mathscr{B}$ is to orthogonal. Therefore, vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are reasonably orthogonal if (5.2) is close to 1.

Babai's Closest Vertex Algorithm finds the closest vertex to the targest point by writing the target vector as a linear combination of the basis vectors and then rounding them to integers. In this algorithm, we use the notation $\lfloor c_i \rceil$ to represent the nearest integer to c_i.

Theorem 5.3.1 (Babai's Closest Vertex Algorithm [14])**.**
Let $\mathscr{L} \subset \mathbb{R}^n$ be a lattice, and let $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ be a reasonably orthogonal basis for $\mathscr{L}$. Suppose $\mathbf{w} \in \mathbb{R}^n$ is an arbitrary vector. Then the following algorithm solves the Closest Vector Problem.

Babai's Closest Vertex Algorithm inputs $\mathbf{w} \in \mathbb{R}^n$, the target vector, and $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, a reasonably orthogonal basis for $\mathscr{L}$. It outputs $\mathbf{v} \in \mathscr{L}$, the closest lattice vector to the target vector $\mathbf{w}$.

Algorithm 3 Babai's Closest Vertex Algorithm

1: Write $\mathbf{w} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_n\mathbf{v}_n$ with $c_i \in \mathbb{R}$, $i = \{1, 2, \ldots, n\}$
2: Set $a_i = \lfloor c_i \rceil$ for $i = \{1, 2, \ldots, n\}$
3: Return $\mathbf{v} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_n\mathbf{v}_n$

Algorithm 3 is sometimes referred to as "Babai's Rounding Algorithm."

Example 5.3.1 (Babai's Algorithm)**.**
Let's revisit our example (Example 5.2.3) of finding the closest lattice point to target point $\mathbf{w} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$ *using the lattice basis* $\mathscr{B} = \left\{ \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right\}$.

We first check that the basis vectors are reasonably orthogonal. We will do this using the Hadamard Ratio. Let $V = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$.

$$\begin{aligned} \left(\frac{|\det(V)|}{\|\mathbf{v}_1\|\|\mathbf{v}_2\|}\right)^{1/2} &= \left(\frac{|(2)(3)-(1)(1)|}{\sqrt{5}\cdot\sqrt{10}}\right)^{1/2} \\ &= \left(\frac{5}{\sqrt{50}}\right)^{1/2} \\ &\approx .840896 \end{aligned}$$

We can do this in Sage:

```
sage: V = Matrix([[2,1],[1,3]])
sage: v1 = V[:,0]
sage: v2 = V[:,1]
sage: (abs(det(V))/(v1.norm()*v2.norm()))^(1/2)
0.8408964152537145
```

Though not as close as we would like, this is reasonably close to 1*, so we can consider* $\mathscr{B}$ *to be a "good" basis.*

We now proceed to using Babai's Algorithm (Algorithm 3):

Step 1 *Write* $\mathbf{w} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2$*. That is, find* $c_1, c_2 \in \mathbb{R}$ *such that* $\mathbf{w} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2$.

$$\begin{bmatrix} 4 \\ 3 \end{bmatrix} = c_1\begin{bmatrix} 2 \\ 1 \end{bmatrix} + c_2\begin{bmatrix} 1 \\ 3 \end{bmatrix}$$
$$\begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 2c_1 + 1c_2 \\ 1c_1 + 3c_2 \end{bmatrix}$$

Setting corresponding components equal to one another yields the system of equations

$$\begin{cases} 2c_1 + c_2 &= 4 \\ c_1 + 3c_2 &= 3 \end{cases}$$

Solving this system of equations gives

$$c_1 = 1.8$$
$$c_2 = 0.4$$

```
sage: V = Matrix([[2,1],[1,3]])
sage: b = vector([4,3])
sage: V\b.n()
(1.80000000000000, 0.400000000000000)
```

Step 2 Round c_1 and c_2 to the nearest integer.

$$\lfloor c_1 \rceil = \lfloor 1.8 \rceil = 2 = a_1$$
$$\lfloor c_2 \rceil = \lfloor 0.4 \rceil = 0 = a_2$$

Step 3 Return closest vector $\mathbf{v} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2$.

$$\begin{aligned}\mathbf{v} &= 2\begin{bmatrix}2\\1\end{bmatrix} + 0\begin{bmatrix}1\\3\end{bmatrix}\\ &= \begin{bmatrix}4\\2\end{bmatrix}\end{aligned}$$

```
sage: v = 2*V[:,0] + 0*V[:,1]
sage: v
[4]
[2]
```

Therefore, the closest lattice point to $\mathbf{w}$ *is* $\mathbf{v}$ *by Babai's Algorithm.*

Example 5.3.2 (Babai's Algorithm with a Bad Basis)**.**
Now, let's look at the same problem with our basis from Figure 5.7. Consider the basis $\mathscr{B}' = \left\{\begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}7\\6\end{bmatrix}\right\}$ *for the same lattice. We will use the same target point,* $\mathbf{w} = \begin{bmatrix}4\\3\end{bmatrix}$.

We must first check to see if the basis is "good." We can again use the Hadamard Ratio.

$$\begin{aligned}\left(\frac{\det \mathscr{L}}{\|\mathbf{v}_1'\|\|\mathbf{v}_2'\|}\right)^{1/2} &= \left(\frac{|(2)(6)-(1)(7)|}{\sqrt{2^2+1^2}\sqrt{7^2+6^2}}\right)^{1/2}\\ &= \left(\frac{5}{\sqrt{425}}\right)^{1/2}\\ &\approx 0.4925\end{aligned}$$

```
sage: B = Matrix([[2,7],[1,6]])
sage: b1 = B[:,0]
sage: b2 = B[:,1]
sage: (abs(det(B))/(b1.norm()*b2.norm()))^(1/2).n()
0.4924790605054523
```

This is reasonably far from 1*, so we can consider* $\mathscr{B}'$ *to be a "bad" basis since the vectors are close to parallel. Though we should stop here, we will continue to use Babai's Algorithm to illustrate a point.*

Step 1 *Write* $\mathbf{w} = c_1\mathbf{v}_1' + c_2\mathbf{v}_2'$*. That is, find* $c_1, c_2 \in \mathbb{R}$ *such that* $\mathbf{w} = c_1\mathbf{v}_1' + c_2\mathbf{v}_2'$*.*

$$\begin{bmatrix} 4 \\ 3 \end{bmatrix} = c_1 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + c_2 \begin{bmatrix} 7 \\ 6 \end{bmatrix}$$
$$\begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 2c_1 + 7c_2 \\ 1c_1 + 6c_2 \end{bmatrix}$$

Setting corresponding components equal to one another yields the system of equations

$$\begin{cases} 2c_1 + 7c_2 &= 4 \\ c_1 + 6c_2 &= 3 \end{cases}$$

Solving this system of equations gives

$$c_1 = 0.6$$
$$c_2 = 0.4$$

```
sage: B = Matrix([[2,7],[1,6]])
sage: w = vector([4,3])
sage: B\w.n()
(0.600000000000000, 0.400000000000000)
```

Step 2 *Round* c_1 *and* c_2 *to the nearest integer.*

$$\lfloor c_1 \rceil = \lfloor 0.6 \rceil = 1 = a_1$$
$$\lfloor c_2 \rceil = \lfloor 0.4 \rceil = 0 = a_2$$

Step 3 Return closest vector $\mathbf{v} = a_1\mathbf{v}_1' + a_2\mathbf{v}_2'$.

$$\begin{aligned}\mathbf{v}' &= 1\begin{bmatrix}2\\1\end{bmatrix} + 0\begin{bmatrix}7\\6\end{bmatrix}\\ &= \begin{bmatrix}2\\1\end{bmatrix}\end{aligned}$$

```
sage: 1*B[:,0] + 0*B[:,1]
[2]
[1]
```

Therefore, the closest vertex to $\mathbf{w}$ *is* $\mathbf{v}'$ *by Babai's Algorithm. However, we know that* $\mathbf{v} = \begin{bmatrix}4\\2\end{bmatrix}$ *is closer, since* $\|\mathbf{w}-\mathbf{v}\| = 1$ *and* $\|\mathbf{w}-\mathbf{v}'\| = \sqrt{8} \approx 2.828$. *It is clear that Babai's algorithm failed here, as we did not find the closest lattice point. This is because we started with a "bad" basis.*

We can also check that $\sigma(\mathscr{L}) = \sqrt{\frac{2}{2\pi e}}(\det(\mathscr{L}))^{1/2} \approx 0.765$, *and* 1 *is much closer to this than* $\sqrt{8}$ *is.*

```
sage: numerical_approx(sqrt(2/(2*pi*e))*(det(V))^(1/2))
0.765178616561644
```

Example 5.3.3 (Babai's Algorithm with Sage)**.**
We will work another short example of Babai's Algorithm using Sage.

Suppose we have a good basis $\mathscr{B} = \left\{\begin{bmatrix}7\\-1\end{bmatrix}, \begin{bmatrix}1\\3\end{bmatrix}\right\}$. *Let* V *be the matrix with these basis vectors as columns. Recall that Sage takes the rows of a matrix, so we enter* V *as follows:*

```
sage: V = Matrix([[7,1],[-1,3]])
```

Suppose our target vector is $\mathbf{w} = \begin{bmatrix}8\\-3\end{bmatrix}$. *Then, we enter*

```
sage: w = vector([8,-3])
```

To find c_1 *and* c_2 *such that* $\mathbf{w} = c_1\begin{bmatrix}7\\-1\end{bmatrix} + c_2\begin{bmatrix}1\\3\end{bmatrix}$, *we simply solve the system* $V\mathbf{c} = \mathbf{w}$. *Since the columns of* V *form a basis for* $\mathscr{L}$, V *is invertible. So, we can simply calculate* $\mathbf{c} = V^{-1}\mathbf{w}$ *by doing*

```
sage: c = V\w
sage: c.n()
(1.22727272727273, -0.590909090909091)
```

Then, for every $c-$value, we need to round to the nearest integer. One way to do this is

```
sage: for x in c:
          a = round(x)
          print(a)
1
-1
```

This will give us 1 *and* -1*. All that is left is to write* $\mathbf{v} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2$*. We'll do this by referencing the columns of* V*.*

```
sage: v = 1*V[:,0] + -1*V[:,1]
sage: v
[ 6]
[-4]
```

This gives us that $\mathbf{v} = \begin{bmatrix} 6 \\ -4 \end{bmatrix}$*. To check that* $\mathbf{v}$ *is close to* $\mathbf{w}$*, we need* $\mathbf{v}$ *to be written as a vector, so we write*

```
sage: v1 = vector([6,-4])
```

and then compute

```
sage: (v1-w).norm()
sqrt(5)
```

to see that $||\mathbf{v}-\mathbf{w}|| = \sqrt{5}$*.*

Remark 5.3.1. *For a discussion on solving the SVP in ideal lattices algebraically, see for example [4] and [5].*

5.4 Exercises

1. Let $\mathscr{L}$ be the lattice spanned by $\mathbf{v}_1 = \begin{bmatrix} 8 \\ -3 \end{bmatrix}$ and $\mathbf{v}_2 = \begin{bmatrix} 1 \\ 14 \end{bmatrix}$.

   (a) Find the volume of a fundamental domain of $\mathscr{L}$.

   (b) Find another basis for $\mathscr{L}$. It may be helpful to draw a picture of the 2−dimensional lattice.

   (c) Find the volume of a fundamental domain of $\mathscr{L}$ using your basis from part (b). How does this compare to your answer from part (a)?

2. Let $\mathscr{L}$ be an integral lattice in $\mathbb{R}^2$ that is generated by the basis $\mathscr{B} = \left\{ \mathbf{v}_1 = \begin{bmatrix} 7 \\ -1 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right\}$.

   Suppose $\mathbf{w} = \begin{bmatrix} 8 \\ 9 \end{bmatrix} \in \mathbb{R}^2$.

   (a) Use Babai's algorithm to find a vector $\mathbf{v} \in \mathscr{L}$ that is close to $\mathbf{w}$.

   (b) Let $\mathscr{B}' = \left\{ \mathbf{v}_1' = \begin{bmatrix} -5 \\ 7 \end{bmatrix}, \mathbf{v}_2' = \begin{bmatrix} -9 \\ 17 \end{bmatrix} \right\}$ be another basis for $\mathscr{L}$. Use Babai's algorithm to find a vector $\mathbf{v}' \in \mathscr{L}$ that is close to $\mathbf{w}$.

   (c) Compare with your answers from parts (a) and (b). Which is a "better" answer? Why?

3. Let $\mathscr{L}$ be the lattice generated by $\left\{ \mathbf{v}_1 = \begin{bmatrix} 7 \\ -1 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right\}$. Verify that $\mathbf{v}_1$ and $\mathbf{v}_2$ are reasonably orthogonal. Find the closest vector in $\mathscr{L}$ to the target point $\mathbf{w} = \begin{bmatrix} 328 \\ 133 \end{bmatrix}$.

4. Explain why Babai's Closest Vector Algorithm requires a "reasonably orthogonal" basis.

5. Let $\mathbf{v}_1 = \begin{bmatrix} 10 \\ -9 \\ 4 \end{bmatrix}$, $\mathbf{v}_2 = \begin{bmatrix} 17 \\ 11 \\ -3 \end{bmatrix}$, and $\mathbf{v}_3 = \begin{bmatrix} -6 \\ 13 \\ 5 \end{bmatrix}$.

   (a) Verify that $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ is a basis for some lattice $\mathscr{L}$.

   (b) Use the Hadamard Ratio to determine if this is a good basis or a bad one.

5.5 Computer Exercises

1. Write a short code that inputs basis vectors and outputs the result of Hadamard's Ratio to check for reasonably orthogonal vectors. Verify your answer from Problem 4 above.

2. Refer to Algorithm 3. Write Python code that executes Babai's Algorithm.

3. (We use an example from [14].) Let $\mathscr{L} \subset \mathbb{R}^2$ be the lattice given by the basis $\mathbf{v}_1 = \begin{bmatrix} 137 \\ 312 \end{bmatrix}$ and $\mathbf{v}_2 = \begin{bmatrix} 215 \\ -187 \end{bmatrix}$. Find the closest vector in $\mathscr{L}$ to target point $\mathbf{w} = \begin{bmatrix} 53172 \\ 81743 \end{bmatrix}$.

4. Complete the last problem again, but this time with basis $\mathbf{v}_1' = \begin{bmatrix} 1975 \\ 438 \end{bmatrix}, \mathbf{v}_2' = \begin{bmatrix} 7548 \\ 1627 \end{bmatrix}$.

5. Suppose you have a good basis for lattice $\mathscr{L}$ of $\mathscr{B} = \left\{ \begin{bmatrix} 2 \\ -4 \\ 6 \end{bmatrix}, \begin{bmatrix} -3 \\ 4 \\ 5 \end{bmatrix}, \begin{bmatrix} 6 \\ 5 \\ -3 \end{bmatrix} \right\}$. Find the closest lattice point to target vector $\mathbf{w} = \begin{bmatrix} 622 \\ 103 \\ 95 \end{bmatrix}$.

6. Generate a random lattice $\mathscr{L}$ of dimension 5 in Sage. Pick a vector $\mathbf{w}$ not in $\mathscr{L}$. (Recall that you can check this with Sage.). Solve the CVP for $\mathbf{w}$ in $\mathscr{L}$.

CHAPTER

6

LATTICE-BASED PUBLIC KEY CRYPTOSYSTEMS

We now discuss some of the most promising lattice-based cryptosystems, the GGH of Golreich, Goldwasser, and Halevi [11], and the NTRUEncrypt of Hoffstein, Pipher, and Silverman [13].

6.1 GGH

The GGH Cryptosystem is named for its creators, Goldreich, Goldwasser, and Halevi. [11]. It is based on the computer's inability to solve the CVP in a reasonable amount of time. Before we get into the discussion, we need one quick definition. The rest of the cryptosystem, we will find, is very familiar.

Definition 65 (Unimodular Matrix)**.**
A matrix $A \in \mathbb{Z}^{n\times n}$ *is* **unimodular** *if* $\det(A) = \pm 1$. *Equivalently,* $A \in \mathbb{Z}^{n\times n}$ *is unimodular if* $A^{-1} \in \mathbb{Z}^{n\times n}$.

Remark 6.1.1. *Two bases for the same lattice, expressed in matrix form, are related by a unimodular matrix. [5]*

We will now proceed with the construction of the GGH Cryptosystem.

Key Creation

Alice chooses a *reasonably orthogonal* basis $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ of $\mathbb{Z}^n$. The closer the Hadamard ratio $\mathscr{H}(\mathscr{B})$ is to 1, the more orthogonal the basis vectors are. She keeps her good basis $\mathscr{B}$ private. Alice then forms a new basis $\mathscr{B}' = \{\mathbf{v}'_1, \mathbf{v}'_2, \ldots, \mathbf{v}'_n\}$, where the vectors are *reasonably parallel*. If we let $V = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_n \end{bmatrix}$ be the matrix with Alice's good basis as columns, we can find a bad basis by computing $V' = VU$, where U is a unimodular matrix. Note that U should not be the identity matrix, and Alice should check the Hadamard Ratio of the columns of V' to be sure that they are reasonably parallel. One way to generate U is to take a product of a large number of randomly chosen elementary matrices. [14] Elementary matrices are matrices that are obtained by applying one row operation to the identity matrix. The columns of V' form the bad basis, $\mathscr{B}'$. Alice publishes $\mathscr{B}'$ as her public key.

Encryption

Suppose that Bob wants to send the plaintext message $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ to Alice. He uses the elements of $\mathbf{p}$ as coefficients in a linear combination of the bad basis vectors. That is, he computes $\mathbf{m} = p_1\mathbf{v}'_1 + p_2\mathbf{v}'_2 + \cdots + p_n\mathbf{v}'_m \in \mathbb{Z}^n$. Bob then chooses a small $\mathbf{r} \in \mathbb{Z}^n$ and computes the *ciphertext* $\mathbf{m}' = \mathbf{m} + \mathbf{r}$. He sends $\mathbf{m}'$ to Alice.

Decryption

Alice receives the ciphertext $\mathbf{c}$ from Bob. Since she knows that $\mathbf{c}$ is close to $\mathbf{m}'$, she simply solves the CVP for $\mathbf{c}$ with the private basis $\mathscr{B}$. Since she is using the good basis, she will recover $\mathbf{m}'$. This allows her to then use the public basis $\mathscr{B}'$ to get back the plaintext message $\mathbf{m}$.

Attack

Let us suppose that Eve intercepts the ciphertext message $\mathbf{c}$. To decrypt the message, she needs to solve the CVP for $\mathbf{c}$. But, Eve only knows the public basis $\mathscr{B}'$. Recall that these vectors are nearly parallel. So, the vector she recovers from solving the CVP will likely not be anywhere close to $\mathbf{m}$.

Remark 6.1.2. *The key to the GGH is the random noise vector* $\mathbf{r}$. $\mathscr{B}$ *should be chosen carefully, so that* $\mathbf{r}$ *does not shift the nearest lattice point.*

Example 6.1.1 (GGH)**.**

Key Creation

Suppose Alice has a good basis of $\mathscr{B} = \left\{ \mathbf{v}_1 = \begin{bmatrix} 7 \\ -1 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right\}$*. The lattice* $\mathscr{L}$ *spanned by* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *has determinant* 22 *and Hadamard ratio*

$$\mathscr{H}(\mathbf{v}_1, \mathbf{v}_2) = \left(\frac{\det(\mathscr{L})}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|} \right)^{1/2} \approx 0.991902.$$

Thus, it is clear that $\mathscr{B}$ *is reasonably orthogonal.* $\mathscr{B}$ *is Alice's secret key.*

Let $V = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} 7 & 1 \\ -1 & 3 \end{bmatrix}$.

Alice chooses a unimodular matrix $U = \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix}$ *and computes* $VU = \begin{bmatrix} 17 & 26 \\ 7 & 12 \end{bmatrix}$.

The columns of VU *give a new basis* $\mathscr{B}' = \left\{ \mathbf{v}_1' = \begin{bmatrix} 17 \\ 7 \end{bmatrix}, \mathbf{v}_2' = \begin{bmatrix} 26 \\ 12 \end{bmatrix} \right\}$.

Recall that $\mathscr{L}$ *has determinant* 22*. (One can check that* $|det(VU)| = 22$*.) The Hadamard ratio of* $\mathscr{B}'$ *is*

$$\mathscr{H}(\mathbf{v}_1', \mathbf{v}_2') = \left(\frac{det(\mathscr{L})}{\|\mathbf{v}_1'\| \|\mathbf{v}_2'\|} \right)^{1/2} \approx 0.20442.$$

Thus, it is clear that $\mathscr{B}'$ *is a reasonably parallel basis for* $\mathscr{L}$*.*

Alice publishes $\mathscr{B}'$*, the public key.*

```
sage: V = Matrix([[7,1],[-1,3]])
sage: (abs(det(V))/(V[:,0].norm()*V[:,1].norm()))^(1/2)
0.9919021676052067

sage: U = Matrix([[2,3],[3,5]])
sage: det(U)
1

sage: W = V*U
sage: W
[17 26]
```

```
[ 7 12]

sage: (abs(det(W))/(W[:,0].norm()*W[:,1].norm()))^(1/2)
0.20442250330688122
```

Encryption

Bob has the public key $\mathscr{B}'$. He wants to send the plaintext message $\mathbf{p} = (24, -3)$ to Alice, but he must first encrypt it.

Bob computes the linear combination of $\mathscr{B}'$ with coefficients from $\mathbf{p}$. $\mathbf{m} = 24\mathbf{v}_1' - 3\mathbf{v}_2' = \begin{bmatrix} 330 \\ 132 \end{bmatrix}$.

Bob then chooses a random noise vector, $\mathbf{r} = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$, *and computes* $\mathbf{m}' = \mathbf{m} + \mathbf{r} = \begin{bmatrix} 328 \\ 133 \end{bmatrix}$.

Bob sends $\mathbf{m}'$ to Alice.

```
sage: m = 24*W[:,0] - 3*W[:,1]
sage: m
[330]
[132]

sage: r = Matrix([[-2],[1]])
sage: mprime = m + r
sage: mprime
[328]
[133]
```

Decryption

Alice receives $\mathbf{m}'$. She uses Babai's Algorithm to solve the CVP for $\mathbf{m}'$ with her private basis $\mathscr{B}$:

$$c_1 \begin{bmatrix} 7 \\ -1 \end{bmatrix} + c_2 \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 328 \\ 133 \end{bmatrix}$$

gives $c_1 \approx 38.3818$ and $c_2 \approx 57.227$. Then $a_1 = \lfloor c_1 \rceil = 39$ and $a_2 = \lfloor c_2 \rceil = 57$.

Alice computes $a_1\mathbf{v}_1 + a_2\mathbf{v}_2 = 39\begin{bmatrix} 7 \\ -1 \end{bmatrix} + 57\begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 330 \\ 132 \end{bmatrix}$. *Note that this is exactly* $\mathbf{m}$.

To recover $\mathbf{p}$, *Alice solves* $p_1\mathbf{v}_1' + p_2\mathbf{v}_2' = \mathbf{m}$.

$$p_1\begin{bmatrix}17\\7\end{bmatrix}+p_2\begin{bmatrix}26\\12\end{bmatrix}=\begin{bmatrix}330\\132\end{bmatrix}$$

gives Alice $\mathbf{p}=(24,-3)$*, which is Bob's plaintext message.*

```
sage: (V.augment(mprime)).rref().n()
[ 1.00000000000000 0.000000000000000  38.6818181818182]
[0.000000000000000  1.00000000000000  57.2272727272727]

sage: s = 39*V[:,0]+57*V[:,1]
sage: s
[330]
[132]

sage: (W.augment(s)).rref()
[ 1  0 24]
[ 0  1 -3]
```

Remark 6.1.3. *Note that when Eve intercepts the message, she will have to solve the CVP using the public key. As we have seen, she will likely not be able to solve it with the bad basis.*

```
sage: (W.augment(mprime)).rref().n()
[ 1.00000000000000 0.000000000000000  21.7272727272727]
[0.000000000000000  1.00000000000000 -1.59090909090909]

sage: a = 22*W[:,0]-2*W[:,1]
sage: a
[322]
[130]

sage: (W.augment(a)).rref()
[ 1  0 22]
[ 0  1 -2]
```

She recovered $\mathbf{p}'=(22,-2)\neq\mathbf{p}$.

As we will see later on, there is a way for her to reduce the bad basis to a good one in order to help her solve the CVP in small dimensions.

6.2 NTRU

The NTRU Public Key Cryptosystem is due to Hoffstein, Pipher, and Silverman. [13] We begin with a discussion of a Congruential Public Key Cryptosystem, the lowest dimensional NTRU, following [14], before proceeding to the NTRUEncrypt [13].

6.2.1 Congruential Public Key Cryptosystem

Key Creation

Alice first chooses a large $q \in \mathbb{Z}_{>0}$. She then chooses $f, g \in \mathbb{Z}_{>0}$ that are small relative to q and such that

1. $f < \sqrt{\frac{q}{2}}$
2. $\sqrt{\frac{q}{4}} < g < \sqrt{\frac{q}{2}}$
3. $\gcd(f, qg) = 1$

Alice then computes $h \equiv f^{-1}g \pmod{q}$. Note that $0 < h < q$. She keeps her *private key* (f, g) a secret and publishes her *public key* (q, h).

Encryption

Bob wants to send a *plaintext message* m to Alice. He chooses m and a random $r \in \mathbb{Z}$ such that

1. $0 < m < \sqrt{\frac{q}{4}}$ and
2. $0 < r < \sqrt{\frac{q}{2}}$

Bob computes the *ciphertext* $c \equiv rh + m \pmod{q}$, noting that $0 < c < q$. He then sends the *ciphertext* c to Alice.

Decryption

Alice computes $a \equiv fc \pmod{q}$ so that $0 < a < q$. She then computes $m' \equiv f^{-1}a \pmod{g}$, where $0 < m' < g$. Alice recovers the *plaintext message* $m' = m$.

Proposition 6.2.1. *In the Congruential Public Key Cryptosystem,* $m' = m$. *That is,* $f^{-1}a \pmod{g} = m$.

Proof. We first consider a that Alice computes during decryption.

$$\begin{aligned}
a &= f\,c \\
&\equiv f(r\,h+m)(\mod q) && \text{(replacing } c\text{)} \\
&\equiv f\,r\,h+f\,m(\mod q) && \text{(distributing } f\text{)} \\
&\equiv f\,r\,f^{-1}g+f\,m(\mod q) && \text{(replacing } h\text{)} \\
&\equiv r\,g+f\,m(\mod q) && \text{(since } f\,f^{-1}=1\text{)}
\end{aligned}$$

From our construction, we note that

$$\begin{aligned}
r\,g+f\,m &\leq \sqrt{\frac{q}{2}}\sqrt{\frac{q}{2}}+\sqrt{\frac{q}{2}}\sqrt{\frac{q}{4}} \\
&< \frac{q}{2}+\frac{q}{2} \\
&< q
\end{aligned}$$

So, we now have that $a = r\,g+f\,m$.

Now, we consider m', the message Alice gets after decrypting c.

$$\begin{aligned}
m' &= f^{-1}a(\mod g) \\
&= f^{-1}(r\,g+f\,m)(\mod g) && \text{(replacing } a\text{)} \\
&= f^{-1}(0+f\,m) && \text{(since } r\,g\equiv 0(\mod g)\text{)} \\
&= f^{-1}f\,m \\
&= m && \text{(since } f^{-1}f=1\text{)}
\end{aligned}$$

□

Example 6.2.1 (Congruential Public Key Cryptosystem)**.**

Key Creation

Alice picks $q = 100$ *and chooses private key* $(f,g) = (7,6)$. *Note that* $f < \sqrt{\frac{q}{2}} = \sqrt{50}$ *and* $5 = \sqrt{\frac{q}{4}} < g < \sqrt{\frac{q}{2}} = \sqrt{50}$. *She then computes* $h \equiv f^{-1}g(\mod q) = 43\cdot 6(\mod 100) = 58$. *She publishes her public key* $(q,h) = (100,58)$.

Encryption

Bob wants to send the plaintext message $m = 2$ *to Alice. He chooses* $r = 3$. *Note that* $0 <$

$m < \sqrt{\frac{q}{4}} = 25$ *and* $0 < r < \sqrt{\frac{q}{2}} = \sqrt{50}$. *He computes* $c \equiv rh + m \pmod{q} = 3 \cdot 58 + 2 \pmod{100} = 176 \pmod{100} = 76$. *He sends ciphertext* $c = 76$ *to Alice.*

Decryption

Alice receives $c = 76$ *and computes* $a \equiv fc \pmod{q} = 7 \cdot 76 \pmod{100} = 32$. *She then computes* $m' \equiv f^{-1}a \pmod{g} = 1 \cdot 32 \pmod{6} = 2$. *Alice has recovered the plaintext message* $m' = m = 2$.

Example 6.2.2 (Congruential Public Key Cryptosystem with Sage)**.**
We work through another example using SageMath.

Key Creation

Alice picks $q = 8675309$ *and chooses private key* $(f, g) = (1642, 1733)$. *She checks that her conditions are met, calculates* h*, and publishes her public key* $(q, h) = (8675309, 681557)$.

```
sage: q = 8675309
sage: q.is_prime()
True

sage: (sqrt(q/2)).n()
2082.70365150686

sage: (sqrt(q/4)).n()
1472.69387518248

sage: f = 1642
sage: g = 1733
sage: gcd(f, q*g)
1

sage: h = mod(f^(-1)*g, q)
sage: h
681557
```

Encryption

Bob wants to send the message $m = 1130$ *to Alice. He chooses* $r = 1822$ *and computes* $c \equiv rh + m \pmod{q} = 1228797$ *to send to Alice.*

```
sage: m = 1130
sage: r = 1822
sage: c = mod(r*h+m, q)
sage: c
1228797
```

Decryption
Alice receives $c = 1228797$ *and computes* $a \equiv f\,c(\mod q)$ *and then* $m' \equiv f^{-1}a(\mod g)$ *to recover* $m' = m$.

```
sage: a = mod(f*c,q)
sage: a
5012986

sage: mod(f^(-1),g)*mod(a,g)
1130
```

This congruential public key cryptosystem can be broken by using the two-dimensional Gaussian Lattice Reduction Algorithm in Section 7.1.

In higher dimensions, NTRU is susceptible to the LLL lattice basis reduction algorithm.

6.2.2 NTRUEncrypt

Before diving into the NTRUEncrypt Cryptosystem, we need a few more definitions and ideas. We follow [14].

Definition 66 (Trinary Polynomial)**.**
A **trinary polynomial** *is a polynomial in* $\mathscr{T}(d_1,d_2)$ *, where*

$$\mathscr{T}(d_1,d_2)=\left\{\begin{array}{ll} & \mathbf{a}(x) \text{ has } d_1 \text{ coefficients equal to } 1 \\ \mathbf{a}(x) & \mathbf{a}(x) \text{ has } d_2 \text{ coefficients equal to } -1 \\ & \mathbf{a}(x) \text{ has all other coefficients equal to } 0 \end{array}\right.$$

for any $d_1, d_2 \in \mathbb{Z}_{>0}$.

Example 6.2.3. *In* $\mathscr{T}(2,1)$*, we could have* $f(x) = x^5 - x^3 + 1$.

Definition 67 (Center Lift)**.**
The **center lift** *of* $f(x) \in \dfrac{\mathbb{Z}_p[x]}{(x^n-1)}$ *to a polynomial in* $\dfrac{\mathbb{Z}[x]}{(x^n-1)}$ *is* $\hat{f}(x)$ *such that*

1. *$\hat{f}(x)$ is congruent to $f(x)(\text{mod } p)$ and*

2. *the coefficients f_i are such that $-\frac{1}{2}p < f_i \leq \frac{1}{2}p$.*

Example 6.2.4 (Center Lift)**.**
Suppose we have $f(x) = 7x^4 + 3x^2 + 2x + 9 \in \mathbb{Z}_{11}[x]$. All coefficients must be between -5.5 and 5.5. So, $\hat{f}(x) = -4x^4 + 3x^2 + 2x - 2$.

Remark 6.2.1. *For p, q distinct primes and n prime, we will let R, R_p, and R_q be the following polynomial rings*

$$R = \frac{\mathbb{Z}[x]}{(x^n - 1)}, \qquad R_p = \frac{\mathbb{Z}_p[x]}{(x^n - 1)}, \qquad R_q = \frac{\mathbb{Z}_q[x]}{(x^n - 1)}.$$

As before, any polynomial in R can be written in either R_p or R_q by reducing its coefficients mod p or q, respectively.

We are now ready to discuss one of the most promising cryptosystem for quantum-resistance, the NTRUEncrypt.

Public Parameters
The NTRUEncrypt has public parameters (n, p, q, d), where n and p are prime, $\gcd(p, q) = 1$, $\gcd(n, q) = 1$, and $q > (6d + 1)p$.

Key Creation
Alice chooses a private key $f(x) \in \mathscr{T}(d + 1, d)$ and $g(x) \in \mathscr{T}(d, d)$. It must be the case that $f(x)$ has an inverse in both R_p and R_q. If the polynomial she chooses is not invertible in both R_p and R_q, she discards it and picks another. She chooses $f(x) \in \mathscr{T}(d + 1, d)$ because polynomials in $\mathscr{T}(d, d)$ do not have inverses in R_q.

We will denote the inverse of $f(x)$ in R_p by $F_p(x)$ and the inverse of $f(x)$ in R_q by $F_q(x)$. That is,

$$F_p(x)f(x) = 1(\text{mod } p) \quad \text{and} \quad F_q(x)f(x) = 1(\text{mod } q).$$

Alice then computes $h(x) = F_q(x)g(x)$ in R_q. $h(x)$ is Alice's *public key*. Her *private key* is $(f(x), g(x))$.

Encryption
Suppose Bob wants to send a message $m(x)$ to Alice. The plaintext message, $m(x) \in R$, that Bob wants to send must be the center lift of a polynomial in R_p. That is, Bob's plaintext message $m(x) \in R$ has coefficients m_i such that $-\frac{1}{2}p < m_i \leq \frac{1}{2}p$. Bob then picks a random

polynomial $r(x) \in \mathscr{T}(d,d)$ and computes $e(x) = p\,r(x)h(x) + m(x)$ in R_q. $e(x)$ is the ciphertext that Bob sends to Alice.

Decryption

Alice computes $a(x) = e(x)f(x)$ in R_q. She then center-lifts $a(x)$ to $\hat{a}(x)$ in R and computes $b(x) = \hat{a}(x)F_p(x)$ in R_p. Alice gets that $b(x) \equiv m(x)$.

Example 6.2.5 (NTRUEncrypt)**.**

Suppose Alice and Bob have agreed to the public parameters $(n,p,q,d) = (3,3,101,1)$. *Note that* $q = 101 > (6d+1)p = 21$. *We have*

$$R = \frac{\mathbb{Z}[x]}{(x^3-1)}, \qquad R_3 = \frac{\mathbb{Z}_3[x]}{(x^3-1)}, \qquad \textit{and} \qquad R_{101} = \frac{\mathbb{Z}_{101}[x]}{(x^3-1)}.$$

Key Creation

Alice picks $f(x) = x^2 + x - 1 \in \mathscr{T}(2,1)$ *and* $g(x) = x - 1 \in \mathscr{T}(1,1)$. *She computes the inverse of* $f(x)$ *in* R_3 *to get* $F_3(x) = 2x^2 + 2x$ *and in* R_{101} *to get* $F_{101}(x) = 51x^2 + 51x$. *Alice then computes* $h(x) = F_{101}(x)g(x)$ *in* R_{101}. *This gives* $h(x) = (51x^2 + 51x)(x-1) = 51x^3 - 51x = -51x + 51$ *in* R_{101}. *She publishes her public key* $h(x) = -51x + 51$.

Encryption

Suppose Bob wants to send a message $m(x) = x$ *to Alice. He chooses a random* $r(x) = x^2 - 1 \in \mathscr{T}(1,1)$ *and computes his ciphertext* $e(x) = p\,h(x)r(x) + m(x)$ *in* R_{101}. *This gives* $e(x) = 3(-51x + 51)(x^2 - 1) + x = 52x^2 + 53x - 3$. *He sends* $e(x)$ *to Alice.*

Decryption

Alice receives $e(x)$ *and computes* $a(x) = e(x)f(x)$ *in* R_{101}. *So,* $a(x) = (x^2 + x - 1)(52x^2 + 53x - 3) = -3x^2 - 4x + 7$. *She then center lifts* $a(x)$ *to* $\hat{a}(x) = -3x^2 - 4x + 7$ *and computes* $b(x) = \hat{a}(x)F_3(x)$ *in* R_3. *She gets* $b(x) = (-3x^2 - 4x + 7)(2x^2 + 2x) = (-x + 1)(2x^2 + 2x) = 4x + 3 = x$. *Note that Alice has recovered the plaintext message as* $b(x) = m(x)$.

We will illustrate the decryption of NTRU with another example, from [14].

Example 6.2.6 (NTRU)**.**

This NTRUEncrypt has public parameters $(n,p,q,d) = (7,3,37,2)$. *Alice's private key is* $(f(x), F_p(x))$, *where*

$$f(x) = -1 + x - x^3 + x^4 + x^5 \quad \textit{and} \quad F_3(x) = 1 + x - x^2 + x^4 + x^5 + x^6.$$

The reader should verify that $F_3 f \equiv 1 (mod\, 3)$. *Alice receives the ciphertext*

$$e(x) = 2 + 8x^2 - 16x^3 - 9x^4 - 18x^5 - 3x^6$$

from Bob. To decrypt the message, Alice first computes $a(x) = e(x)f(x)$ *in* R_{37}.

$$\begin{aligned} a(x) &= e(x)f(x) \\ &= (2 + 8x^2 - 16x^3 - 9x^4 - 18x^5 - 3x^6)(-1 + x - x^3 + x^4 + x^5) \\ &= 33 + 32x + 5x^2 + x^3 + 29x^4 + 3x^5 + 9x^6 \end{aligned}$$

Alice then center-lifts $a(x)$ *to* $\hat{a}(x)$ *in* R *to find* $\hat{a}(x) = -4 - 5x + 5x^2 + x^3 - 8x^4 + 3x^5 + 9x^6$. *Lastly, she computes* $b(x) = \hat{a}(x)F_3(x)$ *in* R_3.

$$\begin{aligned} b(x) &= \hat{a}(x)F_3(x) \\ &= (-4 - 5x + 5x^2 + x^3 - 8x^4 + 3x^5 + 9x^6)(1 + x - x^2 + x^4 + x^5 + x^6) \\ &= x + 2x^4 + x^6 \end{aligned}$$

Hence, the plaintext message was $x + 2x^4 + x^6$.

Theorem 6.2.1. *If the NTRUEncrypt parameters* (n, p, q, d) *satisfy the condition that* $q > (6d+1)p$, *then* $b(x) = m(x)$, *where* $b(x)$ *is the polynomial Alice finds during decryption, and* $m(x)$ *is Bob's plaintext message.*

Proof. We begin by considering Alice's polynomial $a(x)$.

$$\begin{aligned} a(x) &= f(x)e(x) && (\text{in } R_q) \\ &= f(x)[ph(x)r(x) + m(x)] && (\text{replacing ciphertext } e(x)) \\ &= pf(x)h(x)r(x) + f(x)m(x) && (\text{distributing } f(x)) \\ &= pf(x)F_q(x)g(x)r(x) + f(x)m(x) && (\text{replacing } h(x)) \\ &= pg(x)r(x) + f(x)m(x) && (\text{since } f(x)F_q(x) \equiv 1 (\text{mod } q)) \end{aligned}$$

We now consider the size of the coefficients in the polynomial $pg(x)r(x) + f(x)m(x)$ in R.

Let us first look at $pg(x)r(x)$. If $g(x) = g_0 + g_1 x + g_2 x^2 + \cdots + g_{n-1}x^{n-1}$ and $r(x) = r_0 + r_1 x + r_2 x^2 + \cdots + r_{n-1}x^{n-1}$, then the leading coefficient of $g(x)r(x)$ is $g_0 r_{n-1} + g_1 r_{n-2} + \cdots + g_{n-1} r_0$. Since $g(x)$ and $r(x)$ are in $\mathcal{T}(d,d)$, the largest possible coefficient for each is d. Thus, the largest possible coefficient for $g(x)r(x)$ is $2d$. Hence, the largest possible coefficient of

$pg(x)r(x)$ is $p2d$.

We now turn to $f(x)m(x)$. If we let $f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$ and $m(x) = m_0 + m_1 x + m_2 x^2 + \cdots + m_{n-1} x^{n-1}$, then the leading coefficient of $f(x)m(x)$ is $f_0 m_{n-1} + f_1 m_{n-2} + \cdots + f_{n-1} m_0$. Recall that the coefficients of $m(x)$ are such that $-\frac{1}{2}p < m_i \leq \frac{1}{2}p$ and that $f(x) \in \mathscr{T}(d+1, d)$. This gives that the largest possible coefficient of $f(x)m(x)$ is $(2d+1)\frac{1}{2}p$.

Together, this gives that the largest possible coefficient of $a(x) = pg(x)r(x) + f(x)m(x)$ is $p2d + (2d+1)\frac{1}{2}p = p\left(2d + d + \frac{1}{2}\right) = p\left(3d + \frac{1}{2}\right)$. We made the assumption that $(6d+1)p < q$, and dividing by 2 gives us $p\left(3d + \frac{1}{2}\right) \leq \frac{1}{2}q$. This tells us that, when Alice calculates $a(x) = pg(x)r(x) + f(x)m(x)$ in R_q and lifts it to R, she gets the exact values for the coefficients. That is, $a(x) = pg(x)r(x) + f(x)m(x)$ in R.

We now consider $b(x)$, calculated in R_p.

$$\begin{aligned}
b(x) &= F_p(x)a(x) \\
&= F_p(x)[pg(x)r(x) + f(x)m(x)] && \text{(replacing } a(x)\text{)} \\
&= F_p(x)[0 + f(x)m(x)] && \text{(since } pg(x)r(x) \equiv 0 (\text{mod } p)\text{)} \\
&= F_p(x)f(x)m(x) \\
&= m(x) && \text{(since } F_p(x)f(x) \equiv 1 (\text{mod } p)\text{)}
\end{aligned}$$

□

Example 6.2.7 (NTRU with Sage)**.**
We will work through another problem using Sage.
Suppose we have public parameters $(n, p, q, d) = (5, 3, 41, 2)$. *First, we'll define our polynomial ring* $R_3 = \frac{\mathbb{Z}_3[x]}{(x^5-1)}$.

```
sage: R.<x> = PolynomialRing(GF(3),x)
sage: R3.<x> = R.quotient_ring(x^5-1)
```

Next, Alice can choose her $f3 = f(x) = -x^4 + x^3 + x^2 - x + 1 \in \mathscr{T}(3,2)$ *and compute* $F3 = F_3 = f^{-1}(x) = 2x^3 + 2x^2$ *in* R_3*:*

```
sage: f3 = R3(-x^4+x^3+x^2-x+1)
sage: F3 = f3^(-1)
```

```
sage: F3
2*x^3 + 2*x^2
```

Let's define $R41 = R_{41} = \frac{\mathbb{Z}_{41}[x]}{(x^5-1)}$ *and calculate* $F41 = f^{-1}(x)$ *in* R_{41}.

```
sage: S.<x> = PolynomialRing(GF(41),x)
sage: R41.<x> = S.quotient_ring(x^5-1)
sage: f41 = R41(-x^4+x^3+x^2-x+1)
sage: F41 = f41^(-1)
sage: F41
21*x^3 + 21*x^2
```

This returns $F_{41} = 21x^3 + 21x^2$. *Alice then chooses* $g(x) = x^4 + x^3 - x^2 - x \in \mathscr{T}(2,2)$ *and computes* $h(x) = F_{41}(x)g(x) = 40x^4 + 20x^3 + 21x^2 + x$ *by doing the following:*

```
sage: g = R41(x^4+x^3-x^2-x)
sage: h = F41*g
sage: h
40*x^4 + 20*x^3 + 21*x^2 + x
```

To encrypt his plaintext message $m(x) = x^2 + x - 1$, *Bob chooses* $r(x) = x^3 - x^2 + x - 1$ *and calculates ciphertext* $e(x) = 3h(x)r(x) + m(x)$ *by doing*

```
sage: m = R41(x^2+x-1)
sage: r = R41(x^3-x^2+x-1)
sage: e = 2*h*r+m
sage: e
3*x^4 + 20*x^2 + 20*x + 40
```

This code returns $e(x) = 3x^4 + 20x^2 + 20x + 40$, *and then e is sent to Alice.*

To decrypt $e(x)$, *Alice computes* $a(x) = e(x)f(x)$

```
sage: a = e*f41
sage: a
3*x^4 + 37*x^3 + 2*x^2 + 4*x + 37
```

to get that $a(x) = 3x^4 + 37x^3 + 2x^2 + 4x + 37$. *All that is left to do is for Alice to center lift* $a(x)$ *to* $\hat{a}(x)$ *and compute* $b(x) = \hat{a}(x)F_3(x)$. *She does this by doing*

```
sage: ahat = ZZ['x']([coeff.lift_centered() for coeff in a.lift()])
sage: ahat
3*x^4 - 4*x^3 + 2*x^2 + 4*x - 4

sage: b = ahat*F3
sage: b
x^2 + x + 2
```

This gives that $b(x) = x^2 + x + 2$*, which is congruent to* $m(x) = x^2 + x - 1 (\mod 3)$.

The hard problem associated with the NTRU public key cryptosystem is as follows:

Problem 6.2.1 (NTRU Key Recovery)**.**
Given public key $h(x)$*, find trinary polynomials* $f(x)$ *and* $g(x)$ *such that* $f(x)h(x) = g(x)$ *in* R_q*.*

One of the ways that an adversary can attack the NTRU is by framing the problem as a lattice problem. We briefly discuss this reframing, and we discuss the attack in the next two chapters.

6.2.3 NTRUEncrypt with Lattices

Given Alice's public key $h(x) = h_0 + h_1 x + \cdots + h_{n-1} x^{n-1}$, we can construct the NTRU matrix M_h associated with $h(x)$ as follows:

$$M_{\mathbf{h}} = \left[\begin{array}{c|c} I_n & \mathbf{h} \\ \hline 0 & qI_n \end{array}\right]$$

where $\mathbf{h}$ is the matrix of the coefficients of $h(x) \bmod (x^n - 1)$, $x h(x) \bmod (x^n - 1)$, …, $x^{n-1} h(x) \bmod (x^n - 1)$ as rows.

Remark 6.2.2. *The NTRU matrix* $M_{\mathbf{h}}$ *is a* $2n \times 2n$ *matrix.*

We can view $\mathbf{h}$ as the matrix of cyclical permutations of the coefficients of $h(x)$. Then the NTRU matrix is

$$M_{\mathbf{h}} = \left[\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{n-1} \\ 0 & 1 & \dots & 0 & h_{n-1} & h_0 & \dots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & h_1 & h_2 & \dots & h_0 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array}\right]$$

Remark 6.2.3. *The rows of M_h span the lattice $\mathcal{L}_{\mathbf{h}}$ associated with $h(x)$.*

Example 6.2.8 (NTRU Matrix)**.**
Suppose we have $(n,p,q,d) = (3,3,101,1)$ and $h(x) = 51 - 51x$, as in Example 6.2.5. Then the NTRU matrix associated with $h(x)$ is

$$M_{\mathbf{h}} = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 51 & -51 & 0 \\ 0 & 1 & 0 & 0 & 51 & -51 \\ 0 & 0 & 1 & -51 & 0 & 51 \\ \hline 0 & 0 & 0 & 101 & 0 & 0 \\ 0 & 0 & 0 & 0 & 101 & 0 \\ 0 & 0 & 0 & 0 & 0 & 101 \end{array}\right]$$

and the lattice $\mathcal{L}_{\mathbf{h}}$ is spanned by the rows of $M_{\mathbf{h}}$.

Remark 6.2.4. *Often, it is convenient to view the NTRU martrix associated with $h(x)$ as a 2×2 block matrix with real coefficients $\begin{bmatrix} I & \mathbf{h} \\ 0 & qI \end{bmatrix}$.*

In R_q, we know that $h(x) = F_q(x)g(x)$, so $f(x)h(x) = g(x)$. Then, in R, we have that

$$g(x) = f(x)h(x) + q\,u(x) \tag{6.1}$$

for some $u(x) \in R$.

If we identify the pair of polynomials

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1}x^{n-1} \text{ and } u(x) = u_0 + u_1 x + u_2 x^2 + \cdots + u_{n-1}x^{n-1}$$

with the vector of their coefficients

$$(\mathbf{f},\mathbf{u}) = (f_0, f_1, \ldots, f_{n-1}, u_0, u_1, \ldots, u_{n-1}) \in \mathbb{Z}^{2n}$$

and consider

$$(\mathbf{f},\mathbf{u})M_{\mathbf{h}},$$

we see that

$$(\mathbf{f},\mathbf{u})\begin{bmatrix} I & \mathbf{h} \\ 0 & qI \end{bmatrix} = (\mathbf{f}I, \mathbf{f}\mathbf{h} + \mathbf{u}qI)$$

By (6.1), we then have that

$$(\mathbf{f},\mathbf{u})M_{\mathbf{h}} = (\mathbf{f},\mathbf{g}).$$

This says that we can obtain the vector $(\mathbf{f},\mathbf{g})$ by taking some integer linear combination of the rows of the NTRU matrix $M_{\mathbf{h}}$. Thus, $(\mathbf{f},\mathbf{g})$ is in the NTRU lattice $\mathscr{L}_{\mathbf{h}}$. Note that, since $f(x)$ and $g(x)$ are trinary polynomials, $(\mathbf{f},\mathbf{g})$ is a *short vector* in $\mathscr{L}_{\mathbf{h}}$.

Remark 6.2.5. *Other promising quantum-resistant cryptosystems are based on the learning with errors (LWE) problem. See [1] for a summary and discussion of this.*

6.3 Exercises

1. Suppose Alice uses the GGH Cryptosystem with a private basis of $\mathscr{B} = \left\{\mathbf{v}_1 = \begin{bmatrix} 4 \\ 13 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} -57 \\ -45 \end{bmatrix}\right\}$ and public basis $\mathscr{B}' = \left\{\mathbf{v}_1' = \begin{bmatrix} 25453 \\ 9091 \end{bmatrix}, \mathbf{v}_2' = \begin{bmatrix} -16096 \\ -5749 \end{bmatrix}\right\}$.

   (a) Compute the determinant of Alice's lattice and the Hadamard ratio of $\mathscr{B}$ and $\mathscr{B}'$. Interpret the meaning of each Hadamard ratio.

   (b) Bob sends Alice the encrypted message $\mathbf{m}' = \begin{bmatrix} 155340 \\ 55483 \end{bmatrix}$. Use Alice's private basis to decrypt the message and recover the plaintext. Also, determine Bob's perturbation vector $\mathbf{r}$.

   (c) Eve intercepts the encrypted message. Use the public basis to try to decrypt the message. Is the output equal to the plaintext?

2. Let $\mathscr{B} = \left\{\begin{bmatrix} 7 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}\right\}$ and $\mathscr{C} = \left\{\begin{bmatrix} 17 \\ 7 \end{bmatrix}, \begin{bmatrix} 26 \\ 12 \end{bmatrix}\right\}$ be bases for lattice $\mathscr{L}$.

   (a) Determine which basis is a good basis and which is a bad one.

   (b) Alice uses the good basis as her private key in the GGH Cyrptosystem and publishes the bad basis as the public key. Bob wants to send Alice the plaintext message $P = (24, -3)$. Compute $\mathbf{m}$.

   (c) Use the noise vector $\mathbf{r} = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$ to help Bob compute ciphertext $\mathbf{m}'$ to send to Alice.

   (d) Find $\mathbf{m}$ using Alice's private key, and use $\mathbf{m}$ to recover the plaintext message.

   (e) Suppose Eve intercepts the ciphertext message. Try to decrypt the ciphertext using the bad basis. Can Eve recover P?

3. Alice chooses a private basis $\mathscr{B} = \left\{\begin{bmatrix} 1 \\ 45 \end{bmatrix}, \begin{bmatrix} 45 \\ -1 \end{bmatrix}\right\}$ and a public basis $\mathscr{B}' = \left\{\begin{bmatrix} 361 \\ 37 \end{bmatrix}, \begin{bmatrix} 1850 \\ 184 \end{bmatrix}\right\}$ for the GGH Cryptosystem.

   (a) Bob wants to send Alice the plaintext message $P = (35, 27)$. Use the perturbation vector $\mathbf{r} = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$ to encrypt his message to send to Alice.

   (b) Use the private basis to help Alice decrypt the ciphertext and retrieve the plaintext message.

(c) Eve intercepts the message. Try to decrypt the ciphertext from Eve's perspective. Can she recover the plaintext message?

4. Alice and Bob are using the GGH Cryptosytem to exchange messages. Alice chooses a good basis for her lattice to be $\mathscr{B} = \left\{ \begin{bmatrix} 1 \\ 6 \end{bmatrix}, \begin{bmatrix} 8 \\ 2 \end{bmatrix} \right\}$.

   (a) Verify that $\mathscr{B}$ is a good basis.

   (b) Use unimodular matrix $U = \begin{bmatrix} 2 & 5 \\ 1 & 2 \end{bmatrix}$ to find Alice's public basis.

   (c) Bob wants to encrypt the plaintext message $P = (13, -7)$. He picks $\mathbf{r} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$. What is his ciphertext message?

   (d) Use Alice's private basis to decrypt the ciphertext message and recover P.

5. **Collaborative** (Groups of 3). In this problem, you will create your own 2−dimensional GGH Cryptosystem example. You will take on the role of each of Alice, Bob, and Eve.

   (a) (Alice) Individually, pick a private basis, and compute a public basis. Keep your private basis a secret. Publish your public basis (give it to the other members of your group). There are now three different public bases.

   (b) (Bob) Select one of the public bases that is not your own. Be certain that everyone has picked a different one. Decide the plaintext message that you wish to send to Alice. Use the public basis that you chose to encrypt the message. Send it to the creator of the basis.

   (c) (Alice) You should have just received a ciphertext message. Use your private key to recover the plaintext message.

   (d) (Eve) Intercept the message that you did not create or receive. Use the appropriate public basis to try and decrypt the message. Did you recover the plaintext message?

6. In the Congruential Public Key Cryptosystem, let $q = 8675309$, $f = 1642$, and $g = 1733$.

   (a) Compute h.

   (b) Bob wants to send Alice $m = 1130$. Use $r = 1822$ to calculate his ciphertext.

   (c) Decrypt c to recover m.

7. Let $f(x) = 5x^2 + 3x - 2$ and $g(x) = -3x^2 - 4x - 1$. Find $f(x)g(x)$ in $\dfrac{\mathbb{Z}[x]}{x^3 - 1}$ by hand.

8. Suppose $f(x)=3x^2-2x+1$ and $g(x)=2x^2-4x-7$. Compute $f(x)g(x)$ in $\dfrac{\mathbb{Z}_5[x]}{x^3-1}$ by hand.

9. Find the center lift of $a(x)=5x^4-17x^3+13x^2-4x-20$ in $\mathbb{Z}_{21}[x]$.

10. Suppose $a(x)=-14x^3+7x^2-3x+12\in\dfrac{\mathbb{Z}_{15}[x]}{x^5-1}$. Centerlift $a(x)$ to $a(x)$ in $\dfrac{\mathbb{Z}[x]}{x^5-1}$.

11. In the NTRU Cryptosystem, we are given $h(x)=3+14x-4x^2+13x^3-6x^4+2x^5+7x^6$ and $q=29$. Write the NTRU matrix.

12. Find your own set of public parameters for the NTRUEncrypt that satisfies the conditions.

6.4 Computer Exercises

1. Alice chooses a good basis of $\mathscr{B}=\left\{\mathbf{v}_1=\begin{bmatrix}234\\-673\\254\end{bmatrix},\mathbf{v}_2=\begin{bmatrix}-112\\422\\177\end{bmatrix},\mathbf{v}_3=\begin{bmatrix}43\\633\\79\end{bmatrix}\right\}$.

   (a) Let
   $$U=\begin{bmatrix}2&3&5\\3&2&3\\9&5&7\end{bmatrix}.$$
   Verify that U is unimodular.

   (b) Let $V=\begin{bmatrix}\mathbf{v}_1&\mathbf{v}_2&\mathbf{v}_3\end{bmatrix}$. Compute VU to get the public basis.

   (c) Bob wants to send Alice the plaintext message $P=(163,97,246)$. Encrypt this message using noise vector $\mathbf{r}=\begin{bmatrix}-1\\3\\2\end{bmatrix}$.

   (d) Decrypt the ciphertext from both Alice and Eve's perspectives.

2. Alice uses the GGH Cryptosystem with
   private basis $\mathscr{B}=\left\{\mathbf{v}_1=\begin{bmatrix}27\\-19\\99\end{bmatrix},\mathbf{v}_2=\begin{bmatrix}32\\113\\27\end{bmatrix},\mathbf{v}_3=\begin{bmatrix}41\\24\\-13\end{bmatrix}\right\}$ and
   public basis $\mathscr{B}'=\left\{\mathbf{v}_1'=\begin{bmatrix}551\\630\\189\end{bmatrix},\mathbf{v}_2'=\begin{bmatrix}391\\313\\273\end{bmatrix},\mathbf{v}_3'=\begin{bmatrix}437\\469\\188\end{bmatrix}\right\}$.

(a) Bob sends Alice the encrypted message $\mathbf{m}' = \begin{bmatrix} 727 \\ 139 \\ -246 \end{bmatrix}$. Use Alice's private key to decrypt the message and recover the plaintext.

(b) Find Bob's pertubation vector, $\mathbf{r}$.

(c) Try to decrypt Bob's message from Eve's perspective.

(d) Help Alice recover the plaintext message.

3. Verify your answer from numbers 7 and 8 above with a short computer code.

4. Take f and g as in number 8 above. Find $f^{-1}(x)$ and $g^{-1}(x)$ in $\dfrac{\mathbb{Z}_5[x]}{x^3-1}$, if they exist.

5. Take NTRU public parameters to be $(n,p,q,d) = (11,3,61,3)$. Suppose that Alice chooses $f(x) = 1+x+x^2+x^4-x^6-x^8-x^{10}$ and $g(x) = 1+x^2+x^4-x^6-x^8-x^9$.

(a) Find $F_p(x)$, $F_q(x)$, and $h(x)$.

(b) Suppose that Bob wants to send Alice the plaintext message $m(x) = 1+x+x^3-x^4+x^7$. Using $r(x) = 1-x^3+x^4+x^7-x^9$, calculate the ciphertext $e(x)$.

(c) Help Alice calculate $a(x)$ and use it to decrypt the message.

6. Alice and Bob are using the NTRUEncrypt Public Key Cryptosystem with public parameters $(n,p,q,d) = (7,3,41,2)$. Alice's public key is

$$h(x) = 30x^6+20x^5+40x^4+2x^3+38x^2+8x+26.$$

Bob wants to send Alice the message $m(x) = x^4-x^2+2$.. He chooses the random polynomial $r(x) = -x^5+x^3-x^2+x$.

(a) What is the ciphertext, $e(x)$, that Bob should send to Alice?

(b) Alice has private key $f(x) = x^6-x^4+x^3+x^2-1$. Find $F_3(x)$, and use it to decrypt the ciphertext from part (a).

7. Alice and Bob are using public NTRU parameters $(n,p,q,d) = (11,3,53,2)$. Alice has private $f(x) = x^{10}-x^7+x^4+x^3-x$ and public key

$$h(x) = 29x^{10}+43x^9+12x^8+18x^7+27x^5+13x^4+47x^3+44x^2+13x+19.$$

(a) Bob wants to send $m(x) = x^8+x^7-x^5+x^3+x^2-x+1$ to Alice using random $r(x) = x^{10}-x^7-x^3+x$. What is his ciphertext message?

(b) Help Alice decrypt the ciphertext and recover $m(x)$.

8. (**Collaborative**) Use your parameters in number 12 above to choose private keys $f(x)$ and $g(x)$.

   (a) Compute $h(x)$, and give your public parameters and $h(x)$ to a partner.

   (b) Have your partner encrypt a secret message, $m(x)$, and send you the ciphertext, $e(x)$.

   (c) Use your secret key to decrypt the ciphertext message and recover the plaintext message.

CHAPTER

7

LATTICE REDUCTION ALGORITHMS

As we have seen with Babai's Algorithm, one of the keys to public key cryptosystems is the "good" basis of reasonably orthogonal, short vectors. It is natural to wonder how to turn a "bad" basis into a "good" one. We describe two well-known algorithms here. The 2−D Gaussian Lattice Reduction turns a 2−dimensional bad basis into a good one and therefore solves the SVP in two dimensions. The LLL Lattice Reduction Algorithm reduces any basis to a good one in higher dimensions, and thus solves a version of the SVP in higher dimensions.

7.1 2−D Gaussian Lattice Reduction

Suppose we have basis $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2\} \subset \mathbb{R}^2$ that spans a lattice $\mathscr{L}$. In order to find a shortest vector in $\mathscr{L}$, we use the 2−Dimensional Gaussian Lattice Reduction Algorithm, as presented in [14].

- If $\|\mathbf{v}_2\| < \|\mathbf{v}_1\|$, swap $\mathbf{v}_1$ and $\mathbf{v}_2$.
- Compute $m = \left\lfloor \dfrac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|^2} \right\rceil$. That is, round $\dfrac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|^2}$ to the nearest integer.
- If $m = 0$, then v_1 is the shortest vector.

- If $m \neq 0$, set $\mathbf{v}_2^* = \mathbf{v}_2 - m\mathbf{v}_1$.
  - Note that m is the scalar projection of $\mathbf{v}_2$ onto $\mathbf{v}_1$. This is the same scalar as from the Gram-Schmidt process, but we now round m to the nearest integer so that we are still working with lattice vectors.
  - Note also that the vector $\mathbf{v}_2^*$ is the projection of $\mathbf{v}_2$ onto the orthogonal complement of $\mathbf{v}_1$.
- If $\|\mathbf{v}_2\| < \|\mathbf{v}_1\|$, swap $\mathbf{v}_1$ and $\mathbf{v}_2$, and repeat the process.
- If $\|\mathbf{v}_1\| \leq \|\mathbf{v}_2\|$, stop. $\mathbf{v}_1$ is a shortest nonzero lattice vector in $\mathscr{L}$ and $\{\mathbf{v}_1, \mathbf{v}_2\}$ is a "good" basis for $\mathscr{L}$.

Proposition 7.1.1 (2—Dimensional Gaussian Lattice Reduction Algorithm [14])**.**
Let $\mathscr{L} \subset \mathbb{R}^2$ be spanned by $\{\mathbf{v}_1, \mathbf{v}_2\}$. The following algorithm inputs any lattice basis vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ and returns a reasonably orthogonal basis for the same lattice, $\mathscr{L}$. As a result, the algorithm solves the SVP for 2—dimensions.

Algorithm 4 2—D Gaussian Lattice Reduction

1: If $\|\mathbf{v}_2\| < \|\mathbf{v}_1\|$, swap $\mathbf{v}_1, \mathbf{v}_2$
2: Calculate $m = \left\lfloor \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|^2} \right\rceil$
3: If $m = 0$
4: $\quad \mathbf{v}_1, \mathbf{v}_2$
5: $\mathbf{v}_2 = \mathbf{v}_2 - m\mathbf{v}_1$

Example 7.1.1. *We will walk through an example of this reduction by hand.*
Suppose we have a 2—D lattice spanned by the vectors $\mathbf{v}_1 = \begin{bmatrix} -5 \\ 7 \end{bmatrix}$ and $\mathbf{v}_2 = \begin{bmatrix} -9 \\ 17 \end{bmatrix}$, as shown in 7.1. We can see that the vectors are not reasonably orthogonal in the geometric representation, but we will verify this with the Hadamard Ratio. Then, we will reduce this bad basis to a good one using the 2—D Lattice Reduction Algorithm.

```
sage: V = Matrix([[-5,-9],[7,17]])
sage: (abs(det(V))/(V[:,0].norm()*V[:,1].norm()))^(1/2)
0.3646304886549454
```

The Hadamard Ratio is far from 1*, so this is a "bad" basis. We will continue with the* 2*−D lattice reduction.*

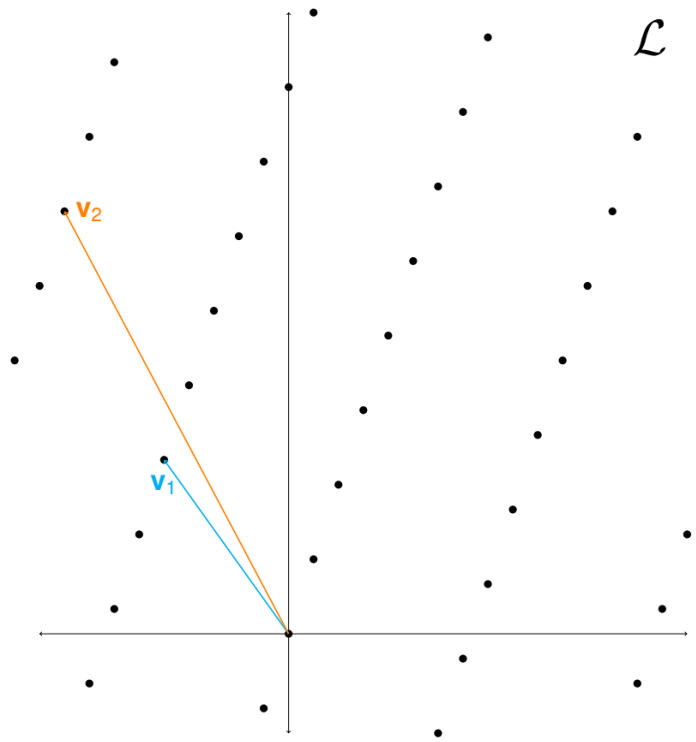


Figure 7.1 $\mathscr{L} = \text{span}\{\mathbf{v}_1, \mathbf{v}_2\}$

We check the Euclidean norms of each to find that $\|\mathbf{v}_1\| \approx 8.6$ *and* $\|\mathbf{v}_2\| \approx 19.2$*. We see that* $\|\mathbf{v}_1\| < \|\mathbf{v}_2\|$*, so we continue to find* m *and replace* $\mathbf{v}_2$ *with* $\mathbf{v}_2 - m\mathbf{v}_1$*.*

```
sage: v1 = vector([-5,7])
sage: v2 = vector([-9,17])
sage: v1.norm().n()
8.60232526704263

sage: v2.norm.n()
19.2353840616713

sage: m = round((v1.dot_product(v2))/(v1.norm())^2)
sage: m
2
```

```
sage: v2 = v2 - m*v1
sage: v2
(1, 3)
```

We now have that $\mathbf{v}_1 = \begin{bmatrix} -5 \\ 7 \end{bmatrix}$ *and* $\mathbf{v}_2 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$, *as shown in Figure 7.2.*

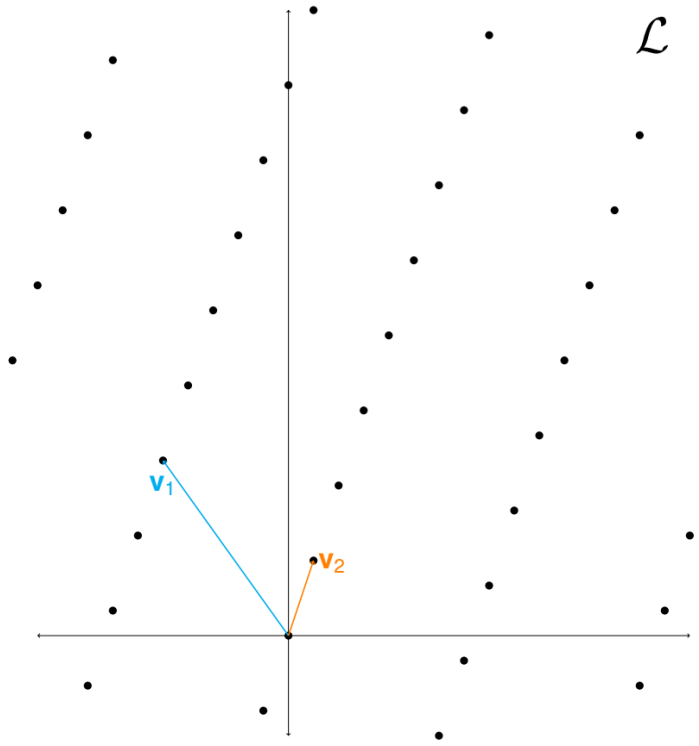


Figure 7.2 The same lattice $\mathscr{L}$ with old $\mathbf{v}_1$ and new $\mathbf{v}_2$.

We again check the norms of $\mathbf{v}_1$ *and* $\mathbf{v}_2$. *We still have that* $\|\mathbf{v}_1\| \approx 8.6$, *and we find that* $\|\mathbf{v}_2\| \approx 3.2$. *Since* $\|\mathbf{v}_2\| < \|\mathbf{v}_1\|$, *we swap* $\mathbf{v}_1$ *and* $\mathbf{v}_2$.

```
sage: v1.norm().n()
8.60232526704263

sage: v2.norm().n()
3.16227766016838
```

```
sage: v1 = vector([1,3])
sage: v2 = vector([-5,7])
```

Now we have that $\mathbf{v}_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ *and* $\mathbf{v}_2 = \begin{bmatrix} -5 \\ 7 \end{bmatrix}$, *as shown in Figure 7.3.*

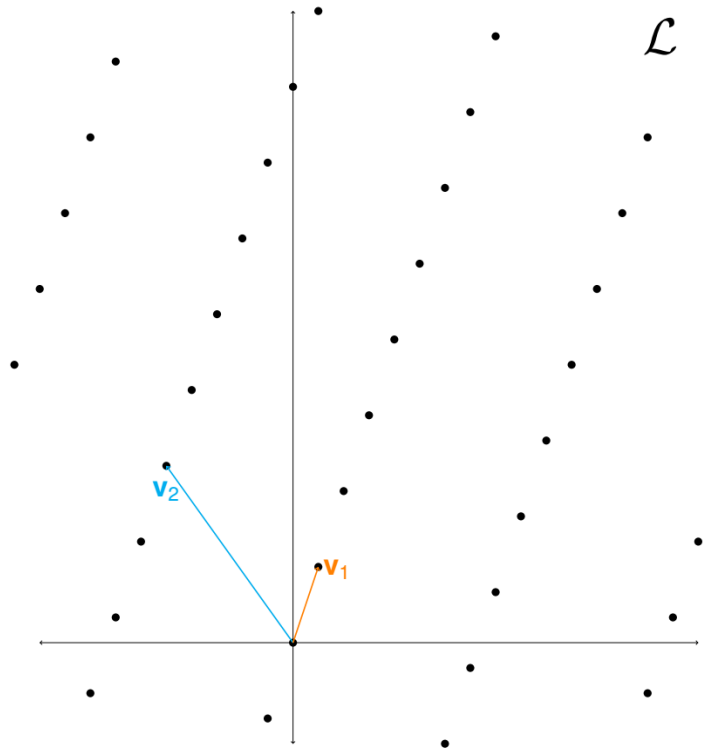


Figure 7.3 $\mathscr{L}$ with swapped $\mathbf{v}_1$ and $\mathbf{v}_2$

We proceed to calculate m, *and we see that* $m = 2$. *So, we replace* $\mathbf{v}_2$ *with* $\mathbf{v}_1 - 2\mathbf{v}_2$.

```
sage: m = round((v1.dot_product(v2))/(v1.norm())^2)
sage: m
2

sage: v2 = v2 - m*v1
sage: v2
(-7, 1)
```

Now, we have that $\mathbf{v}_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ *and* $\mathbf{v}_2 = \begin{bmatrix} -7 \\ 1 \end{bmatrix}$, *as shown in 7.4.*

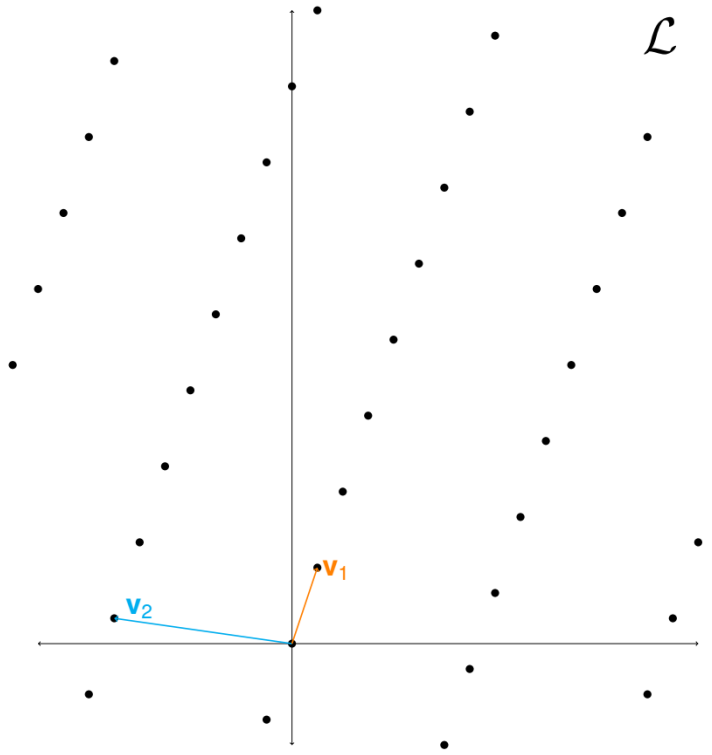


Figure 7.4 $\mathscr{L}$ with good basis $\mathbf{v}_1$ and $\mathbf{v}_2$

We check $\|\mathbf{v}_1\|$ *and* $\|\mathbf{v}_2\|$. *We still have that* $\|\mathbf{v}_1\| \approx 3.2$, *and now we have that* $\|\mathbf{v}_2\| \approx 7.1$. *Since* $\|\mathbf{v}_1\| < \|\mathbf{v}_2\|$, *we do not swap. We proceed to calculate* m.

```
sage: v1.norm().n()
3.16227766016838

sage: v2.norm().n()
7.07106781186548

sage: m = round((v1.dot_product(v2))/(v1.norm())^2)
sage: m
0
```

Since $m = 0$, *we are done. Our good basis is then* $\mathbf{v}_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ *and* $\mathbf{v}_2 = \begin{bmatrix} -7 \\ 1 \end{bmatrix}$, *as in Figure 7.4.*

We can verify that this is a good basis with the Hadamard Ratio.

```
sage: V = Matrix([[1,-7],[3,1]])
```

```
sage: (abs(det(V))/(V[:,0].norm()*V[:,1].norm()))^(1/2)
0.9919021676052067
```

This is reasonably close to 1*, so our new basis is a "good" one. Note, also, that this algorithm solves the SVP, because* $\mathbf{v}_1$ *is a shortest vector in* $\mathscr{L}$*.*

In higher dimensions, this extends to the LLL Lattice Reduction Algorithm of Lenstra, Lenstra, and Lovász. [18]

7.2 LLL Lattice Reduction Algorithm

For the next three definitions, we let $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ be a basis for lattice $\mathscr{L}$, and $\mathscr{B}' = \{\mathbf{v}_1^*, \mathbf{v}_2^*, \ldots, \mathbf{v}_n^*\}$ be the associated Gram-Schmidt orthogonal basis. The Gram-Schmidt coefficients are $\mu_{i,j} = \frac{\mathbf{v}_i \cdot \mathbf{v}_j^*}{\|\mathbf{v}_j^*\|^2}$. [16]

Definition 68 (Size Reduced)**.**
$\mathscr{B}$ *is* **size reduced** *if all Gram-Schmidt coefficients* $\mu_{i,j} = \frac{\mathbf{v}_i \cdot \mathbf{v}_j^*}{\|\mathbf{v}_j^*\|^2}$ *satisfy the condition that* $|\mu_{i,j}| < \frac{1}{2}$ *for all* $1 \leq j < i \leq n$*.*

Definition 69 (Lovász Condition)**.**
$\mathscr{B}$ *and* $\mathscr{B}'$ *are said to satisfy the Lovász Condition if* $\|\mathbf{v}_i^*\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right)\|\mathbf{v}_{i-1}^*\|^2$ *for all* $1 < i \leq n$*.*

Definition 70 (LLL reduced)**.**
The basis $\mathscr{B}$ *is* **LLL reduced** *if it satisfies the* **size reduction condition** *and* **Lovász Condition** *described above.*

The LLL Lattice Reduction Algorithm (5) takes as an input basis vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ for $\mathscr{L}$ and outputs an LLL reduced basis for $\mathscr{L}$. We follow [14] for the LLL Algorithm.

Algorithm 5 LLL Lattice Reduction Algorithm

Set $k = 2$
Set $\mathbf{v}_1^* = \mathbf{v}_1$
Loop while $k \leq n$
    Loop down $j = k-1, k-2, \ldots, 2, 1$
        Set $\mathbf{v}_k = \mathbf{v}_k - \lfloor \mu_{kj} \rceil \mathbf{v}_j$
    End j loop
    If $\|\mathbf{v}_k^*\|^2 \geq \left(\frac{3}{4} - \mu_{k,k-1}^2\right) \|\mathbf{v}_{k-1}^*\|^2$
        Set $k = k+1$
    Else
        Swap $\mathbf{v}_{k-1}$ and $\mathbf{v}_k$
        Set $k = \max(k-1, 2)$
    End If
End k loop

Note: At each step in the LLL Algorithm presented in Algorithm 5, $\mathbf{v}_1^*, \mathbf{v}_2^*, \ldots, \mathbf{v}_k^*$ is the orthogonal set of vectors obtained by applying Gram-Schmidt to the current values of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$, and $\mu_{i,j}$ is the associated quantity $\frac{\mathbf{v}_i \cdot \mathbf{v}_j^*}{\|\mathbf{v}_j^*\|^2}$.

Remark 7.2.1. *The LLL algorithm solves the approximate SVP problem within a constant factor. The details of the proof of this can be found in [14]. We will forego the details of the proof and use the Hadamard Ratio to check orthogonality.*

Example 7.2.1 (LLL by Hand)**.**
We illustrate the LLL using a small example that we adapted from [27]. We want to reduce the basis $\left\{\begin{bmatrix} 201 \\ 37 \end{bmatrix}, \begin{bmatrix} 1648 \\ 297 \end{bmatrix}\right\}$ *to an LLL reduced basis. We start by defining*

$$\mathbf{v}_1 = \begin{bmatrix} 201 \\ 37 \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} 1648 \\ 297 \end{bmatrix}.$$

We let $\mathbf{v}_1^* = \mathbf{v}_1$*, and apply Gram-Schmidt to get*

$$\mathbf{v}_2^* = \begin{bmatrix} 1648 \\ 297 \end{bmatrix} - \frac{\begin{bmatrix} 201 \\ 37 \end{bmatrix} \cdot \begin{bmatrix} 1648 \\ 297 \end{bmatrix}}{\begin{bmatrix} 201 \\ 37 \end{bmatrix} \cdot \begin{bmatrix} 201 \\ 37 \end{bmatrix}} \begin{bmatrix} 201 \\ 37 \end{bmatrix} = \begin{bmatrix} 1.133 \\ -6.155 \end{bmatrix}.$$

We now have

$$\mathbf{v}_1 = \begin{bmatrix} 201 \\ 37 \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} 1648 \\ 297 \end{bmatrix}, \qquad \mathbf{v}_1^* = \begin{bmatrix} 201 \\ 37 \end{bmatrix}, \qquad \mathbf{v}_2^* = \begin{bmatrix} 1.133 \\ -6.155 \end{bmatrix}.$$

We will use $\mathbf{v}_1$ to reduce $\mathbf{v}_2$:

$$\begin{aligned} \mathbf{v}_2 &= \begin{bmatrix} 1648 \\ 297 \end{bmatrix} - \left\lfloor \frac{\begin{bmatrix} 1648 \\ 297 \end{bmatrix} \cdot \begin{bmatrix} 201 \\ 37 \end{bmatrix}}{\begin{bmatrix} 201 \\ 37 \end{bmatrix} \cdot \begin{bmatrix} 201 \\ 37 \end{bmatrix}} \right\rceil \begin{bmatrix} 201 \\ 37 \end{bmatrix} \\ &= \begin{bmatrix} 1648 \\ 297 \end{bmatrix} = 8 \begin{bmatrix} 201 \\ 37 \end{bmatrix} \\ &= \begin{bmatrix} 40 \\ 1 \end{bmatrix} \end{aligned}$$

We now have

$$\mathbf{v}_1 = \begin{bmatrix} 201 \\ 37 \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} 40 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_1^* = \begin{bmatrix} 201 \\ 37 \end{bmatrix}, \qquad \mathbf{v}_2^* = \begin{bmatrix} 1.133 \\ -6.155 \end{bmatrix}.$$

Next, we check the Lovász Condition:

$$\|\mathbf{v}_1^*\|^2 = 41770, \qquad \|\mathbf{v}_2^*\|^2 = 39.16, \qquad \mu_{2,1} = \frac{\begin{bmatrix} 40 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 201 \\ 37 \end{bmatrix}}{\begin{bmatrix} 201 \\ 37 \end{bmatrix} \cdot \begin{bmatrix} 201 \\ 37 \end{bmatrix}} \approx 0.193, \qquad \left(\frac{3}{4} - \mu_{2,1}^2\right) \approx 0.713$$

We have that

$$39.16 = \|\mathbf{v}_2^*\|^2 \not\geq \left(\frac{3}{4} - \mu_{2,1}^2\right) \|\mathbf{v}_1^*\|^2 = 29782.01.$$

This means that we should swap the vectors. Now, we have that

$$\mathbf{v}_1 = \begin{bmatrix} 40 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} 201 \\ 37 \end{bmatrix}, \qquad \mathbf{v}_1^* = \begin{bmatrix} 201 \\ 37 \end{bmatrix}, \qquad \mathbf{v}_2^* = \begin{bmatrix} 1.133 \\ -6.155 \end{bmatrix}.$$

Using $\mathbf{v}_1^* = \mathbf{v}_1$, we again apply Gram-Schmidt reduction:

$$\mathbf{v}_2^* = \begin{bmatrix} 201 \\ 37 \end{bmatrix} - \frac{\begin{bmatrix} 201 \\ 37 \end{bmatrix} \cdot \begin{bmatrix} 40 \\ 1 \end{bmatrix}}{\begin{bmatrix} 40 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 40 \\ 1 \end{bmatrix}} \begin{bmatrix} 40 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.799 \\ 31.956 \end{bmatrix}.$$

We now have that

$$\mathbf{v}_1 = \begin{bmatrix} 40 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} 201 \\ 37 \end{bmatrix}, \qquad \mathbf{v}_1^* = \begin{bmatrix} 40 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_2^* = \begin{bmatrix} -0.799 \\ 31.956 \end{bmatrix}.$$

We then use $\mathbf{v}_1$ to reduce $\mathbf{v}_2$:

$$\begin{aligned} \mathbf{v}_2 &= \begin{bmatrix} 201 \\ 37 \end{bmatrix} - \left\lfloor \frac{\begin{bmatrix} 201 \\ 37 \end{bmatrix} \cdot \begin{bmatrix} 40 \\ 1 \end{bmatrix}}{\begin{bmatrix} 40 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 40 \\ 1 \end{bmatrix}} \right\rfloor \begin{bmatrix} 40 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ 32 \end{bmatrix} \end{aligned}$$

We now have that

$$\mathbf{v}_1 = \begin{bmatrix} 40 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} 1 \\ 32 \end{bmatrix}, \qquad \mathbf{v}_1^* = \begin{bmatrix} 40 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_2^* = \begin{bmatrix} -0.799 \\ 31.956 \end{bmatrix}.$$

Next, we check the Lovász Condition:

$$\|\mathbf{v}_1^*\|^2 = 1601, \qquad \|\mathbf{v}_2^*\|^2 = 1021.76, \qquad \mu_{2,1} = \frac{\begin{bmatrix} 1 \\ 32 \end{bmatrix} \cdot \begin{bmatrix} 40 \\ 1 \end{bmatrix}}{\begin{bmatrix} 40 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 40 \\ 1 \end{bmatrix}} \approx 0.045, \qquad \left(\frac{3}{4} - \mu_{2,1}^2\right) \approx 0.748$$

We have that

$$1021 = \|\mathbf{v}_2^*\|^2 \not\geq \left(\frac{3}{4} - \mu_{2,1}^2\right) \|\mathbf{v}_1^*\|^2 = 1197.548.$$

This means that we should swap again, giving

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 32 \end{bmatrix}, \qquad \mathbf{v}_2 \begin{bmatrix} 40 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_1^* = \begin{bmatrix} 40 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_2^* = \begin{bmatrix} -0.799 \\ 31.956 \end{bmatrix}.$$

We again take $\mathbf{v}_1^* = \mathbf{v}_1$, and use Gram-Schmidt:

$$\mathbf{v}_2^* = \begin{bmatrix} 40 \\ 1 \end{bmatrix} = \frac{\begin{bmatrix} 40 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 32 \end{bmatrix}}{\begin{bmatrix} 1 \\ 32 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 32 \end{bmatrix}} \begin{bmatrix} 1 \\ 32 \end{bmatrix} = \begin{bmatrix} 39.93 \\ -1.25 \end{bmatrix}.$$

Keeping track of our vectors, we have

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 32 \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} 40 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_1^* = \begin{bmatrix} 1 \\ 32 \end{bmatrix}, \qquad \mathbf{v}_2^* = \begin{bmatrix} 39.93 \\ -1.25 \end{bmatrix}.$$

Then, we use $\mathbf{v}_1$ to reduce $\mathbf{v}_2$:

$$\mathbf{v}_2 = \begin{bmatrix} 40 \\ 1 \end{bmatrix} - \left\lfloor \frac{\begin{bmatrix} 40 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 32 \end{bmatrix}}{\begin{bmatrix} 1 \\ 32 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 32 \end{bmatrix}} \right\rceil \begin{bmatrix} 1 \\ 32 \end{bmatrix} = \begin{bmatrix} 40 \\ 1 \end{bmatrix}.$$

This means that $\mathbf{v}_2$ remains $\begin{bmatrix} 40 \\ 1 \end{bmatrix}$.

Next, we check the Lovász Condition again:

$$\|\mathbf{v}_1^*\|^2 = 1025, \qquad \|\mathbf{v}_2^*\|^2 = 1595.94, \qquad \mu_{2,1} = \frac{\begin{bmatrix} 40 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 32 \end{bmatrix}}{\begin{bmatrix} 1 \\ 32 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 32 \end{bmatrix}} \approx 0.070, \qquad \left(\frac{3}{4} - \mu_{2,1}^2\right) \approx 0.745$$

This gives us that

$$1595.94 = \|\mathbf{v}_2^*\|^2 \geq \left(\frac{3}{4} - \mu_{2,1}^2\right) \|\mathbf{v}_1^*\|^2 = 763.625.$$

So, we can now move on to reducing the next vector. However, there are only two vectors here. So we are done. We have that our reasonably orthogonal basis vectors are $\mathbf{v}_1 = \begin{bmatrix} 1 \\ 32 \end{bmatrix}$ and $\mathbf{v}_2 = \begin{bmatrix} 40 \\ 1 \end{bmatrix}$. We can check that these are reasonably orthogonal by computing the Hadamard ratio to be

$$\begin{aligned}\mathscr{H}(\mathbf{v}_1,\mathbf{v}_2) &= \left(\frac{|(1)(1)-(32)(40)|}{\sqrt{1025}\sqrt{1601}}\right)^{1/2} \\ &= \left(\frac{1279}{\sqrt{1,641,025}}\right)^{1/2} \\ &\approx 0.9992093124\end{aligned}$$

This is very close to 1, so our vectors are, in fact, reasonably orthogonal. We have already checked the Lovász and size conditions, so this is an LLL reduced basis.

Example 7.2.2 (LLL with Sage)**.**
Suppose we wanted to reduce the basis from our previous example in Sage. We can simply use the LLL function in Sage. We just need to remember that we use column vectors, and Sage uses row vectors. So, we will need to use the transpose of the matrix and then find the transpose of the LLL reduced matrix to get our reduced basis.

```
sage: V = Matrix([[201,1648],[37,297]])
sage: V.transpose().LLL().transpose()
[ 1 40]
[32  1]
```

7.3 Exercises

1. Let $\mathscr{B} = \left\{ \begin{bmatrix} 10 \\ 9 \end{bmatrix}, \begin{bmatrix} 17 \\ 11 \end{bmatrix} \right\}$ be a basis for lattice $\mathscr{L}$.
   (a) Verify that $\mathscr{B}$ is a "bad" basis.
   (b) Use the 2—dimensional Gaussian Lattice Reduction Algorithm to reduce $\mathscr{B}$ to a "good" basis by hand.
   (c) Verify that your new basis is "good."
2. Let $\mathscr{B} = \left\{ \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 7 \\ 9 \end{bmatrix} \right\}$ be a basis for lattice $\mathscr{L}$.
   (a) Verify that $\mathscr{B}$ is a "bad" basis.
   (b) Use the LLL Lattice Reduction Algorithm to reduce $\mathscr{B}$ to a "good" basis by hand.
   (c) Verify that your new basis is "good."

7.4 Computer Exercises

1. Write Python or Sage code that executes the 2—dimensional Gaussian Lattice Reduction Algorithm. It should take any 2—D basis as input, and it should output a "good" basis.
2. Verify your answer from Problem 1a of the non-computer exercises with your code.
3. Write a Python or Sage code that checks the Lovász condition.
4. Write a Python or Sage code that checks the size condition.
5. Write a Python or Sage code that uses code from the previous two problems to execute the LLL Latice Reduction Algorithm.
6. Check your answer from Problem 2 of the non-computer exercises with your code.
7. Let $\mathscr{B} = \left\{ \begin{bmatrix} 1 \\ 9 \\ 4 \end{bmatrix}, \begin{bmatrix} 7 \\ 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 13 \\ 5 \end{bmatrix} \right\}$.
   (a) Verify that $\mathscr{B}$ is a basis for a 3—dimensional lattice $\mathscr{L}$.
   (b) Find an LLL reduced basis for $\mathscr{L}$.

(c) What is the shortest basis vector?

(d) Use the LLL basis to find the closest lattice vector to $\mathbf{w} = \begin{bmatrix} -1 \\ -10 \\ 5 \end{bmatrix}$.

CHAPTER

8

THE LLL AND LATTICE-BASED CRYPTOSYSTEMS

Though not originally developed for the purpose of cryptography, the LLL Lattice Reduction Algorithm presented in Algorithm 5 has many cryptographic applications. In small dimensions, the LLL can reduce bad bases to good ones and therefore pose a threat to the lattice-based cryptosystems that we have studied to this point. We will show its importance via examples of attacks on the Knapsack, GGH, Congruential Public Key, and NTRU Cryptosystems. We follow [14] for the basis of the constructions.

8.1 LLL on the Knapsack

Say that the message Eve intercepts is $\mathbf{M} = (m_1, m_2, \ldots, m_n)$ and the sum is S. She can form the matrix

$$A_{\mathbf{M},S} = \begin{bmatrix} 2 & 0 & 0 & \dots & 0 & 1 \\ 0 & 2 & 0 & \dots & 0 & 1 \\ 0 & 0 & 2 & \dots & 0 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2 & 1 \\ m_1 & m_2 & m_3 & \dots & m_n & S \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \dots & \mathbf{v}_n & \mathbf{v}_{n+1} \end{bmatrix},$$

and look at all of the integer linear combinations of $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n, \mathbf{v}_{n+1}$. As we know, this is simply the lattice

$$\mathscr{L} = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_n\mathbf{v}_n + a_{n+1}\mathbf{v}_{n+1} : a_1, a_2, \dots, a_n, a_{n+1} \in \mathbb{Z}\}.$$

We can find the LLL-reduced basis for $\mathscr{L}$. We can then use the first LLL reduced basis vector to find the solution to the Subset-Sum problem. We will see this through an example.

Example 8.1.1 (LLL on the Knapsack)**.**
From our Example 4.3.3, Eve intercepts the message
$\mathbf{M} = (492, 305, 177, 723, 767, 545, 101)$ *and knows the sum* $S = 2527$. *She needs to solve the Subset-Sum problem. She sets up the matrix*

$$A_{\mathbf{M},S} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 492 & 305 & 177 & 723 & 767 & 545 & 101 & 2527 \end{bmatrix},$$

where the columns of $A_{\mathbf{M},S}$ *form a basis for* $\mathscr{L}$. *We LLL-reduce the basis using SageMath:*

```
sage: A = Matrix([[2,0,0,0,0,0,0,1],[0,2,0,0,0,0,0,1],
          [0,0,2,0,0,0,0,1],[0,0,0,2,0,0,0,1],[0,0,0,0,2,0,0,1],
          [0,0,0,0,0,2,0,1],[0,0,0,0,0,0,2,1],
          [492,305,177,723,767,545,101,2527]])
sage: A.transpose().LLL().transpose()
[-1  0 -3  2  0  1 -2 -2]
[ 1  0  1  2  0 -3 -4  2]
```

```
[ 1  2 -1 -2 -4  1 -2 -2]
[-1 -2 -1 -2 -2 -1  0 -2]
[-1  0  1  0 -2  1 -2  0]
[-1  2 -1  0  0 -3  0  4]
[ 1  0 -1  2 -2 -1  2  0]
[ 0 -1 -3 -2 -1  3 -1  3]
```

We extract the first column of the resulting matrix, and write it as a linear combination of the original basis vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n, \mathbf{v}_{n+1}$. *That is, we are solving the system given by*

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 492 & 305 & 177 & 723 & 767 & 545 & 101 & 2527 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 0 \end{bmatrix}$$

We solve this in SageMath

```
sage: b = vector([-1,1,1,-1,-1,-1,1,0])
sage: A\b
(-1, 0, 0, -1, -1, -1, 0, 1)
```

This gives $\mathbf{x} = \begin{bmatrix} -1 & 0 & 0 & -1 & -1 & -1 & 0 & 1 \end{bmatrix}^T$. *Note that* $(-1)(492)+(0)(305)+(0)(177)+(-1)(723)+(-1)(767)+(-1)(545)+(0)(101)+(1)(2527) = 0$, *so we have solved the Subset-Sum problem.*

8.2 LLL on the GGH

The security of the GGH relies on the idea that solving the CVP is hard. In small dimensions, Eve can intercept the ciphertext and reduce the matrix of the bad basis to a good one using the LLL. This allows her to solve the CVP and recover the plaintext. We will work through an example, following the construction in [14].

Example 8.2.1 (LLL on the GGH)**.**

Refer to Example 6.1.1, and recall that the private basis for $\mathscr{L}$ *is* $\mathscr{B} = \left\{ \begin{bmatrix} 7 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right\}$ *and*

the public basis is $\mathscr{B}' = \left\{\begin{bmatrix}17\\7\end{bmatrix}, \begin{bmatrix}26\\12\end{bmatrix}\right\}$. *Bob sends Alice the ciphertext message* $\mathbf{m}' = \begin{bmatrix}328\\133\end{bmatrix}$. *Suppose that Eve intercepts this. She only knows* $\mathscr{B}'$. *As we showed in Example 6.1.1, Eve cannot solve the CVP and recover the plaintext with this bad basis. Let* W *be the matrix with column vectors* $\mathbf{v}_1', \mathbf{v}_2'$. *She first computes the LLL reduced matrix of* W. *Note that, since we use column vectors, we need to use the transpose of the matrix and of the LLL. She gets the LLL reduced matrix to be* $\begin{bmatrix}1 & -7\\3 & 1\end{bmatrix}$. *She computes the Hadamard Ratio to see that it is, in fact, a good basis.*

```
sage: W = Matrix([[17,26],[7,512]])
sage: mprime = Matrix([[328],[133]])
sage: L = (W.transpose()).LLL().transpose()
sage: L
[ 1 -7]
[ 3  1]

sage: (abs(det(L))/(L[:,0].norm()*L[:,1].norm()))^(1/2)
0.9919021676052067
```

Eve then solves the CVP for $\mathbf{m}'$ using the LLL reduced basis. She gets $\mathbf{m} = \begin{bmatrix}330\\132\end{bmatrix}$, and she uses this to retrieve $\mathbf{p} = (24, -3)$, the plaintext message.

```
sage: (L.augment(mprime)).rref().n()
[ 1.00000000000000 0.000000000000000  57.2272727272727]
[0.000000000000000  1.00000000000000 -38.6818181818182]

sage: m = 57*L[:,0] - 39*L[:,1]
sage: m
[330]
[132]

sage: (W.augment(m)).rref()
[ 1  0 24]
[ 0  1 -3]
```

8.3 LLL on the Congruential Public Key Cryptosystem

We again follow the construction in [14]. Recall from Section 6.2.1 that Alice has private key (f,g), both small integers, and public key (q,h), where $h \equiv f^{-1}g(\mod q)$. So, if Eve intercepts the message during transit, she knows (q,h) and needs to recover (f,g).

Since $h \equiv f^{-1}g(\mod q)$, this amounts to Eve being able to find (F,G) such that $Fh \equiv G(\mod q)$. Using properties of modular arithmetic, we see that this is the same as finding (F,G) such that $Fh = G + qR$. We can reframe this once more to the following: Eve needs to find (F,G) such that

$$F\begin{bmatrix}1\\h\end{bmatrix} - R\begin{bmatrix}0\\q\end{bmatrix} = \begin{bmatrix}F\\G\end{bmatrix},$$

where F and R are unknown integers, $\begin{bmatrix}1\\h\end{bmatrix}$ and $\begin{bmatrix}0\\q\end{bmatrix}$ are known vectors, and $\begin{bmatrix}F\\G\end{bmatrix}$ is an unknown short vector. If we let $\mathbf{v}_1 = \begin{bmatrix}1\\h\end{bmatrix}$ and $\mathbf{v}_2 = \begin{bmatrix}0\\q\end{bmatrix}$, this can be viewed as a lattice problem. That is, Eve is looking for short nonzero vector $\mathbf{w} = \begin{bmatrix}F\\G\end{bmatrix} \in \mathscr{L} = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 : a_1, a_2 \in \mathbb{Z}\}$.

Example 8.3.1 (LLL on the Congruential Public Key Cryptosystem)**.**
In example 6.2.1, Alice published $(q,h) = (100,58)$ *as her public key and kept* $(f,g) = (7,6)$ *as her private key.*

Eve needs to then find F *and* G *such that*

$$F\begin{bmatrix}1\\58\end{bmatrix} - R\begin{bmatrix}0\\100\end{bmatrix} = \begin{bmatrix}F\\G\end{bmatrix}.$$

In other words, she wants a short vector in $\mathscr{L} = \left\{a_1\begin{bmatrix}1\\58\end{bmatrix} + a_2\begin{bmatrix}0\\100\end{bmatrix} : a_1, a_2 \in \mathbb{Z}\right\}$.

She can perform the $2-D$ *Lattice Reduction Algorithm in 4 to find a short vector that will likely serve as a decryption key. The* $2-D$ *Lattice Reduction Algorithm outputs short vector* $\begin{bmatrix}-7\\-6\end{bmatrix}$*, which is exactly* $-1 \cdot \begin{bmatrix}7\\6\end{bmatrix}$*. So, Eve has recovered the private key and can decrypt the ciphertext message. We could also use the LLL Algorithm in 5 to find this shortest vector.*

```
sage: L = Matrix([[1,0],[58,100]])
sage: L.transpose().LLL().transpose()
[ -7   5]
[ -6 -10]
```

We take the first column of the resulting matrix as $\mathbf{v}_1$*, the shortest vector.*

8.4 LLL on the NTRU

We illustrate the LLL attack on the NTRU with an example, continuing our example in 6.2.7. The NTRU matrix is

$$M_{\mathbf{h}} = \left[\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 21 & 20 & 40 \\ 0 & 1 & 0 & 0 & 0 & 40 & 0 & 1 & 21 & 20 \\ 0 & 0 & 1 & 0 & 0 & 20 & 40 & 0 & 1 & 21 \\ 0 & 0 & 0 & 1 & 0 & 21 & 20 & 40 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 21 & 20 & 40 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 41 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 41 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 41 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 41 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 41 \end{array}\right]$$

On Sage, we find the LLL reduced matrix of $M_{\mathbf{h}}$:

```
sage: M.LLL()
[-1 -1 -1 -1 -1  0  0  0  0  0]
[ 1 -1 -1  1 -1  1  1 -1 -1  0]
[-1  1 -1 -1  1  0  1  1 -1 -1]
[ 1 -1  1 -1 -1 -1  0  1  1 -1]
[-1 -1  1 -1  1  1 -1 -1  0  1]
[-3  0  8  0 -3 -7  8  0 -8  7]
[ 4 -7 -7  4  7 -1 -8  8  1  0]
[ 3  3  0 -8  0 -7  7 -8  0  8]
[ 4  7  4 -7 -7  1  0 -1 -8  8]
[ 0 -3 -3  3  5 11  4 12 11  3]
```

The LLL gives us short vectors. We'll skip the first row, since it zeroed out half of the entries. The next few rows of the LLL reduced matrix have the same length. We can see that the third row of our our LLL reduced matrix gives us the coefficients of $f(x)$ and $g(x)$, up to rotation. Split the row down the middle, and the left side gives coefficients of $f(x)$ while the right gives coefficients of $g(x)$. Here, we get

$$f(x) = -1 + x - x^2 - x^3 + x^4 \qquad \text{and} \qquad g(x) = x + x^2 - x^3 - x^4.$$

Note that the LLL reduction gave us the negative of both of our original $f(x)$ and $g(x)$ functions. Any rotation of $f(x)$ and $g(x)$ will work as the private keys.

8.5 Computer Exercises

1. Suppose Eve knows the public key $M = (951, 668, 327, 1213, 575, 308, 544)$ and intercepts the integer message $S = 3283$ for the Knapsack Cryptosytem. Use the LLL Lattice Reduction Algorithm to solve the Subset-Sum Problem.

2. Alice and Bob are exchanging messages with the GGH Cryptosytem. Alice's public basis is $\mathscr{B}' = \left\{ \begin{bmatrix} 10 \\ 14 \end{bmatrix}, \begin{bmatrix} 21 \\ 34 \end{bmatrix} \right\}$. Eve intercepts the ciphertext $\mathbf{m}' = \begin{bmatrix} -28 \\ -69 \end{bmatrix}$. Use the LLL Lattice Reduction Algorithm to solve the CVP. Use your result to recover the plaintext message, P.

3. Alice and Bob are again using the GGH Cryptosystem to communicate. This time, Alice has public basis $\mathscr{B}' = \left\{ \begin{bmatrix} 62 \\ -72 \\ 142 \end{bmatrix}, \begin{bmatrix} 62 \\ -31 \\ 84 \end{bmatrix}, \begin{bmatrix} 111 \\ -30 \\ 115 \end{bmatrix} \right\}$. Eve intercepts the ciphertext message $\mathbf{m}' = \begin{bmatrix} 5605 \\ -3476 \\ 8560 \end{bmatrix}$. Use the LLL Lattice Reduction Algorithm to find a good basis. Use the good basis to recover $\mathbf{m}$ and then the plaintext message, P.

4. In the Congruential Public Key Cryptosystem, Eve intercepts a ciphertext message $c = 982$. She knows Alice's public keys $(q, h) = (1023, 659)$. Eve wants to recover the plaintext message.

   (a) Help Eve turn this into a lattice problem, and use the 2—dimensional Gaussian Lattice Reduction Algorithm to solve the SVP and find the private key.

   (b) Use the private key to figure out the plaintext message that Bob wanted to send to Alice.

5. Alice and Bob are exchanging messages with the Congruential Public Key Cryptosystem. Alice published the key $(q, h) = (45237, 14249)$. Bob sent the ciphertext message $c = 26941$ to Alice, and Eve intercepted it.

   (a) Turn Eve's problem into a lattice problem. Use the LLL Lattice Reduction Algorithm to solve the SVP and find the private key.

   (b) Use the private key to decrypt the ciphertext.

6. Refer to Example 6.2.5. Use the public key, $h(x)$, to write the NTRU Matrix. Then, use the LLL Lattice Reduction Algorithm to recover $f(x)$ and $g(x)$. Help Eve recover the plaintext message.

7. The NTRU Public Key Cryptosystem that Alice and Bob are using has public parameters $(n, p, q, d) = (11, 3, 53, 2)$. Alice has public key

$$h(x) = 29x^{10} + 43x^9 + 12x^8 + 18x^7 + 27x^5 + 13x^4 + 47x^3 + 44x^2 + 13x + 19.$$

   (a) Write the NTRU matrix using $h(x)$ and q.

   (b) Use the LLL Lattice Reduction Algorithm to find the LLL reduced form of your matrix from part (a).

   (c) What are Alice's secret keys, $f(x)$ and $g(x)$?

   (d) Eve intercepts the ciphertext message

$$e(x) = 44x^{10} + 44x^9 + 11x^8 + 47x^7 + 13x^6 + 42x^5 + 8x^3 + 41x^2 + 31x + 52.$$

   Decrypt this ciphertext to recover the plaintext message.

CHAPTER

9

SIGNATURE SCHEMES

Signing a paper document allows for the recipient of the message to be certain that the document came from the signer. In the computer-based world, things are a little tougher to sign. A digital signature helps verify that the message actually came from the authentic document signer. It binds the person to that digital document.

In this chapter, Samantha is the signer and Victor is the verifier. The components of a Digital Signature Scheme are as follows:

1. Key Generation Algorithm (completed by Samantha)
2. Signing Algorithm (a private algorithm for Samantha to securely sign a message)
3. Verification Algorithm (a public algorithm for anyone, particularly Victor, to verify the signature)

At first glance, a cryptosystem and a digital signature scheme may look very similar. But, there is a significant difference. Think of a cryptosystem as an after hours bank depository. Anyone can deposit money into the narrow deposit box, but only the bank manager has the key to open the box and retrieve the money. The narrow deposit box is like the public encryption key, while the key to unlock the box is like the private decryption key. Think of a digital signature scheme as a signet ring. Only the person wearing the ring can make

the impression in the wax, while anyone can verify that the wearer of the ring made the impression. The ring itself is like the private signing algorithm, and the verification of the wax impression is like the public verification algorithm.

We will show a signature scheme for each of the major cryptosystems that we have discussed: RSA, GGH, and NTRU. The RSA one is well-known, the GGH follows [14], and the NTRU scheme is new, to our knowledge.

9.1 RSA Signature Scheme

The RSA Signature Scheme is well known and appears many places. See, for example, [14].

Key Creation

Samantha chooses secret primes p and q and computes $n = pq$. She computes $\phi(n) = (p-1)(q-1)$ and chooses verification exponent e with $\gcd(e, \phi(n)) = 1$.

Signing

Samantha wants to sign document D, where $1 < D < n$. She computes exponent $d \equiv e^{-1} \pmod{\phi(n)}$. She signs D by computing $S \equiv D^d \pmod{n}$. She sends her signature S to Victor to verify.

Verification

Victor receives S and computes $D' \equiv S^e \pmod{n}$. He verifies that $D' = D$.

Example 9.1.1 (RSA Signature Scheme)**.**

Key Creation

Samantha picks primes $p = 17$ *and* $q = 37$ *and computes* $n = pq = 629$.

```
sage: p = 17
sage: q = 37
sage: n = p*q
sage: n
629
```

Samantha computes $\phi(n) = (p-1)(q-1) = 576$. *She then chooses a verification exponent* $e = 233$. *She checks to be certain that* $\gcd(e, \phi(n)) = 1$.

```
sage: phi = (p-1)*(q-1)
sage: phi
576
```

```
sage: e = 233
sage: gcd(e, phi)
1
```

Signing

Samantha wants to sign document $D = 543$. *She computes her signing exponent* $d \equiv e^{-1} (\mod \phi(n)) = 89$.

```
sage: D = 543
sage: d = mod(e^(-1), phi)
sage: d
89
```

Lastly, Samantha computes her signature $d \equiv D^d (\mod n) = 373$ *to send to Victor.*

```
sage: S = mod(D^d,n)
sage: S
373
```

Verification

Victor receives $S = 343$ *and knows the verification exponent* $e = 233$. *He computes* $D' \equiv S^e (\mod n) = 543$, *which is exactly* D.

```
sage: mod(S^e,n)
543
```

9.2 GGH Signature Scheme

We now present the GGH Signature Scheme, as described in [14].

Key Creation

Samantha chooses a good basis $\mathscr{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ and creates a bad basis $\mathscr{B}' = \{\mathbf{v}'_1, \mathbf{v}'_2, \ldots, \mathbf{v}'_n\}$ for lattice $\mathscr{L}$. She creates the bad basis in the same way she did for the GGH cryptosystem - by multiplying the matrix of $\mathscr{B}$ by a unimodular matrix. She publishes the public key $\mathscr{B}'$.

Signing

Samantha has document $\mathbf{d}$ to sign. She uses Babai's Algorithm in 3 with the good basis $\mathscr{B}$

to compute a vector $\mathbf{s}$ that is close to $\mathbf{d}$. She writes $\mathbf{s} = a_1\mathbf{v}'_1 + a_2\mathbf{v}'_2 + \cdots + a_n\mathbf{v}'_n$ and publishes her signature $(a_1, a_2, \ldots, a_n)$.

Verification

Victor receives Samantha's signature and computes $\mathbf{s} = a_1\mathbf{v}'_1 + a_2\mathbf{v}'_2 + \cdots + a_n\mathbf{v}'_n$. He verifies that $\mathbf{s}$ is sufficiently close to $\mathbf{d}$.

Example 9.2.1 (GGH Signature Scheme)**.**

Key Creation

Samantha chooses good basis $\mathscr{B} = \left\{\begin{bmatrix} 7 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}\right\}$. *She multiplies the matrix of* $\mathscr{B}$ *by unimodular matrix* $\begin{bmatrix} -1 & -2 \\ 2 & 5 \end{bmatrix}$ *to obtain her bad basis* $\mathscr{B}' = \left\{\begin{bmatrix} -5 \\ 7 \end{bmatrix}, \begin{bmatrix} -9 \\ 17 \end{bmatrix}\right\}$. *She then publishes* $\mathscr{B}'$ *and keeps* $\mathscr{B}$ *a secret.*

```
sage: V = Matrix([[7,1],[-1,3]])
sage: U = Matrix([[-1,-2],[2,5]])
sage: W = V*U
sage: W
[-5 -9]
[ 7 17]
```

Signing

Samantha wants to sign document $\mathbf{d} = \begin{bmatrix} 8 \\ -3 \end{bmatrix}$. *She uses Babai's Algorithm to solve the CVP with* $\mathscr{B}$ *for* $\mathbf{d}$ *to get* $\mathbf{s} = \begin{bmatrix} 6 \\ -4 \end{bmatrix}$.

```
sage: d = vector([8,-3])
sage: c = V\d
sage: c
(27/22, -13/22)

sage: for x in c:
          a = round(x)
          print(a)
1
-1
```

```
sage: s = 1 * V[:,0] + -1 * V[:,1]
sage: s
[ 6]
[-4]
```

She then writes $\mathbf{s}$ *as a linear combination of her bad basis to get the coordinates of her signature* $(-3,1)$.

```
sage: W\s
[-3]
[1]
```

Verification

Victor receives Samantha's signature $(-3,1)$ *and computes* $\mathbf{s} = -3\begin{bmatrix}-5\\7\end{bmatrix} + 1\begin{bmatrix}-9\\17\end{bmatrix} = \begin{bmatrix}6\\-4\end{bmatrix}$.

```
sage: -3*W[:,0] + 1*W[:,1]
[6]
[-4]

sage: s = vector([6,-4])
sage: (s-d).norm()
sqrt(5)
```

He then computes $\|\mathbf{s}-\mathbf{d}\| = \sqrt{5}$*, which is sufficiently short. So, he accepts Samantha's signature on the document.*

9.3 NTRU Signature Scheme

The NTRU Signature Scheme given below is our contribution.

The NTRU Signature Scheme has public parameters (n,p,q,d) as in the cryptosystem, where n and p are prime, $\gcd(n,q)=1$, $\gcd(p,q)=1$, and $4dp+2d<q$.

Key Creation

Samantha picks $f(x)\in\mathcal{T}(d+1,d)$ such that both $F_p(x)=f^{-1}(x)$ in R_p and $F_q(x)=f^{-1}(x)$ in R_q exist. She also picks $g(x)\in\mathcal{T}(d+1,d)$ such that $G_q(x)=g^{-1}(x)$ in R_q exists. She computes $F_p(x)$, $F_q(x)$, and $G_q(x)$. She then computes $h(x)=F_q(x)g(x)$ in R_q and makes

$h(x)$ public. Note that $h^{-1}(x) = f(x)G_q(x)$ in R_q.

Signing

Samantha wants to sign message $m(x) \in \mathcal{T}(d,d)$. She picks $r(x) \in \mathcal{T}(d,d)$ and computes signature $e(x) = p\,r(x)f(x) + h^{-1}(x)m(x)$ in R_q. She sends $e(x)$ and $m(x)$ to Victor.

Verification

Victor receives signature $e(x)$ and message $m(x)$. He computes $b(x) = h(x)e(x)$ in R_q. Note that

$$\begin{aligned} b(x) &= h(x)e(x) \\ &= h(x)(p\,r(x)f(x) + h^{-1}m(x)) \\ &= h(x)p\,r(x)f(x) + h(x)h^{-1}(x)m(x) \\ &= h(x)p\,r(x)f(x) + m(x) \\ &= p\,h(x)r(x)f(x) + m(x) \\ &= p\,F_q(x)g(x)r(x)f(x) + m(x) \\ &= p\,g(x)r(x) + m(x) \end{aligned}$$

Victor centerlifts $b(x)$ to $\hat{b}(x)$ in R and reduces in R_p to get $m(x)$. He accepts that the message came from Samantha.

Example 9.3.1 (NTRU Signature Scheme)**.**
Suppose an NTRU Signature Scheme have public parameters $(n,p,q,d) = (5,3,37,2)$. *Then, we have*

$$R = \frac{\mathbb{Z}[x]}{x^5-1}, \qquad R_p = R_3 = \frac{\mathbb{Z}_3[x]}{x^5-1}, \qquad R_q = R_{37} = \frac{\mathbb{Z}_{37}[x]}{x^5-1}.$$

```
sage: R.<x> = PolynomialRing((GF(3),x)
sage: R3.<x> = R.quotient_ring(x^5-1)
sage: S.<x> = PolynomialRing(GF(37),x)
sage: R37.<x> = S.quotient_ring(x^5-1)
```

Key Creation

Samantha chooses $f(x) = 1 + x - x^2 - x^3 + x^4 \in \mathcal{T}(3,2)$ *and computes* $F_3(x) = 2x^4 + 2x$ *and* $F_{37}(x) = 19x^4 + 19x$.

```
sage: f3 = R3(1+x-x^2-x^3+x^4)
sage: F3 = f3^(-1)
```

```
sage: F3
2*x^4 + 2*x

sage: f37 = R37(1+x-x^2-x^3+x^4)
sage: F37 = f37^(-1)
sage: F37
19*x^4 + 19*x
```

She also chooses $g(x) = 1 - x - x^2 + x^3 + x^4 \in \mathscr{T}(3,2)$ *and computes* $G_{37}(x) = 19x^2 + 19$.

```
sage: g = R37(1-x-x^2+x^3+x^4)
sage: G37 = g^(-1)
sage: G37
19*x^2 + 19
```

Lastly, she computes $h(x) = F_{37}(x)g(x) = x^4$ *and makes* $h(x)$ *public.*

```
sage: h = F37*g
sage: h
x^4
```

Signing

Samantha has message (or document) $m(x) = x - x^2 + x^3 - x^4 \in \mathscr{T}(2,2)$ *and chooses* $r(x) = -1 - x + x^3 + x^4 \in \mathscr{T}(2,2)$. *She computes her signature* $e(x) = 3r(x)f(x) + h^{-1}(x)m(x)$ *in* R_{37}. *She sends* $e(x) = 7x^4 + 11x^3 + x^2 + 25x + 30$ *to Victor as her signature along with message* $m(x)$.

```
sage: m = x-x^2+x^3-x^4
sage: r = -1-x+x^3+x^4
sage: e = 3*r*f37+h^(-1)*m
sage: e
7*x^4 + 11*x^3 + x^2 + 25*x + 30
```

Verification

Victor receives $m(x)$ *and* $e(x)$. *He computes* $b(x) = h(x)e(x) = 30x^4 + 7x^3 + 11x^2 + x + 25$.

```
sage: b = h*e
sage: b
30*x^4 + 7*x^3 + 11*x^2 + x + 25
```

He then centerlifts $b(x)$ to $\hat{b}(x)$ in R and reduces it in R_3 to get $2x^4+x^3+2x^2+x$.

```
sage: bhat = ZZ['x']([coeff.lift_centered() for coeff in b.lift()])
sage: bhat
-7*x^4 + 7*x^3 + 11*x^2 + x - 12

sage: R3(bhat)
2*x^4 + x^3 + 2*x^2 + x
```

Note that this is equivalent to $m(x)=-x^4+x^3-x^2+x$. Victor accepts the signature and message as Samantha's.

9.4 Computer Exercises

1. Suppose Samantha wants to send document $D = 74312$ to Victor, and she want to sign it. She uses the RSA Signature Scheme, which has a public modulus of $n = 3962249$ and a signing exponent of $e = 961$. Verify that the document came from Samantha if she sends signature $S = 1939192$ with her document.

2. Samantha is using he RSA Signature Scheme to sign her document $D = 62290$. She chooses private keys $p = 1907$ and $q = 1223$. She also chooses public signing exponent $e = 3361$.

   (a) What signature, S, should Samantha send to Victor?

   (b) Help Victor verify that the document came from Samantha.

3. Samantha is using the GGH Signature Scheme to sign her document $\mathbf{d} = \begin{bmatrix} 11 \\ 25 \end{bmatrix}$. She publishes a basis for lattice $\mathscr{L}$ to be $\mathscr{B}' = \left\{ \begin{bmatrix} 30 \\ 26 \end{bmatrix}, \begin{bmatrix} 74 \\ 61 \end{bmatrix} \right\}$, and she sends her signature $(15, -6)$ to Victor. Verify that the document came from Samantha.

4. In the GGH Signature Scheme, Samantha uses a private basis of $\mathscr{B} = \left\{ \begin{bmatrix} 4 \\ -7 \end{bmatrix}, \begin{bmatrix} 5 \\ 2 \end{bmatrix} \right\}$ and a public basis of $\mathscr{B}' = \left\{ \begin{bmatrix} -7 \\ -20 \end{bmatrix}, \begin{bmatrix} -13 \\ -31 \end{bmatrix} \right\}$. She wants to sign her document, $\mathbf{d} = \begin{bmatrix} 12 \\ 18 \end{bmatrix}$.

   (a) What should Samantha send as her signature?

   (b) Using your answer for part (a), verify that the document came from Samantha.

5. Samantha is using the GGH Signature Scheme to sign a document and send it to Victor for verification. She picks a good lattice basis of $\mathscr{B} = \left\{ \begin{bmatrix} 9 \\ -8 \\ 7 \end{bmatrix}, \begin{bmatrix} -10 \\ 4 \\ 11 \end{bmatrix}, \begin{bmatrix} 3 \\ 11 \\ 9 \end{bmatrix} \right\}$ and publishes a bad lattice basis of $\mathscr{B}' = \left\{ \begin{bmatrix} 5 \\ 99 \\ 139 \end{bmatrix}, \begin{bmatrix} 25 \\ 50 \\ 97 \end{bmatrix}, \begin{bmatrix} 9 \\ 73 \\ 110 \end{bmatrix} \right\}$. Suppose she wants to sign document $\mathbf{d} = \begin{bmatrix} 17 \\ 33 \\ 6 \end{bmatrix}$.

   (a) What is Samantha's signature?

(b) Help Victor verify that **d** came from Samantha.

6. Samantha is using the NTRU Signature Scheme to sign her message $m(x) = x^5 - x^3 + x^2 + x$. The public parameters for this signature scheme are $(n, p, q, d) = (7, 3, 61, 2)$, and the public key is

$$h(x) = 56x^6 + 46x^5 + 2x^4 + 39x^3 + 60x^2 + 11x + 31.$$

Samantha's signature is

$$e(x) = 55x^6 + 60x^5 + 50x^4 + 27x^3 + 23x^2 + 27x + 4.$$

Help Victor verify that the message came from Samantha.

7. In the NTRU Signature Scheme, the public parameters are $(n, p, q, d) = (7, 3, 41, 2)$. Samantha chooses $f(x) = x^5 + x^4 - x^2 + x - 1$ and $g(x) = -x^6 + x^5 - x^4 + x^2 + 1$.

(a) Find Samantha's public key, $h(x)$.

(b) Suppose that Samantha wants to send message $m(x) = x^6 - x^4 + x^3 - 1$. She chooses $r(x) = -x^5 - x^3 + x^2 + x$. What is Samantha's signature?

(c) Verify that your answers are correct by helping Victor verify that the message came from Samantha.

CHAPTER

10

BLIND SIGNATURE SCHEMES

Suppose you want to vote in an election. You need your ballot signed by a third party before it can be sent to its destination to be counted. However, you do not want the third party signer to know how you voted. So, how do you secure it? You could place your original document in an envelope on carbon copy paper, and seal the envelope. The signer could sign the outside of the envelope without ever opening it and send it back to you. This way, the signer has not seen the document, but she has signed it. When you open the envelope back up, you can see her signature on the carbon copy paper. This is the idea behind a Blind Signature Scheme. In a blind signature scheme, the signer cannot see the document during the process of the signature generation, and the signer cannot match the signed document to the author of the message. We can think of the signer as the election authority, the author of the message as the voter, and the message recipient as the voting center. For the purposes of this chapter, Samantha will be the election authority (signer), Alice will be the voter (author and message sender), and Victor will be the voting center (verifier).

Blind signature schemes involve what is known as a challenge-response problem. Here, Samantha is a third party signer, Alice is the message sender, and Victor is the message recipient and verifier. Samantha sets up the parameters of the scheme and makes some of them public. Alice wants to send a message, but she does not want the document to be revealed to the signer. She blinds, or disguises, the document in some way before sending

it to Samantha. Samantha signs the document and sends it back to Alice. Then, Alice sends the document and the signature to Victor. Victor can verify the document. We will work through the RSA blind signature scheme, as shown in [14] and many other places, and then we will proceed to new schemes that we have developed based on the GGH and NTRU cryptosystems.

10.1 RSA Blind Signature Scheme

The RSA Blind Signature Scheme is well-known. We follow [14].

Key Creation
Samantha sets up the regular RSA parameters $p, q, n, \phi(n), e, d$, and she makes n and e public.

Document Blinding
Alice has message m. She picks r such that $\gcd(r, n) = 1$ and computes $z \equiv m r^e \pmod{n}$. z is sent to Samantha as the blinding factor.

Signing
Samantha computes $y \equiv z^d \equiv m^d r^{ed} \equiv m^d r \pmod{n}$ and sends y back to Alice.

Unblinding and Message Sending
Alice computes $s \equiv r^{-1} y \equiv r^{-1} z^d \equiv r^{-1}(m^d r) \pmod{n} = m^d$, which is sent to Victor, the message recipient, along with m.

Verification
Victor computes $s^e \equiv (m^d)^e \pmod{n} = m$ and compares it to m. He accepts that the message came from Alice with a blind signature from Samantha.

Example 10.1.1 (RSA Blind Signature Scheme)**.**
Key Creation
Samantha chooses the parameters for the RSA Blind Signature Scheme. She chooses $p = 67$ *and* $q = 31$. *She computes* $n = pq = 2077$ *and* $\phi(n) = (p-1)(q-1) = 1980$. *She then chooses* $e = 101$ *and checks that* $\gcd(e, \phi(n)) = 1$. *Lastly, she calculates* $d \equiv e^{-1} \pmod{\phi(n)}$.

```
sage: p = 67
sage: q = 31
```

```
sage: n = pq
sage: n
2077

sage: phi = (p-1)*(q-1)
sage: phi
1980

sage: e = 101
sage: gcd(e, phi)
1

sage: d = mod(e^(-1), phi)
sage: d
941
```

Document Blinding

Alice has message $m = 1990$ *that she wants to send. She chooses* $r = 81$ *and checks that* $\gcd(n, r) = 1$. *She computes* $z \equiv m r^e (\mod (n)) = 1641$ *to send to Samantha as the blinding factor.*

```
sage: m = 1990
sage: r = 81
sage: gcd(n, r)
1

sage: z = mod(m*r^e, n)
sage: z
1641
```

Signing

Samantha receives z *and computes* $y = z^d (\mod n) = 680$.

```
sage: y = mod(z^d, n)
sage: y
680
```

Unblinding and Message Sending

Alice receives y and computes her signature $s \equiv r^{-1} y \pmod{n} = 1111$. She sends her signature and the message to Victor.

```
sage: s = mod(r^(-1)*y, n)
sage: s
1111
```

Verification

Victor receives s and m. He computes $s^e \pmod{n} = 1990$, which is exactly m.

```
sage: mod(s^e, n)
1990
```

Alice's message has been verified.

10.2 GGH Blind Signature Scheme

In this section, we present an original Blind Signature Scheme for the GGH.

Key Creation

Samantha sets up the GGH with good basis $\mathscr{B}$ and bad basis $\mathscr{B}'$. She makes $\mathscr{B}'$ public.

Document Blinding

Alice has document $\mathbf{d}$ to be signed, but she does not want $\mathbf{d}$ to be revealed to Samantha. She picks $\mathbf{u} \in \mathscr{L}$ and computes $\mathbf{d} + \mathbf{u} = \mathbf{m}$, which is sent to Samantha as the blinding factor.

Signing

Samantha finds the closest vector $\mathbf{s} \in \mathscr{L}$ to $\mathbf{m}$ using $\mathscr{B}$. $\mathbf{s}$ is sent to Alice as Samantha's signature.

Unblinding and Message Sending

Alice computes $\mathbf{t} = \mathbf{s} - \mathbf{u}$. $\mathbf{t}$ is the Alice's signature. Note that $\mathbf{t} \in \mathscr{L}$. $\mathbf{t}$ and $\mathbf{d}$ are sent to Victor.

Verification

Victor checks that $||\mathbf{t} - \mathbf{d}|| = ||\mathbf{s} - \mathbf{u} - \mathbf{d}|| = ||\mathbf{s} - \mathbf{m}||$ is small and $\mathbf{t} \in \mathscr{L}$. This checks that $\mathbf{d}$ came from Alice.

Example 10.2.1 (GGH Blind Signature Scheme)**.**

Key Creation

Samantha chooses good basis $\mathscr{B} = \left\{\begin{bmatrix} 7 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}\right\}$ *and computes bad basis* $\left\{\begin{bmatrix} 17 \\ 7 \end{bmatrix}, \begin{bmatrix} 26 \\ 12 \end{bmatrix}\right\}$*. She makes* $\mathscr{B}'$ *public.*

```
sage: V = Matrix([[7,1],[-1,3]])
sage: U = Matrix([[2,3],[3,5]])
sage: W = V*U
sage: W
[17 26]
[ 7 12]
```

Document Blinding

Alice wants to sign document $\mathbf{d} = \begin{bmatrix} -13 \\ 4 \end{bmatrix}$*. She picks* $\mathbf{u} = \begin{bmatrix} -4 \\ -12 \end{bmatrix} \in \mathscr{L}$ *and computes* $\mathbf{m} = \mathbf{d} + \mathbf{u} = \begin{bmatrix} -17 \\ -8 \end{bmatrix}$*.* $\mathbf{m}$ *is sent to Samantha.*

```
sage: d = matrix([[-13],[4]])
sage: u = matrix([[-4],[12]])
sage: m = d + u
sage: m
[-17]
[ -8]
```

Signing

Samantha receives $\mathbf{m}$ *and solves the CVP using* $\mathscr{B}$*. The closest vector,* $\mathbf{s} = \begin{bmatrix} -17 \\ -7 \end{bmatrix}$ *is sent to Alice.*

```
sage: (V.augment(m)).rref().n()
[ 1.00000000000000 0.000000000000000 -1.95454545454545]
[0.000000000000000  1.00000000000000 -3.31818181818182]

sage: s = -2*V[:,0]-3*V[:,1]
sage: s
[-17]
[ -7]
```

Unblinding and Message Sending

Alice receives $\mathbf{s}$ *and computes her signature* $\mathbf{t} = \mathbf{s} - \mathbf{u} = \begin{bmatrix} -13 \\ 5 \end{bmatrix}$. *She sends* $\mathbf{t}$ *and* $\mathbf{d}$ *to Victor.*

```
sage: t = s - u
sage: t
[-13]
[  5]
```

Verification
Victor receives $\mathbf{t}$ *and* $\mathbf{d}$. *He checks that* $\|\mathbf{t}-\mathbf{d}\|$ *is small and that* $\mathbf{t} \in \mathscr{L}$.

```
sage: (t-d).norm()
1.0
```

```
sage: (W.augment(t)).rref().n()
[ 1.00000000000000 0.000000000000000 -13.0000000000000]
[0.000000000000000  1.00000000000000  8.00000000000000]
```

Note that row reducing the matrix $\left[W \;\middle|\; \mathbf{t} \right]$ *gives integer solutions. This means that* $\mathbf{t} \in \mathscr{L}$.

10.3 NTRU Blind Signature Scheme

In this section we present an original Blind Signature Scheme for the NTRU and show that our protocol has the desired properties.

Samantha sets up the typical NTRU protocol by picking (n, p, q, d), where n and p are prime, $\gcd(p, q) = 1$, and $\gcd(n, q) = 1$.

Key Creation
Samantha picks $f(x) \in \mathscr{T}(d+1, d)$ that is invertible in R_q and R_p. She also picks $g(x) \in \mathscr{T}(d+1, d)$ that is invertible in R_q. She computes $F_q(x) = f^{-1}(x)$ in R_q, $F_p(x) = f^{-1}(x)$ in R_p, and $h(x) = F_q(x)g(x)$ in R_q. Note that $h(x)$ has an inverse in R_q. $h(x)$ is made public along with n, p, q, and d.

Document Blinding
Alice wants to send $m(x)$ to Victor, and she wants it to have a blind signature. $m(x)$ should have coefficients m_i with $\frac{-1}{2}p < m_i \leq \frac{1}{2}p$, just as in the cryptosystem. Alice picks $t(x) \in \mathscr{T}(d+1, d)$ with inverse in R_q. Alice computes $s(x) = m(x)t(x)$ and sends it to Samantha

as her blinding factor.

Signing
Samantha picks $r(x) \in \mathscr{T}(d+1,d)$ and computes $e(x) = p\,r(x)f(x) + h^{-1}(x)s(x)$ in R_q. Samantha sends $e(x)$ back to Alice as her signature.

Unblinding and Message Sending
Alice finds $a(x) = h(x)t^{-1}(x)e(x)$ in R_q. Note that

$$\begin{aligned} a(x) &= h(x)t^{-1}(x)e(x) \\ &= h(x)t^{-1}(x)[p\,r(x)f(x) + h^{-1}(x)s(x)] \\ &= h(x)t^{-1}(x)[p\,r(x)f(x) + h^{-1}(x)m(x)t(x)] \\ &= h(x)t^{-1}(x)p\,r(x)f(x) + h(x)t^{-1}(x)h^{-1}(x)m(x)t(x) \\ &= F_q(x)g(x)t^{-1}(x)p\,r(x)f(x) + m(x) \\ &= p\,g(x)t^{-1}(x)r(x) + m(x) \end{aligned}$$

Alice sends $m(x)$ and $a(x)$ to Victor.

Verification
Victor centerlifts $a(x)$ to $\hat{a}(x)$ in R and reduces it in R_p to get back $m(x)$. Note that

$$\begin{aligned} \hat{a}(x) &= p\,g(x)t^{-1}(x)r(x) + m(x) \\ &= m(x). \end{aligned}$$

Example 10.3.1 (NTRU Blind Signature Scheme)**.**

Public Parameters
Samantha sets up an NTRU protocol with $(n,p,q,d) = (5,3,41,2)$*. That is, she has the following public rings:*

$$R = \frac{\mathbb{Z}[x]}{x^5-1} \qquad R_p = R_3 = \frac{\mathbb{Z}_3[x]}{x^5-1} \qquad R_q = R_{41} = \frac{\mathbb{Z}_{41}[x]}{x^5-1}.$$

Key Creation
Samantha chooses $f(x) = -x^4 + x^3 + x^2 - x + 1 \in \mathscr{T}(3,2)$*, and finds* $F_3(x) = f^{-1}(x)$ *in* R_3 *is* $F_3(x) = 2x^3 + 2x^2$ *and that* $F_{41}(x) = f^{-1}(x)$ *in* R_{41} *is* $F_{41}(x) = 21x^3 + 21x^2$*. She also chooses* $g(x) = x^4 + x^3 - x^2 + x - 1 \in \mathscr{T}(3,2)$ *and finds that* $G_{41}(x) = g^{-1}(x)$ *in* R_{41} *is* $G_{41}(x) = 21x^2 + 21x$*.*

She computes $h(x) = F_{41}(x)g(x) = x$. She makes and $h(x)$ public.

```
sage: R.<x> = PolynomialRing(GF(3),x)
sage: R3.<x> = R.quotient_ring(x^5-1)
sage: f3 = R3(-x^4+x^3+x^2-x+1)
sage: F3 = f3^(-1)
sage: F3
2*x^3 + 2*x^2

sage: S.<x> = PolynomialRing(GF(41),x)
sage: R41.<x> = S.quotient_ring(x^5-1)
sage: f41 = R41(-x^4+x^3+x^2-x+1)
sage: F41 = f41^(-1)
sage: F41
21*x^3 + 21*x^2

sage: g = R41(x^4+x^3-x^2+x-1)
sage: G41 = g^(-1)
sage: G41
21*x^2 + 21*x

sage: h = F41*g
sage: h
x
```

Document Blinding

Alice wants to send document $m(x) = x^2 + x - 1$ to Victor. She chooses $t(x) = x^4 - x^3 + x^2 - x + 1 \in \mathscr{T}(3,2)$ and finds that $T_{41}(x) = t^{-1}(x)$ in R_{41} is $T_{41}(x) = 21x + 21$. She then computes $s(x) = m(x)t(x) = 40x^4 + x^3 + 40x^2 + 3x + 40$, and sends $s(x)$ to Samantha as her blinding factor.

```
sage: m = x^2 + x - 1
sage: t = R41(x^4-x^3+x^2-x+1)
sage: T41 = t^(-1)
sage: T41
21*x + 21
```

```
sage: s = m*t
sage: s
40*x^4 + x^3 + 40*x^2 + 3*x + 40
```

Signing

Samantha receives $s(x)$. She chooses $r(x) = -x^4 - x^3 + x^2 + x + 1 \in \mathcal{T}(3,2)$ and finds that $R(x) = r^{-1}(x)$ in R_{41} is $R(x) = 21x^3 + 21$. She computes $e(x) = p\,r(x)f(x) + h^{-1}(x)s(x) = 2x^4 + 2x^3 + 4x^2 + 31x + 6$, and she sends $e(x)$ back to Alice as her signature.

```
sage: r = R41(-x^4-x^3+x^2+x+1)
sage: R = r^(-1)
sage: R
21*x^3 + 21

sage: e = 3*r*f41 + h^(-1)*s
sage: e
2*x^4 + 2*x^3 + 4*x^2 + 31*x + 6
```

Unblinding and Message Sending

Alice receives $e(x)$ and computes $a(x) = h(x)T_{41}(x)e(x) = 3x^4 + 38x^3 + 39x^2 + 4x + 2$. Alice sends $a(x)$ and $m(x)$ to Victor.

```
sage: a = h*T41*e
sage: a
3*x^4 + 38*x^3 + 39*x^2 + 4*x + 2
```

Verification

Victor receives $a(x)$ and $m(x)$. He centerlifts $a(x)$ to $\hat{a}(x)$ in R and reduces it in R_3 to get back $x^2 + x + 2$. He compares this to $m(x) = x^2 + x - 1$. Note that $x^2 + x + 2 \equiv x^2 + x - 1 \pmod 3$, which is $m(x)$.

```
sage: ahat = ZZ['x']([coeff.lift_centered() for coeff in a.lift()])
sage: ahat
3*x^4 - 3*x^3 - 2*x^2 + 4*x + 2

sage: R3(ahat)
x^2 + x + 2
```

10.4 Computer Exercises

1. Suppose that Alice wants to send message $m = 5938542$ to Victor with the RSA Blind Signature Scheme, but she wants to blind it before it is signed by Samantha. Samantha has public keys $(n, e) = (27631523, 3943)$. Alice chooses $r = 1218$. What is Alice's blinding factor, z, that she sends to Samantha?

2. Suppose that Samantha has private keys $(p, q) = (5023, 5501)$ and she received the blinding factor from the previous problem. Calculate d, and use it to find the value of y, Samantha's signature.

3. Samantha, Alice, and Victor are using the RSA Blind Signature scheme. Samantha keeps primes $p = 6421$ and $q = 6323$ private and publishes $n = 40599983$ and $e = 3001$.

   (a) Alice wants to send message $m = 4887586$. She chooses $r = 1011$. Find her blinding factor, z.

   (b) Samantha receives z. What is Samantha's signature, y?

   (c) Alice receives Samantha's signature, y. Find the value of s that Alice will send to Victor along with her value of m.

   (d) Help Victor verify that the message came from Samantha.

4. Use the GGH Blind Signature Scheme to answer this question. Samantha picks good basis $\mathscr{B} = \left\{ \begin{bmatrix} 5 \\ -1 \end{bmatrix}, \begin{bmatrix} 2 \\ 5 \end{bmatrix} \right\}$ and bad basis $\mathscr{B}' = \left\{ \begin{bmatrix} 12 \\ 3 \end{bmatrix}, \begin{bmatrix} 29 \\ 5 \end{bmatrix} \right\}$. Alice wants to send document $\mathbf{d} = \begin{bmatrix} 604 \\ 1964 \end{bmatrix}$. She chooses $\mathbf{u} = \begin{bmatrix} 1224 \\ 36 \end{bmatrix} \in \mathscr{L}$.

   (a) What is $\mathbf{m}$ that Alice sends to Samantha as her blinding factor?

   (b) Find $\mathbf{s}$ for Samantha to send back to Alice.

   (c) Compute $\mathbf{t}$ for Alice to send to Victor, along with $\mathbf{d}$.

   (d) Victor receives $\mathbf{d}$ and $\mathbf{t}$. Verify that the document came from Alice.

5. Samantha sets up the GGH Blind Signature scheme with a good basis of $\mathscr{B} = \left\{ \begin{bmatrix} 2 \\ -4 \\ 6 \end{bmatrix}, \begin{bmatrix} -3 \\ 4 \\ 5 \end{bmatrix}, \begin{bmatrix} 6 \\ 5 \\ -3 \end{bmatrix} \right\}$ and a bad basis of $\mathscr{B}' = \left\{ \begin{bmatrix} 46 \\ 53 \\ 5 \end{bmatrix}, \begin{bmatrix} 36 \\ 26 \\ 10 \end{bmatrix}, \begin{bmatrix} 37 \\ 39 \\ 6 \end{bmatrix} \right\}$.

(a) Alice needs to have document $\mathbf{d} = \begin{bmatrix} 622 \\ 103 \\ 95 \end{bmatrix}$ signed. She chooses lattice vector $\mathbf{u} = \begin{bmatrix} 258 \\ 364 \\ 102 \end{bmatrix}$. Find her blinding factor, $\mathbf{m}$.

(b) Solve the CVP using the good basis to find signature $\mathbf{s}$ for Samantha.

(c) Find $\mathbf{t}$, Alice's document signature to be sent to Victor.

(d) Help Victor verify that the document came from Alice.

6. Samantha sets up the NTRU Blind Signature Scheme with public parameters $(n, p, q, d) = (7, 3, 43, 2)$. She makes $h(x) = 41x^6 + 42x^5 + 2x^4 + 41x^2 + 2x + 2$ public.

(a) Suppose that Alice wants to send message $m(x) = -x^5 + x^3 + x^2 - 1$. She picks $t(x) = x^6 + x^4 - x^3 - x^2 + 1$ to help blind her document. Calculate $s(x)$ to send to Samantha.

(b) Suppose that Samantha's private keys are $f(x) = x^6 - x^5 + x^4 - x^3 + x^2$ and $g(x) = -x^4 + x^3 + x^2 - x + 1$, and suppose that she chooses $r(x) = x^6 + x^4 + x^3 - x - 1$. Compute her signature, $e(x)$.

(c) What is Alice's signature, $a(x)$?

(d) If Victor receives $m(x)$ and $a(x)$, verify that $m(x)$ came from Alice.

CHAPTER

11

ZERO KNOWLEDGE PROOFS

How do you prove to someone that you know information without giving them the information? That is the idea behind a Zero Knowledge Proof. The concept here is that Victor (the verifier) sends a random ciphertext to Peggy (the prover), and Peggy answers with the plaintext of the ciphertext. The following example is adapted from [24].

The door that blocks a path has a secret password. Peggy claims that she knows it. How can she prove to Victor that she knows it without also proving to Eve that she knows it and without also giving away the password? Peggy enters the cave and picks one of the two paths to the door. Victor stays outside so that he does not see which way she goes in. Once Peggy is at the door, Victor enters the cave. He shouts to Peggy to tell her which route she should take out. If she does not know the password to the door, she has to go back the way that she came in. Otherwise, she can do as Victor asks. If she knows the password, she will always be able to do as Victor asks. If she does not know the password, the probability that she comes out the right door each time is 50%. After n times, if she does not know the password, she only has $\left(\frac{1}{2}\right)^n$ chance of doing as Victor asks. After n times, it is shown that she either does not have the password or she has a $1-\left(\frac{1}{2}\right)^n$ chance of having the password.

She has not proven to a third party that she has the password. If Eve observes all that has happened, Eve only knows what Victor asked for and where Peggy appeared. This does not prove that Peggy has the password, because Victor and Peggy could have arranged in

advance a sequence of ways that Victor called out and Peggy then appears. Eve does not know for sure if they cheated. Also, Victor does not learn the password, just that Peggy knows it. No knowledge of Peggy's has been shown to Victor other than that she has the password. This is a zero knowledge proof.

Zero knowledge proofs are a series of challenge and response protocol that are popularly used in identity verification. In a Zero Knowledge Proof, the prover presents the verifier with a transcript that satisfies

1. Completeness (the verifier accepts an honest prover),
2. Soundness (the trasncript implies that the prover knows), and
3. Zero-Knowledge (does not reveal any additional information) [14].

We will show a common Zero Knowledge Proof for the Discrete Log Problem before showing a new Zero Knowledge Proof for the GGH and the NTRU cryptosystems. The Discrete Log Zero Knowledge Proof follows [6], and the GGH and NTRU protocols are our contributions.

11.1 Discrete Logarithm Zero Knowledge Proof

We begin with a discussion of a Discrete Log Zero Knowledge Proof, as discussed in [6]. The Discrete Log Problem is as follows:

Problem 11.1.1 (Discrete Log Problem)**.**
Given a, b, p with p prime, what is x such that $a^x \equiv b (mod\ p)$?

Peggy claims that she knows x and wants to prove this to Victor without revealing x.

Key Creation
Peggy picks r, computes $c \equiv a^r (\text{mod } p)$, and sends c to Victor.

Challenge
Victor asks Peggy for either r or $(x+r)(\text{mod } (p-1))$. No information from the original problem has been revealed by this step.

Response
Peggy sends either r or $(x+r)(\ \text{mod } (p-1))$.

Verification

If Peggy sends r, then Victor finds $a^r \equiv c \pmod{p}$, as should be the case. If Peggy sends $(x+r) \pmod{(p-1)}$, then Victor computes $a^{(x+r) \pmod{(p-1)}} \pmod{p} = a^x a^r \pmod{p} = b\,c \pmod{p}$, and it checks that Peggy knows x.

Remark 11.1.1. *$(x+r)(mod\,(p-1)) = x + r + (\tau(p-1))$, so $a^{(x+r)(mod\,(p-1))}(mod\,p) = a^x a^r a^{\tau(p-1)}(mod\,p) = a^x a^r(mod\,p)$. Also, $a^x a^r(mod\,p) = b\,c \implies a^x(mod\,p) = c$, and x must be known to Peggy.*

Remark 11.1.2. *Why do Peggy and Victor need Peggy to send either r or $x+r$ rather than just using $x+r$?*

If all Peggy needs to do is use $x+r$, then she can instead cheat as follows:

She could pick r' and compute $c' \equiv a^{r'}(a^x)^{-1}(mod\,p)$. She sends c' to Victor. As before, Victor computes $c'b = a^r(a^x)^{-1}a^x(mod\,p) = a^{r'}(mod\,p)$. Remember that r' was sent to Victor as $x+r$, so it looks like $c'b \equiv a^{x+r}(mod\,p)$ as expected, but Peggy does not need to know x in order to do this.

Notice that Victor never finds out x. Eve would never find out x if she was eavesdropping either. This is a zero knowledge proof.

11.2 GGH Zero Knowledge Proof

We will now present an original Zero Knowledge Proof based on the GGH Cryptosystem. Peggy has a good basis $\mathscr{B}$ for lattice $\mathscr{L}$ and wants to convince Victor of this without giving him the basis. She'll do this by solving the CVP.

Key Creation

Peggy picks a good basis $\mathscr{B}$ for $\mathscr{L}$ and computes a bad basis $\mathscr{B}'$ to send to Victor.

Challenge

Victor takes a vector $\mathbf{x}$ and sends the coordinates in terms of $\mathbb{R}^n$ to Peggy.

Response

Peggy uses her good basis to get a closest vector $\mathbf{y}$ to $\mathbf{x}$. She writes $\mathbf{y}$ in terms of the bad basis $\mathscr{B}'$ and sends it to Victor.

Verification

Victor uses the coordinates he receives to find $\mathbf{y}$, and then he computes $\|\mathbf{y}-\mathbf{x}\|$ to see that

the vectors are close. If they are sufficiently close, he accepts Peggy's response as valid. They repeat this a number of times.

Remark 11.2.1. *The GGH Zero Knowledge Proof Protocol satisfies the following:*

1. *Completeness: If* $\mathbf{y}$ *is close to* $\mathbf{x}$*, then Victor will always accept Peggy's* $\mathbf{y}$*. We use the bound that* $\|\mathbf{y}-\mathbf{x}\| \leq \sqrt{n}\det(\mathscr{L})^{1/n}$.
2. *Soundness: If* $\mathbf{y}$ *is not close to* $\mathbf{x}$*, Victor will likely not accept Peggy's* $\mathbf{y}$*. In other words, there is a small probability that Victor will accept all of Peggy's responses.*
3. *Zero Knowledge: Victor never finds out the good basis for* $\mathscr{L}$.

Example 11.2.1 (GGH Zero Knowledge Proof)**.**

Key Creation

Peggy has good basis $\mathscr{B} = \left\{\begin{bmatrix}6\\1\end{bmatrix}, \begin{bmatrix}1\\4\end{bmatrix}\right\}$*. Note that the Hadamard Ratio of this basis is* 0.9576*, as calculated below with Sage.*

```
sage: V = Matrix([[6,1],[1,4]])
sage: v1 = vector([6,1])
sage: v2 = vector([1,4])
sage: numerical_approx((det(V)/(v1.norm()*v2.norm()))^(1/2))
0.957637747926237
```

She forms $V = \begin{bmatrix}6 & 1\\ 1 & 4\end{bmatrix}$ *and chooses unimodular* $U = \begin{bmatrix}2 & -3\\ 1 & -2\end{bmatrix}$ *to compute* $VU = \begin{bmatrix}13 & -20\\ 6 & -11\end{bmatrix}$.

```
sage: U = Matrix([[2,-3],[1,-2]])
sage: V*U
[ 13 -20]
[  6 -11]
```

This gives her bad basis $\mathscr{B}' = \left\{\begin{bmatrix}13\\6\end{bmatrix}, \begin{bmatrix}-20\\-11\end{bmatrix}\right\}$*. She publishes her bad basis.*

Challenge

Victor has the bad basis $\mathscr{B}'$*. He picks vector* $\mathbf{x} = \begin{bmatrix}1\\-5\end{bmatrix}$ *to send to Peggy.*

Response

Peggy solves the CVP for $\mathbf{x}$ *using good basis* $\mathscr{B}$.

She sets up $a_1\begin{bmatrix}6\\1\end{bmatrix}+a_2\begin{bmatrix}1\\4\end{bmatrix}=\begin{bmatrix}1\\-5\end{bmatrix}$ *and solves the system on Sage.*

```
sage: V = Matrix([[6,1],[1,4]])
sage: b = vector([1,-5])
sage: V\b.n()
(0.391304347826087, -1.34782608695652)
```

She rounds to get $c_1=0$ *and* $c_2=-1$*, and computes* $\mathbf{y}=0\begin{bmatrix}6\\1\end{bmatrix}-1\begin{bmatrix}1\\4\end{bmatrix}=\begin{bmatrix}-1\\-4\end{bmatrix}$*. She writes* $\mathbf{y}$ *as coordinates with respect to the bad basis* $\mathscr{B}'$*.*

```
sage: B = Matrix([[13, -20],[6,-11]])
sage: y = vector([-1,-4])
sage: B\v
(3, 2)
```

Peggy sends $\begin{bmatrix}3\\2\end{bmatrix}$ *to Victor to be verified.*

Verification

Victor finds $\mathbf{y}$ *by computing* $\mathbf{y}=3\begin{bmatrix}13\\6\end{bmatrix}+2\begin{bmatrix}-20\\-11\end{bmatrix}=\begin{bmatrix}-1\\-4\end{bmatrix}$*. He then computes* $\|\mathbf{y}-\mathbf{x}\|=\sqrt{5}$ *to verify that* $\mathbf{y}$ *is close to* $\mathbf{x}$*.*

```
sage: y = vector([-1,-4])
sage: x = vector([1,-5])
sage: (y-x).norm()
sqrt(5)
```

He verifies that $\|\mathbf{y}-\mathbf{x}\|\leq\sqrt{n}\det(L)^{1/n}$*.*

$$\begin{aligned}\|\mathbf{y}-\mathbf{x}\| &\leq \sqrt{n}\det(L)^{1/n}\\ &\leq \sqrt{2}\left|\det\left(\begin{bmatrix}13 & -20\\ 6 & -11\end{bmatrix}\right)\right|^{(1/2)}\\ &\leq \sqrt{2}(23)^{1/2}\\ &\approx 6.7823\end{aligned}$$

In fact, $\sqrt{5}\approx 2.236\leq 6.7823$*, so* $\mathbf{y}$ *is close to* $\mathbf{x}$*. Victor accepts Peggy's response.*

11.3 NTRU Zero Knowledge Proofs

In this section we present an original Zero Knowledge Proof protocol for the NTRU and show that our protocol has the desired properties. (See, for example, Theorem 11.3.1.)

The NTRU Zero Knowledge Proof Protocol has public parameters (n, p, q, d), where n and p are prime, $\gcd(n,q) = \gcd(p,q) = 1$, and $4d^2 + 2d < q$.

As with the NTRU Cryptosystem, we have

$$R = \frac{\mathbb{Z}[x]}{(x^n - 1)}, \qquad R_p = \frac{\mathbb{Z}_p[x]}{(x^n - 1)}, \qquad \text{and} \qquad R_q = \frac{\mathbb{Z}_q[x]}{(x^n - 1)}.$$

Peggy wants to prove to Victor that she knows $f(x)$ and $g(x)$ without him knowing enough information to determine either.

Key Creation
Peggy picks $f(x) \in \mathcal{T}(d+1, d)$ and $g(x) \in \mathcal{T}(d+1, d)$ such that $f(x)$ has inverse $F_q(x)$ in R_q and $g(x)$ has inverse $G_p(x)$ in R_p. She computes $h(x) = f(x)g(x)$ in R and makes $h(x)$ public.

Challenge
Victor picks an $m(x) \in \mathcal{T}(d, d)$ and computes $e(x) = h(x)m(x)$ to send to Peggy.

Response
Peggy receives $e(x)$ and computes $a(x) = F_q(x)e(x)$ in R_q. Note that $a(x) = F_q(x)e(x) = F_q(x)h(x)m(x) = F_q(x)f(x)g(x)m(x) = g(x)m(x)$ in R_q. Peggy then centerlifts $a(x)$ to R and reduces it in R_p to get $\hat{a}(x)$. She then computes $b(x) = G_p(x)\hat{a}(x)$ in R_p to send back to Victor.

Verification
Victor verifies that $b(x) = m(x)$ to accept Peggy's response. Note that $b(x) = G_p(x)\hat{a}(x) = G_p(x)g(x)m(x) = m(x)$.

Remark 11.3.1. *The NTRU Zero Knowledge Proof Protocol satisfies the following:*

1. *Completeness: If Peggy knows $f(x)$ and $g(x)$, she can find $m(x)$ every time. This will convince Victor that she does, in fact, know them.*

2. *Soundness: If Peggy does not know $f(x)$ and $g(x)$, she does not know $F_q(x)$ or $G_p(x)$, and so she cannot recover $m(x)$, forcing Victor to not accept her response.*

3. *Zero Knowledge: Victor cannot use the strategy that Peggy does, because he only knows $h(x)$. So, he cannot find $f(x)$ and $g(x)$.*

Theorem 11.3.1. *If the NTRU Zero Knowledge Protocol parameters (n,p,q,d) satisfy the condition that $q > 4d^2+2d$, then $b(x) = m(x)$, where $b(x)$ is the polynomial Peggy finds during her response, and $m(x)$ is Victor's plaintext challenge polynomial.*

Proof. Let $f(x), g(x) \in \mathcal{T}(d+1,d)$. We know that $h(x) = f(x)g(x)$. In $h(x)$, a coefficient is of maximum size when all $(d+1)$ 1s of $f(x)$ match up in multiplication with all $(d+1)$ 1s of $g(x)$. This yields a coefficient value of $d+1$. The same argument is true of the $(d)-1$s in $f(x)$ and $g(x)$, yielding a coefficient value of d. This means that the largest possible coefficient of $h(x)$ is $2d+1$.

We also have that $m(x) \in \mathcal{T}(d,d)$, and we calculate $e(x) = h(x)m(x)$. The maximum coefficient of $e(x)$ happens when the largest coefficient values of $h(x)$, $2d+1$, match in multiplication with the same signed values in $m(x)$ for a maximum of $(2d+1)d = 2d^2+d$. We assumed that $4d^2+2d < q$. Dividing by 2 gives $2d^2+d < \frac{q}{2}$. So, when Peggy centerlifts $a(x)$, she gets the exact coefficients back in R.

Now, we look at $b(x)$:

$$\begin{aligned} b(x) &= G_p(x)\hat{a}(x) \\ &= G_p(x)F_q(x)e(x) \\ &= G_p(x)F_q(x)h(x)m(x) \\ &= G_p(x)F_q(x)f(x)g(x)m(x) \\ &= G_p(x)g(x)m(x) \\ &= m(x) \end{aligned}$$

□

Example 11.3.1 (NTRU Zero Knowledge Proof)**.**
An NTRU Zero Knowledge Proof Protocol has public parameters $(n,p,q,d) = (5,7,23,2)$. Note that $23 = q > 4(2)^2+2(2) = 20$. We have

$$R = \frac{\mathbb{Z}[x]}{x^5-1} \qquad R_p = R_7 = \frac{\mathbb{Z}_7[x]}{x^5-1} \qquad R_q = R_{23} = \frac{\mathbb{Z}_{23}[x]}{x^5-1}$$

Key Creation

Peggy chooses $f(x) = -1 - x + x^2 + x^3 + x^4 \in \mathcal{T}(3,2)$ *and finds* $F_{23} = f^{-1}(x)$ *in* R_{23} *to be* $F_{23}(x) = 12x^3 + 12x^2$.

```
sage: R.<x> = PolynomialRing(GF(23),x)
sage: R23.<x> = R.quotient_ring(x^5-1)
sage: f = R23(-1-x+x^2+x^3+x^4)
sage: F = f^(-1)
sage: F
12*x^3 + 12*x^2
```

She also chooses $g(x) = 1 - x + x^2 - x^3 + x^4 \in \mathcal{T}(3,2)$ *and finds* $G_7(x) = g^{-1}(x)$ *in* R_7 *to be* $G_7(x) = 4x + 4$.

```
sage: S.<x> = PolynomialRing(GF(7),x)
sage: S7.<x> = S.quotient_ring(x^5-1)
sage: g = S7(1-x+x^2-x^3+x^4)
sage: G = g^(-1)
sage: G
4*x + 4
```

Peggy then computes $h(x) = f(x)g(x)$ *in* R *to get* $h(x) = x^4 + x^3 + x^2 + x - 3$.

```
sage: h = f*g
sage: h
x^4 + x^3 + x^2 + x - 3
```

She publishes $h(x)$. *Peggy wants to prove to Victor that she knows* $f(x)$ *and* $g(x)$ *without giving Victor any information.*

Challenge

Victor picks $m(x) = x - x^2 + x^3 - x^4$ *in* $\mathcal{T}(2,2)$. *He computes his challenge polynomial* $e(x) = h(x)m(x) = 4x^4 - 4x^3 + 4x^2 - 4x$ *and sends that back to Peggy.*

```
sage: m = x - x^2 + x^3 - x^4
sage: e = h*m
sage: e
4*x^4 - 4*x^3 + 4*x^2 - 4*x
```

Response

Peggy receives $e(x)$ *and computes* $a(x) = F_{23}(x)e(x) = 19x^4 + 2x^3 + 21x + 4$.

```
sage: a = F*e
sage: a
19*x^4 + 2*x^3 + 21*x + 4
```

She then center lifts $a(x)$ *to* $\hat{a}(x)$ *in* R *and reduces it in* R_7 *to get that* $\hat{a}(x) = -4x^4 + 2x^3 - 2x + 4 = 3x^4 + 2x^3 + 5x^2 + 4$. *She then calculates* $b(x) = G_7(x)\hat{a}(x)$ *to recover* $m(x)$ *and prove to Victor that she knows* $f(x)$ *and* $g(x)$.

```
sage: ahat = ZZ['x']([coeff.lift_centered() for coeff in a.lift()])
sage: ahat
-4*x^4 + 2*x^3 - 2*x + 4

sage: G*ahat
6*x^4 + x^3 + 6*x^2 + x
```

Verification

Notice that Sage returns $6x^4 + x^3 + 6x^2 + x$. *This is equivalent to* $-x^4 + x^3 - x^2 + x = m(x)$. *Victor accepts Peggy's response as valid.*

11.4 Computer Exercises

1. Peggy wants to prove to Victor that she knows the good basis $\mathscr{B} = \left\{\begin{bmatrix} 5 \\ -1 \end{bmatrix}, \begin{bmatrix} 2 \\ 5 \end{bmatrix}\right\}$ without telling him that basis. She publishes $\mathscr{B}' = \left\{\begin{bmatrix} 12 \\ 3 \end{bmatrix}, \begin{bmatrix} 29 \\ 5 \end{bmatrix}\right\}$. Victor sends $\mathbf{x} = \begin{bmatrix} 9 \\ 7 \end{bmatrix}$ to Peggy as a challenge. What is her response?

2. Using the GGH Zero Knowledge Proof Protocol, Peggy publishes her public basis as $\mathscr{B}' = \left\{\begin{bmatrix} 127 \\ -62 \\ 37 \end{bmatrix}, \begin{bmatrix} 20 \\ -12 \\ 12 \end{bmatrix}, \begin{bmatrix} 25 \\ -16 \\ 19 \end{bmatrix}\right\}$ for lattice $\mathscr{L}$. Victor chooses $\mathbf{x} = \begin{bmatrix} 428 \\ 510 \\ 324 \end{bmatrix}$ to send to Peggy as a challenge. She returns the coordinates of her vector in terms of the bad basis to be $\begin{bmatrix} 178 \\ -3310 \\ 1761 \end{bmatrix}$.

   (a) What is Peggy's vector, $\mathbf{y}$?

   (b) Determine whether or not $\mathbf{x}$ and $\mathbf{y}$ are reasonably close. Should Victor accept Peggy's response?

3. Peggy chooses a good lattice basis for a GGH Zero Knowledge Proof to be $\mathscr{B} = \left\{\begin{bmatrix} 3 \\ -6 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \\ 6 \end{bmatrix}, \begin{bmatrix} 7 \\ 2 \\ -3 \end{bmatrix}\right\}$.

   (a) Verify that $\mathscr{B}$ is a good basis for $\mathscr{L}$.

   (b) Find a bad basis for the same lattice by multiplying the matrix of the good basis by a unimodular matrix. Verify that your new basis is, in fact, bad. This is Peggy's public basis, $\mathscr{B}'$.

   (c) Suppose that Victor chooses $\mathbf{x} = \begin{bmatrix} 319 \\ 437 \\ 413 \end{bmatrix}$. Find $\mathbf{y}$, the closest vector to $\mathbf{x}$ in $\mathscr{L}$, for Peggy.

   (d) Write your answer from part (c) in terms of the bad basis that you chose so that Peggy can send it to Victor.

   (e) Help Victor verify that the vectors $\mathbf{x}$ and $\mathbf{y}$ are close so that he can accept Peggy's response.

4. The NTRU Zero Knowledge Proof Protocol has public parameters $(n,p,q,d)=(5,3,31,2)$. Peggy chooses secret $f(x)\in\mathscr{T}(3,2)$ to be $f(x)=x^4+x^3-x^2-x+1$ and secret $g(x)\in\mathscr{T}(3,2)$ to be $g(x)=-x^4+x^3-x^2+x+1$.

   (a) Find Peggy's public $h(x)$.

   (b) Victor chooses $m(x)\in\mathscr{T}(2,2)$ to be $m(x)=x^4-x^2-x+1$. What is his challenge, $e(x)$, that he sends to Peggy?

   (c) Peggy receives the challenge, $e(x)$. Calculate $a(x)$, and use it to find the value of $b(x)$ that Peggy sends to Victor as her response.

   (d) Should Victor accept Peggy's response as valid?

BIBLIOGRAPHY

[1] Ahrens, K. A. "Combinatorial Applications of the k-Fibonacci Numbers: A Cryptographically Motivated Analysis". PhD thesis. North Carolina State University, 2020.

[2] Axler, S. *Linear Algebra Done Right*. Springer International Publishing, 2015.

[3] Babai, L. "On Lovász'lattice reduction and the nearest lattice point problem". *Combinatorica* **6**.1 (1986), pp. 1–13.

[4] Batson, S. C. "The linear transformation that relates the canonical and coefficient embeddings of ideals in cyclotomic integer rings". *International Journal of Number Theory* **13**.09 (2017), pp. 2277–2297.

[5] Batson, S. C. "On the Relationship Between Two Embeddings of Ideals into Geometric Space and the Shortest Vector Problem in Principal Ideal Lattices". PhD thesis. North Carolina State Univeristy, 2015.

[6] Chaum, D., Evertse, J.-H. & Graaf, J. v. d. "An improved protocol for demonstrating possession of discrete logarithms and some generalizations". *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1987, pp. 127–141.

[7] Cybersecurity Education, J. T. F. on. *Cybersecurity Curricula 2017*. Tech. rep. 2017.

[8] Diffie, W. & Hellman, M. "New directions in cryptography". *IEEE transactions on Information Theory* **22**.6 (1976), pp. 644–654.

[9] Effectiveness, O. E. C. *Judging the Quality of K-12 Mathematics Evaluations*. 2004.

[10] Gallian, J. A. *Contemporary abstract algebra*. Chapman and Hall/CRC, 2021.

[11] Goldreich, O., Goldwasser, S. & Halevi, S. "Public-key cryptosystems from lattice reduction problems". *Annual International Cryptology Conference*. Springer. 1997, pp. 112–131.

[12] Hoffman, K. *Linear algebra*. Englewood Cliffs, NJ, Prentice-Hall, 1971.

[13] Hoffstein, J., Pipher, J. & Silverman, J. H. "NTRU: A ring-based public key cryptosystem". *International algorithmic number theory symposium*. Springer. 1998, pp. 267–288.

[14] Hoffstein, J., Pipher, J. & Silverman, J. H. *An Introduction to Mathematical Cryptography*. Springer, 2014.

[15] Hungerford, T. W. *Abstract algebra: an introduction*. Cengage Learning, 2012.

[16] Knospe, H. *A Course in Cryptography*. Vol. 40. American Mathematical Soc., 2019.

[17] *Largest known prime number*. 2022.

[18] Lenstra, A. K., Lenstra, H. W. & Lovász, L. "Factoring polynomials with rational coefficients". *Mathematische annalen* **261** (1982), pp. 515–534.

[19] May SJ, M. "Using Maple worksheets to enable student explorations of cryptography". *Cryptologia* **33**.2 (2009), pp. 151–157.

[20] NSA. *Home*. 2022.

[21] Peikert, C. et al. "A decade of lattice cryptography". *Foundations and Trends® in Theoretical Computer Science* **10**.4 (2016), pp. 283–424.

[22] Python. *Applications for Python*. 2021. URL: `https://www.python.org/about/apps/` (visited on 05/02/2021).

[23] *QR decomposition*. 2022.

[24] Quisquater, J.-J. et al. "How to explain zero-knowledge protocols to your children". *Conference on the Theory and Application of Cryptology*. Springer. 1989, pp. 628–631.

[25] Rivest, R. L., Shamir, A. & Adleman, L. "A method for obtaining digital signatures and public-key cryptosystems". *Communications of the ACM* **21**.2 (1978), pp. 120–126.

[26] Shor, P. W. "Algorithms for quantum computation: discrete logarithms and factoring". *Proceedings 35th annual symposium on foundations of computer science*. IEEE. 1994, pp. 124–134.

[27] Suzuki, J. *LLL example 1 - youtube*. 2015.

[28] Wehden, K., Faro, I. & Gambetta, J. *IBM's roadmap for building an open quantum software ecosystem*. 2021. URL: `https://www.ibm.com/blogs/research/2021/02/quantum-development-roadmap/` (visited on 07/26/2021).