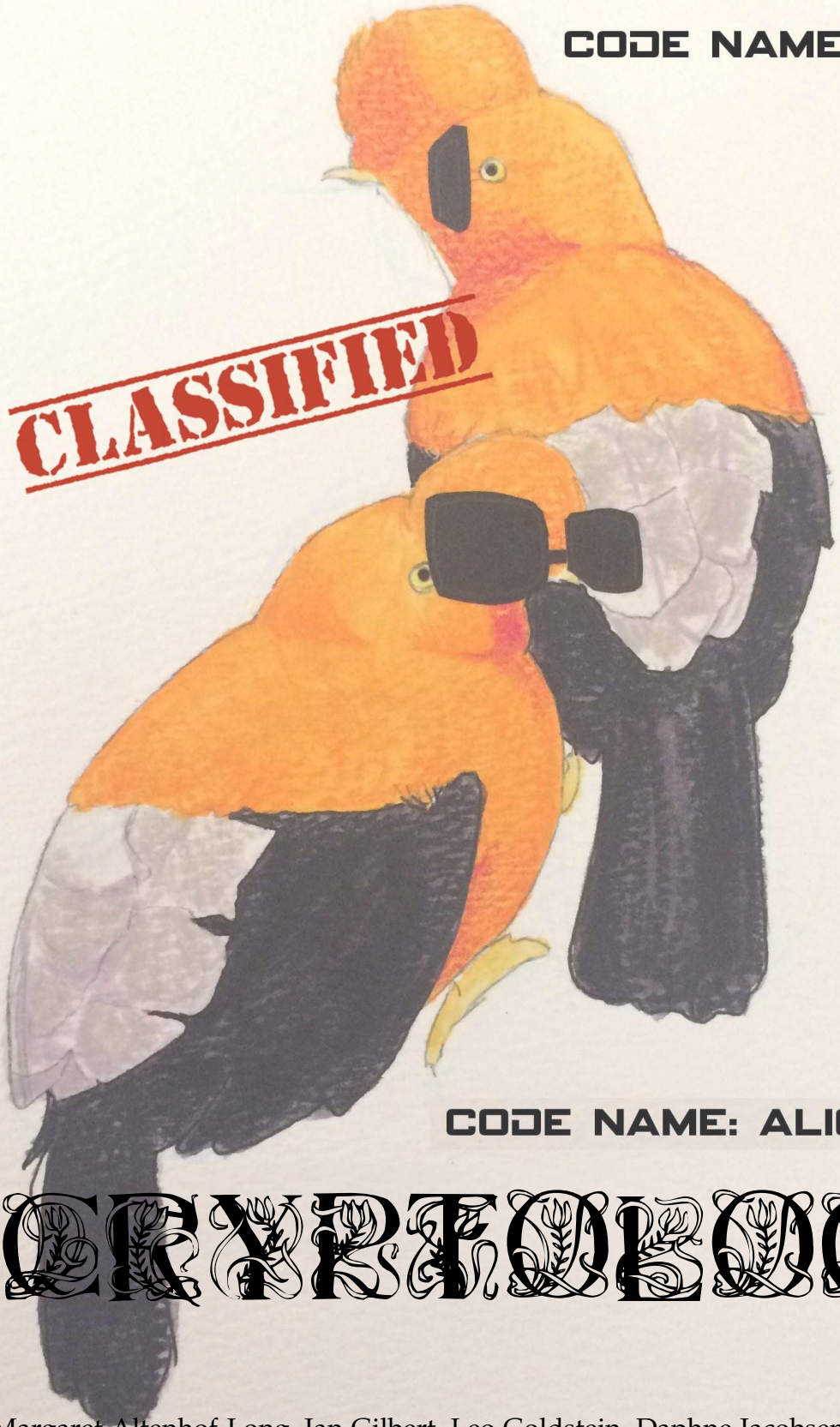
CODE NAME: BOB



CODE NAME: ALICE

CRYPTOLOGY

Margaret Altenhof-Long, Ian Gilbert, Leo Goldstein, Daphne Jacobsen, Jaime McConachie, Erin McNicholas, Alicia Nichols, Dusan Pekich, Journey Penney, Miles Smith

A Bevy of Stupendous Seniors:

The mathematics in this Cryptology text was written collaboratively by the Mathematics Senior Seminar class of 2018 at Willamette University under my direction.

At the beginning of the semester, students were given a “skeleton" of this book. It included definitions and statements of theorems. Many definitions were left incomplete for students to formalize. Without consulting outside sources, students discovered all the mathematical proofs and examples included in the co-authored sections. During class they presented their solutions to each other. After a solution was agreed upon, they took turns LaTeXing solutions, generating accompanying figures, and embedding their work in the skeleton text.

In addition to filling in the skeleton text, students researched individual projects - and did an amazing job! They presented their findings to the University community, and contributed sections on their individual project to the text. This is the resulting document.

Individually written/created sections:

- Cover Art[1] "Sexy Spy Birds," by Jaime McConachie
- Section 2.4: A History of Symmetric Key Ciphers, by Jaime McConachie
- Section 3.4: Solving the Discrete Logarithm Problem, by Daphne Jacobsen
- Chapter 5: Complexity and Primality, by Leo Goldstein
- Section 6.1: Information Theory, by Journey Penney
- Section 6.2: Probabilistic Encryption, by Margaret Altenhof-Long
- Chapter 8: Choosing a Secure Elliptic Curve, by Dusan Pekich
- Chapter 9: Demystifying the Weil Pairing, by Alicia Nichols
- Chapter 10: Multivariate Cryptosystems, by Miles Smith
- Chapter 11: Lattices in Cryptology, by Ian Gilbert
- Appendix C: Sexy Spy Birds, by Jaime McConachie

They demonstrated an amazing spirit of collaboration, tenacity, creativity, content mastery, and skill. They worked hard, hopefully had fun, and should definitely be proud of their accomplishments. I am extremely proud of each of them. They have all taught me so much and impressed me with their resilience, compassion, and insight.

The Cryptology texts used in the creation of the skeleton of this book are *An Introduction to Mathematical Cryptography* [1], *The Code Book* [2], and *Cryptography Theory and Practice* [3]. In particular, I borrowed heavily from *An Introduction to Mathematical Cryptography*. Any mistakes or inconsistencies found in the skeleton of the text, however, are certainly my own.

Congratulations Maggie, Ian, Leo, Daphne, Jaime, Alicia, Dan, Journey, and Miles! You did a fantastic job! I will miss you dearly!

Professor Erin McNicholas

May 13, 2018.

1 Throughout the semester, students shared "fun facts." Jaime enthralled us with tales and drawings of cool animals, including the Andean cock-of-the-rock. Her fascinating tidbits about these birds inspired our "Sexy Spy Birds" cover art. For more information, see Appendix C.

CONTENTS

1

MATHEMATICAL UNDERPINNINGS

It is a cold wet dawn on the morning of August 4, 1914. Germany has invaded neutral Belgium and England is on the brink of war. Under the cloak of darkness, the British postal ship CS Alert approaches the German shore. With all lights on board extinguished, the crew drops anchor. As rain lashes the deck, the crew hauls up a clutch of undersea cables. They cut the cables and drop them back into the Atlantic, effectively cutting Germany's most secure means of communication. For the rest of the war, Germany will have to transmit messages along cables crossing Britain.

January 9, 1917. Although they continue to provide the Allies with supplies, a reluctant America has managed to stay out of the conflict. The loss of 128 US civilian lives on the *Lusitania* would have surely brought the US into the war in 1915 had it not been for the German's assurances that henceforth U-boats would surface before attack, reducing accidental attacks on civilian ships. Two years later, a frustrated Supreme High Command meets with the Kaiser to convince him to renege on this promise. It is time to embark on a course of unrestricted submarine warfare. The U-boats are almost invulnerable if they launch their torpedoes while still submerged. The Kaiser is convinced. War with the US is inevitable.

Germany needs to force an Allied surrender before the US can mobilize its troops and make an impact on the European front. The newly appointed German Foreign Minister Arthur Zimmermann has a plan to delay, if not prevent, American involvement in the war. On January 16, 1917 he cables the German Ambassador in Mexico. Because of the actions of the CS Alert over two years earlier, this cable passes along lines controlled by the British.

Meanwhile, in room 40 of the British Admiralty's cipher bureau, the Reverend Montgomery, a gifted translator of German theological works, is handed Zimmermann's intercepted telegram, see Figure 1. Room 40 houses an eclectic group of linguists, classical scholars, and puzzle enthusiasts. Although far from trivial, Montgomery is able to crack Zimmermann's codebook cipher.

> We intend to begin unrestricted submarine warfare on the first of February. We shall endeavor in spite of this to keep the United States neutral. In the event of this not succeeding, we make Mexico a proposal of alliance on the

> following basis: make war together, make peace together, generous financial support, and an understanding on our part that Mexico is to reconquer the lost territory in Texas, New Mexico and Arizona. The settlement in detail is left to you.
>
> You will inform the President [of Mexico] of the above most secretly, as soon as the outbreak of war with the United States is certain, and add the suggestion that he should, on his own initiative, invite Japan to immediate adherence and at the same time mediate between Japan and ourselves.
>
> Please call the President's attention to the fact that the unrestricted employment of our submarines now offers the prospect of compelling England to make peace within a few months. Acknowledge receipt. – Zimmermann.

Once the telegram is released to the press, America's reluctance to join the conflict is turned to outrage. Woodrow Wilson, who at the start of 1917 said it would be a "crime against civilization" to lead his nation into war, was encouraging Congress to do just that by April.

> In itself the Zimmermann telegram was only a pebble on the long road of history. But a pebble can kill a Goliath, and this one killed the American illusion that we could go about our business happily separate from other nations. In world affairs it was a German Minister's minor plot. In the lives of the American people it was the end of innocence. – Barbara Tuchman, American Historian

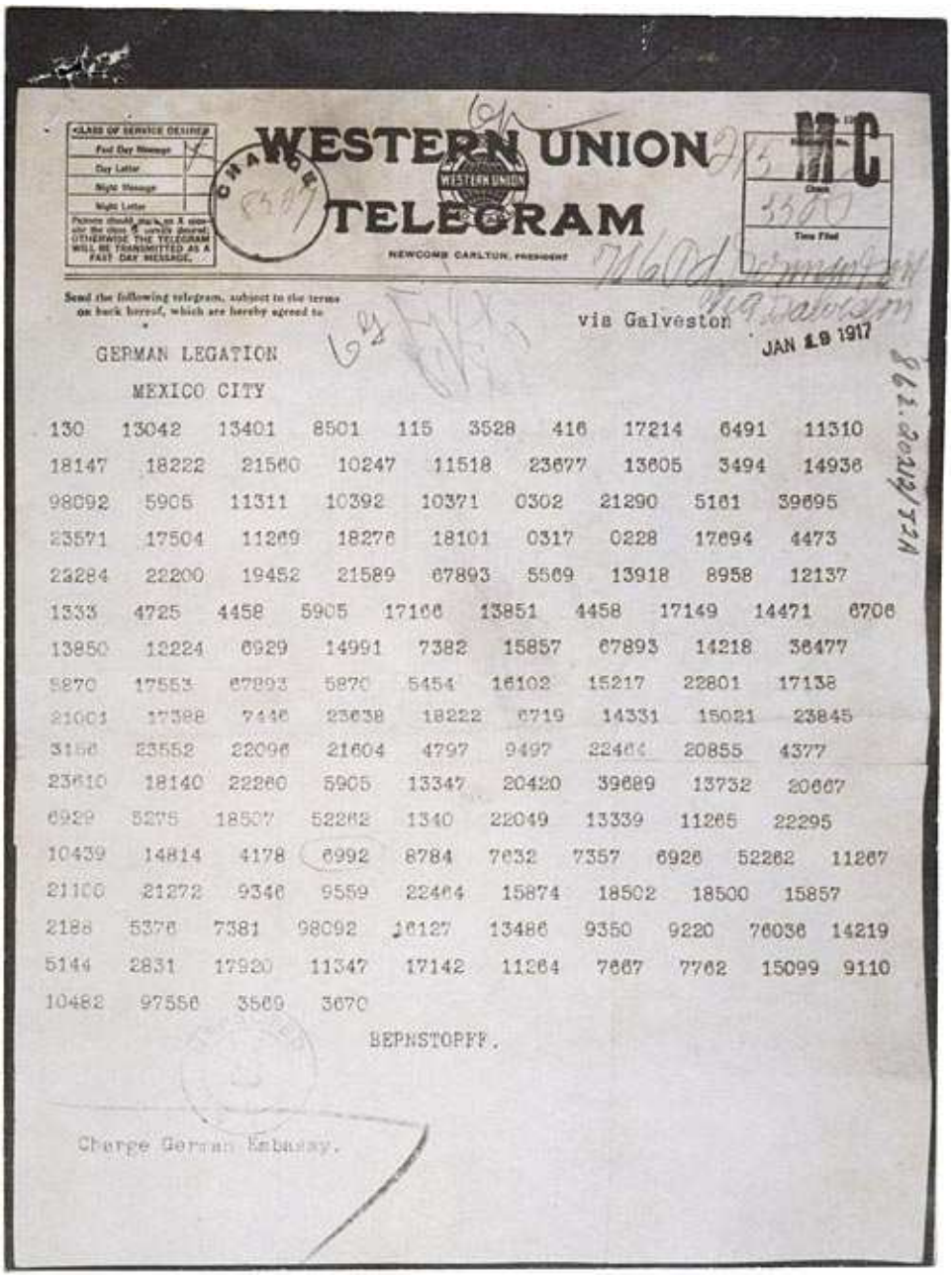
As noted by Simon Singh, author of *The Code Book*, "A single breakthrough by Room 40 cryptanalysts had succeeded where three years of British diplomacy had failed."

In this text we'll examine **cryptology**, the combined study of cryptography and cryptanalysis. **Cryptography** is the science of enciphering and deciphering messages using a known secret key. **Cryptanalysis** is the art and science of recovering information from ciphers without knowledge of the secret key. Both branches of cryptology have had significant impacts on the course of human history.

1.1 A SPRINKLING OF NUMBER THEORY

> "No one has yet discovered any warlike purpose to be served by the theory of numbers or relativity, and it seems unlikely that anyone will do so for many years." – G.H. Hardy

As a committed pacifist and someone who was deeply troubled by the use of science to advance chemical warfare during World War I, Hardy took pride and comfort in his belief that no discovery of his had made, or was "likely to make, directly or indirectly,



WESTERN UNION TELEGRAM

NEWCOMB CARLTON, PRESIDENT

Send the following telegram, subject to the terms on back hereof, which are hereby agreed to

via Galveston

JAN 19 1917

GERMAN LEGATION

MEXICO CITY

130 13042 13401 8501 115 3528 416 17214 6491 11310
18147 18222 21560 10247 11518 23677 13605 3494 14936
98092 5905 11311 10392 10371 0302 21290 5161 39695
23571 17504 11269 18276 18101 0317 0228 17694 4473
23284 22200 19452 21589 67893 5569 13918 8958 12137
1333 4725 4458 5905 17166 13851 4458 17149 14471 6706
13950 12224 6929 14991 7382 15857 67893 14218 36477
5870 17553 67893 5870 5454 16102 15217 22801 17138
21001 17388 7446 23638 18222 6719 14331 15021 23845
3156 23552 22096 21604 4797 9497 22464 20855 4377
23610 18140 22260 5905 13347 20420 39689 13732 20667
6929 5275 18507 52262 1340 22049 13339 11265 22295
10439 14814 4178 6992 8784 7632 7357 6926 52262 11267
21100 21272 9346 9559 22464 15874 18502 18500 15857
2188 5376 7381 98092 16127 13486 9350 9220 76036 14219
5144 2831 17920 11347 17142 11264 7667 7762 15099 9110
10482 97556 3569 3670

BEPNSTORFF.

Chrge German Embassy.

Figure 1.: Zimmerman's encoded telegraph

for good or ill, the least difference to the amenity of the world.[1]" How could Hardy have known that number theory, the queen of mathematics[2], was about to join the fray of applied mathematics? Not only was number theory instrumental in breaking the

1 G.H. Hardy, A Mathematician's Apology, Cambridge University Press. 1940

2 "Mathematics is the queen of the sciences, and number theory is the queen of mathematics." – attributed to Carl Friedrich Gauss

German ciphers of WWII, it is the mathematical structure upon which modern commerce and communication are based.

1.1.1 *A Dash of Divisibility*

The following algorithms and definitions attributed to Euclid date back to at least 300BC. They are described in Euclid's *Elements*, arguably the most influential treatise in the history of mathematics. In *Elements*, Euclid not only organized and presented the foundations of geometry and number theory, he helped establish the use of deductive logic as the basis of mathematical proof.

Definition 1.1.1. Let a and b be integers with $b \neq 0$. We say b *divides* a, or that a is *divisible* by b, if there exists an integer c such that $a = bc$. The statement "b divides a" is denoted $b|a$.

Definition 1.1.2. A *common divisor* of two integers a and b is a positive integer d that divides both a and b. The *greatest common divisor* of a and b is the largest positive integer d such that $d|a$ and $d|b$. The greatest common divisor of a and b is denoted $\gcd(a, b)$.

Definition 1.1.3. A *common multiple* of two integers a and b is a positive integer l that is divisible by both a and b. The *least common multiple* of a and b is the smallest positive integer l such that $a|l$ and $b|l$. The least common multiple of a and b is denoted $\operatorname{lcm}(a, b)$.

Remark 1.1.4. For $t \in \mathbb{Z}$, if $t|a$ and $t|b$ then $t|\gcd(a, b)$.

Definition 1.1.5 (Division Algorithm)**.** Let a and b be positive integers. Then there exist unique integers q (the *quotient*) and r (the *remainder*) such that

$$a = bq + r,$$

and $0 \leq r < b$.

Theorem 1.1.6 (The Euclidean Algorithm)**.** *Let a and b be positive integers with $a \geq b$. The following algorithm computes $\gcd(a, b)$ in a finite number of steps.*

1. *Let $r_0 = a$ and $r_1 = b$.*
2. *set $i = 1$*
3. *Divide r_{i-1} by r_i to get a quotient q_i and remainder r_{i+1}*
4. *If the remainder $r_{i+1} = 0$, then $r_i = \gcd(a, b)$ and the algorithm terminates*
5. *Otherwise, set $i = i + 1$ and go to step 3.*

Example 1.1.7. To find the $\gcd(16261, 85652)$ using the Euclidean Algorithm, first let $a = 85652$ and $b = 16261$. Now begin the algorithm: let $r_0 = a$ and $r_1 = b$, and set $i = 1$.

$$\frac{r_{i-1}}{r_i} = \frac{r_0}{r_1} = \frac{85652}{16261} = 5 \qquad \text{remainder} \qquad 4347$$

In other words, $85652 = 5 * 16261 + 4347$. Since $r_{i+1} = r_2 = 4347 \neq 0$ we continue with the algorithm.

$$16261 = 4347 * 3 + 3220$$
$$4347 = 3220 * 1 + 1127$$
$$3220 = 1127 * 2 + 966$$
$$1127 = 966 * 1 + 161$$
$$966 = 161 * 6 + 0$$

Notice that $r_{i+1} = r_7 = 0$, and so by the Euclidean Algorithm $r_i = \gcd(a, b)$. Thus $r_6 = \gcd(16261, 85652) = 161$.

Definition 1.1.8. (Linear Combination) Given two integers x and y, a linear combination of x and y is an integer z of the form

$$ax + by = z$$

where a and b are integers.

Lemma 1.1.9. *Given the sequence of $r_0, r_1, ..., r_n$ as defined by the Euclidean Algorithm, an arbitrary remainder, r_k, such that $k \geq 2$ can be expressed as a linear combination of r_0 and r_1.*

Proof. Consider the sequence r_i and q_i as defined by the Euclidean Algorithm. Let r_k be an arbitrary remainder of said sequence Thus $r_{k-2} = r_{k-1} * q_{k-1} + r_k$ and

$$r_k = r_{k-2} - r_{k-1} * q_{k-1}$$

such that q_k is an arbitrary quotient.

Base Case: Let $k = 2$, thus $r_2 = r_{2-2} - r_{2-1} * q_{2-1} = r_0 - q_1 * r_1$, which is a linear combination of r_0 and r_1. Now let $k = 3$, thus $r_3 = r_{3-2} - r_{3-1} * q_{3-1} = r_1 - q_2 * r_2$

Induction Hypothesis: Suppose that $r_i = ar_0 + br_1$ for $2 \leq i \leq k$ such that a, b are integers.

Consider $r_{k+1} = r_{k-1} - r_k q_k$ by the Euclidean Algorithm. By our induction hypothesis there exists integers w, x, y, z such that

$$r_{k+1} = wr_0 + x_1 - (yr_0 + zr_1)q_k = (w - q_k y)r_0 + (x - q_k z)r_1$$

Since the integers are closed over multiplication and addition, this is a linear combination of r_0 and r_1. Therefore by induction, an arbitrary remainder r_k such that $k \geq 2$ can be expressed as a linear combination of r_0 and r_1. □

Theorem 1.1.10 (Extended Euclidean Algorithm)**.** *Let a and b be positive integers. Then there exist integers u and v satisfying*

$$au + bv = \gcd(a, b).$$

Furthermore, if (u_0, v_0) is any one solution, then every solution has the form

$$u = u_0 + \frac{bk}{\gcd(a,b)} \qquad \text{and} \qquad v = v_0 - \frac{ak}{\gcd(a,b)} \qquad \text{for some } k \in \mathbb{Z}.$$

Proof. Consider r_i and q_i as defined by the Euclidean Algorithm and let $a = r_0$ and $b = r_1$ for some integers a, b. So the $gcd(a, b) = r_n$. Thus, by 1.1.9 it follows that the $gcd(a, b)$ can be expressed as a linear combination of a and b

$$au + bv = gcd(a, b) \qquad \text{for some} \qquad u, v \in \mathbb{Z}.$$

Since $au + bv = gcd(a, b)$ we can write

$$v = \frac{gcd(a,b)}{b} - \frac{a}{b}u$$

which represents the equation of a line. Now let (u_0, v_0) be a particular solution and let (u, v) be an arbitrary solution. We know that $u = u_0 + bz$ and $v = v_0 - az$ since (u_0, v_0) is a point on a line with slope $-\frac{a}{b}$ for some real number z. Since u, v, u_0, v_0 are integers, bz and az must also be integers. Thus we have 2 cases in which z is either a rational or an irrational.

Case 1: Let z be rational. Thus $z = \frac{j}{d}$ such that $j$and d are integers, $d|a$ and $d|b$. By remark 1.1.4 $d|\gcd(a, b)$ and by the definition of divides, $\gcd(a, b) = dx$ for some x in the integers. Now consider z. We know that

$$z = \frac{j}{d} = \frac{j}{d}(1) = \frac{j}{d}\left(\frac{x}{x}\right) = \frac{jx}{dx} = \frac{jx}{\gcd(a,b)}$$

Thus $z = \frac{jx}{\gcd(a,b)}$. Since j and x were arbitrary integers they can be represented by some integer k. By substituting for z we get

$$u = u_0 + \frac{bk}{\gcd(a,b)}, \quad v = v_0 - \frac{ak}{\gcd(a,b)}$$

for some k in the integers.

Case 2: Let z be an irrational. This implies that az and bz are irrational, which is a contradiction. So z cannot be irrational.

Therefore if (u_0, v_0) is any one solution, then every solution has the form

$$u = u_0 + \frac{bk}{\gcd(a,b)} \qquad \text{and} \qquad v = v_0 - \frac{ak}{\gcd(a,b)} \qquad \text{for some } k \in \mathbb{Z}.$$

□

1.1.2 *A Peppering of Primes*

Definition 1.1.11. An integer p is called *prime* if $p \geq 2$ and if the only positive integers dividing p are p and 1.

Definition 1.1.12. Let a and b be integers. We say that a and b are *relatively prime* if $\gcd(a, b) = 1$.

Theorem 1.1.13 (Generalization of Euclid's Lemma)**.** *For $t \in \mathbb{Z}$, if $t|ab$ and $\gcd(a, t) = 1$, then $t|b$.*

Theorem 1.1.14. *For all a, b, and $m \in \mathbb{Z}$, if $a|m$ and $b|m$ and $\gcd(a, b) = 1$, then $ab|m$.*

Theorem 1.1.15 (The Fundamental Theorem of Arithmetic)**.** *Let $a \geq 2$ be an integer. Then a can be factored as a product of prime numbers. Furthermore, up to a rearrangement of terms, this factorization into primes is unique.*

Theorem 1.1.16 (Euclid's Theorem)**.** *There are infinitely many primes.*

Proof. By way of contradiction, suppose there are a finite number of primes, $p_1, p_2, \ldots, p_n$ and consider $x = p_1 * p_2 * \ldots * p_n$. Consider $\gcd(x, x + 1) = a$ for some integer a, since $a|x$ and $a|(x + 1)$, it must also divide 1 and therefore $a = 1$. By the Fundamental Theorem of Arithmetic, $x + 1$ can be factored as a product of some number of primes $p_k, ..., p_j$. Since the $\gcd(x, x + 1) = 1$, none of these prime divisors $p_k, ..., p_j$ can divide x. But x was defined as the product of all primes, and so we have reached a contradiction. Therefore there are infinitely many primes. □

The previous theorem leads naturally to an examination of the distribution of primes among the natural numbers. What is the probability that a number chosen at random is prime? This seemingly innocuous line of questioning leads to one of the most beautiful and widely studied unsolved mysteries of modern mathematics.

> "There are two facts about the distribution of prime numbers of which I hope to convince you so overwhelmingly that they will be permanently engraved in our hearts. The first is that, despite their simple definition and role as the building blocks of the natural numbers, the prime numbers belong to the most arbitrary and ornery objects studied by mathematicians: they grow like weeds among the natural numbers, seeming to obey no other law than that of chance, and nobody can predict where the next one will sprout. The second fact is even more astonishing, for it states just the opposite: that the prime numbers exhibit stunning regularity, that there are laws governing their

> behaviour, and that they obey these laws with the almost military precision."
> - Don Zagier[3]

One of the most famous laws governing the behavior of the primes is the Prime Number Theorem. First conjectured by Gauss, it was proved by de Vallee and Hadamard in 1896.

Theorem 1.1.17 (Prime Number Theorem)**.** *Given a natural number X, the number of primes less than or equal to X is asymptotically $\sim \int_2^X \frac{dt}{\ln(t)}$, i.e.*

$$\lim_{X\to\infty} \frac{\pi(X)}{X/\ln(X)} = 1$$

where $\pi(X) = |\{p \in primes | 2 \leq p \leq X\}|$.

Example 1.1.18. To use the Prime Number Theorem to approximate the probability that a number between 900000 and 1100000 is prime, first calculate $\pi(1100000) - \pi(900000)$ and then divide by $1100000 - 900000$. Assuming 900000 and 1100000 to be sufficiently close to infinity the function reduces to the following:

$$\pi(1100000) \approx \frac{1100000}{ln(1100000)} \qquad \text{and} \qquad \pi(900000) \approx \frac{900000}{ln(900000)}$$

So the probability that a number between 900000 and 1100000 is prime is approximately

$$\frac{79075 - 65645}{1100000 - 900000} = \frac{13430}{200000} = 0.067 \qquad \text{or} \qquad 6.7\%$$

Calculating $\pi(X)$ for $X < 10^{20}$, we find the $\int_2^X \frac{dt}{\ln(t)}$ gives an overestimate for the number of primes less than or equal to X. Searching for an upper bound on X for which this integral is not an overestimate led to $10^{10^{10^{34}}}$, the largest number with any known significant mathematical meaning[4]. This bound, known as Skewes number, has since been whittled down to 1.39822×10^{316}.

In his 1859 article, "On the Number of Primes Less Than a Given Magnitude," Bernhard Riemann presented a way of calculating $\pi(X)$ from the non-trivial zeros of the function

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} \qquad \text{for } s \in \mathcal{C}.$$

This function, first formulated by Euler, is known as the Riemann Zeta Function. The Riemann Hypothesis, which states that all nontrivial zeros (i.e. zeros not equal to a negative even integer value) of $\zeta(s)$ have real part equal to $\frac{1}{2}$, is one of the open millennium

3 D. Zagier, *The First 50 Million Prime Numbers*, The Mathematical Intelligencer, Vol. 0, August 1977

4 This statement is as false as $10^{10^{10^{34}}}$ is less than Graham's Number. – Ed.

prize problems posted by the Clay Mathematics Institute[5]. Riemann proved that if the Riemann Hypothesis is true, then we can find $\pi(X)$ by adding wave functions whose frequency depends on the complex components of the nontrivial zeros of $\zeta(s)$.

> "That the distribution of prime numbers can be so accurately represented in [this way] is absolutely amazing and incredibly beautiful. It tells of an arcane music and a secret harmony composed by the prime numbers." - E. Bombieri[6]

1.2 AN ASSORTMENT OF ABSTRACT ALGEBRA

Abstract algebra is the study of algebraic structures, sets with operations defined on the elements of the set.

1.2.1 *A Gathering of Groups, Rings, and Fields*

Definition 1.2.1 (Group)**.** Let G be a nonempty set together with a binary operation $*$ (usually called multiplication). We say G is a *group* under this operation if the following four properties are satisfied.

1. G is closed under the operation, i.e. for all $a, b \in G$, $ab = a * b$ is in G
2. The operation is *associative*
3. There is an *identity* element e such that $ae = ea = a$ for all $a \in G$
4. For each element a in G, there is an element a^{-1} in G (called the *inverse* of a) such that $aa^{-1} = a^{-1}a = e$.

If, in addition, the operation on G is commutative, the group is called an *Abelian Group*.

Definition 1.2.2. The number of elements in the group G is called the *order* of the group and denoted $|G|$. Given an element $a \in G$, the order of a, denoted $|a|$, is the smallest positive integer n such that $a^n = e$.

Proposition 1.2.3. *Given an element g with order n, if $g^k = e$ then $n|k$.*

Proof. Let g be an element in G such that $|g| = n$ and $g^k = e$ for some integers n and k. By definition of order, $n \leq k$ and by the division algorithm, there exist integers q, r such that $k = n * q + r$, where $0 \leq r < n$. Thus, we can rewrite $g^k = g^{n*q} * g^r$. By way

5 So prove or disprove it and you will win $1,000,000.

6 E. Bombieri, *Prime Territory: Exploring the Infinite Landscape at the Base of the Number System*, The Sciences. Vol 32(5), 1992

of contradiction, suppose $r \neq 0$. By definition of order, $g^{n*q} = e$, and by substitution we have, $g^k = g^{n*q} * g^r$ equals $e = e * g^r$, thus $e = g^r$. Since r was strictly less than n and $g^r = e$, we have a contradiction and thus $r = 0$. Therefore, $k = n * q$, so by the definition of divides, $n|k$. □

Proposition 1.2.4. *For all elements g in a group G,*

$$|g| = |g^{-1}|.$$

Proof. Let $g \in G$ and $|g| = n$, so by the definition of order $g^n = e$. Consider

$$(g^{-1})^n = g^{-n} = (g^n)^{-1} = (e)^{-1} = e$$

Thus $|g^{-1}| \leq n$. Now let $|g^{-1}| = k$ where $k < n$ so that $(g^{-1})^k = e$. Notice $(g^k)^{-1} = e = e^{-1}$ which gives us $g^k = e$. Thus $|g| = n \leq k$, which is a contradiction. Therefore $|g^{-1}| = n = |g|$. □

Definition 1.2.5 (Subgroup)**.** Let G be a group and let H be a non-empty set such that $H \subseteq G$. We say that H is a subgroup of G if H is a group.

Definition 1.2.6 (Cyclic subgroup)**.** Let G be a group, g be an element in that group, and $|g| = n$. We say that $\langle g \rangle = \{g, g^2, ..., g^{n-1}, e\}$ is the cyclic subgroup generated by g.

Theorem 1.2.7 (Lagrange's Theorem)**.** *If G is a finite group and H is a subgroup of G, then* $|H|$ *divides* $|G|$.

Corollary 1.2.8. *In a finite group, the order of each element divides the order of the group.*

Proof. Let g be an arbitrary element of a finite group G such that $\langle g \rangle = \{g, g^2, ..., g^{|g|-1}, e\}$, i.e. the set of powers of g, up to $g^{|g|} = e$. Note that $\langle g \rangle$ has an identity, is closed, inherits associativity from G, and each element g^k has some inverse g^m, where $m = n - k$. Therefore $\langle g \rangle$ is a subgroup of G, and by 1.2.7, $\left|\langle g \rangle\right|$ divides $|G|$. But $\langle g \rangle$ contains precisely $|g|$ elements, and so $\left|\langle g \rangle\right| = |g|$ and therefore $|g|$ divides $|G|$. □

Proposition 1.2.9. *Given a finite group G, for all elements* $b \in G$,

$$|b^j| = \frac{|b|}{\gcd(j, |b|)}, \qquad \text{for all nonzero } j \in \mathbb{Z}.$$

Proof. Let b be an arbitrary element of a finite group G, where $|b| = n$. Let $g = \gcd(j, n)$, and consider $(b^j)^{\frac{n}{g}} = (b^n)^{\frac{j}{g}}$, which we can do since g by definition divides both n and j. Further, since $|b| = n$, $(b^j)^{\frac{n}{g}} = (b^n)^{\frac{j}{g}} = e^{\frac{j}{g}} = e$. By 1.2.3, $|b^j| \big| \frac{n}{g}$.

Note $(b^j)^{|b^j|} = e = b^n$. Therefore $b^{j*|b^j|} = e$, and by 1.2.3, $n|j * |b^j|$ and so there exists x such that $nx = j * |b^j|$. If we divide both sides by g, we have $\frac{n}{g} * x = \frac{j}{g} * |b^j|$. Since g divides both n and j, $\frac{n}{g}$ and $\frac{j}{g}$ are both integers. Moreover, since we have divided out all their common factors, $\frac{n}{g}$ is relatively prime to $\frac{j}{g}$. Therefore, by Euclid's Lemma, $\frac{n}{g}\Big||b^j|$.

Since we have that $|b^j|\Big|\frac{n}{g}$ and $\frac{n}{g}\Big||b^j|$, $|b^j| = \frac{|b|}{\gcd(j,|b|)}$ □

Definition 1.2.10 (Ring)**.** A *ring* R is a set with two binary operations, addition and multiplication, such that

1. The set together with the additive operation forms an Abelian Group
2. Multiplication is associative on the set
3. Multiplication distributes over addition

If the multiplicative operation is commutative, the ring is called a *commutative ring*. The multiplicative identity element, if it exists, is called the *unity*. Elements that have multiplicative inverses are called *units*.

Definition 1.2.11 (Field)**.** A *field* is a commutative ring with unity in which every element, other than the additive identity, has a multiplicative inverse. In other words a field forms an Abelian group with addition, the field without the additive identity forms an Abelian group with multiplication, and multiplication distributes over addition.

Many of the concepts of divisibility from Section 1.1.1 can be extended to the setting of polynomial rings.

Theorem 1.2.12 (Division Algorithm in Polynomial Rings)**.** *Consider the set* $\mathbb{Q}[x]$ *of polynomials with rational coefficients. Given* $f(x)$ *and* $h(x)$ *in* $\mathbb{Q}[x]$*, there exist polynomials* $q(x)$ *and* $r(x)$ *in* $\mathbb{Q}[x]$ *such that*

$$f(x) = q(x)h(x) + r(x),$$

and $0 \leq deg(r(x)) < deg(h(x))$.

Proposition 1.2.13. *Let a and b be nonzero elements of a finite field F. The order of the product ab divides* $\operatorname{lcm}\big(|a|,|b|\big)$*, and is divisible by* $\frac{\operatorname{lcm}(|a|,|b|)}{\gcd(|a|,|b|)}$.

Proof. Let a and b be nonzero elements of a finite field F. Let $l = \operatorname{lcm}\big(|a|,|b|\big)$ and consider $(ab)^l$. Since field multiplication is commutative, $(ab)^l = a^l b^l$. By definition of lcm, there exist x and y such that $l = |a|x$ and $l = |b|y$. Therefore by substitution we have $(ab)^l = a^{|a|x}b^{|b|y}$, which equals the identity. Thus $(ab)^l = e$, and so by 1.2.3, $|ab|\Big|l$.

Consider $e = (ab)^{|ab||a|} = \left(a^{|a|}b^{|a|}\right)^{|ab|} = \left(b^{|a|}\right)^{|ab|}$. By 1.2.3, $\left|b^{|a|}\right|$ divides $|ab|$. By 1.2.9, $\left|b^{|a|}\right| = \frac{|b|}{\gcd(|a|,|b|)}$. So $\frac{|b|}{\gcd(|a|,|b|)}$ divides $|ab|$. By repeating the process with $(ab)^{|ab||b|}$, we also have that $\frac{|a|}{\gcd(|a|,|b|)}$ divides $|ab|$. Since we have divided out all common factors, it follows that $\frac{|b|}{\gcd(|a|,|b|)}$ and $\frac{|a|}{\gcd(|a|,|b|)}$ are relatively prime, and so their product must also divide $|ab|$. We can write that product as $|a| * \frac{|b|}{\gcd(|a|,|b|)} * \frac{1}{\gcd(|a|,|b|)}$. But $|a| * \frac{|b|}{\gcd(|a|,|b|)}$ is precisely the $\operatorname{lcm}\left(|a|,|b|\right)$, being the unique parts of both $|a|$ and $|b|$ and only one copy of the factors they share. Thus we have that $\frac{\operatorname{lcm}(|a|,|b|)}{\gcd(|a|,|b|)}\Big||ab|$. □

Proposition 1.2.14. *Given a finite field F, let A be a nonzero element of F^* whose order is greater than or equal to the orders of all other elements in F^*. Then for all $a \in F^*$, the order of a divides $|A|$.*

Proof. Let A be an element of a finite field F^* with maximal order. By way of contradiction, let a be an element of F^* such that $|a|$ does not divide $|A|$. By the the Fundamental Theorem of Arithmetic, there must exist some prime p such that p divides $|a|$ and p does not divide $|A|$. Consider the element $a^{\frac{|a|}{p}}$, which by 1.2.9 has order $\frac{|a|}{\gcd(|a|,|a|/p)}$ which equals $\frac{|a|}{|a|/p}$ which equals p. Thus we have constructed an element $a^{\frac{|a|}{p}}$ which we shall call x, such that $|x| = p$. Note that since p is prime, and p does not divide $|A|$, p and $|A|$ are relatively prime. Consider the element xA, by 1.2.13, $|xA|\Big|\operatorname{lcm}\left(p,|A|\right)$, and $\frac{\operatorname{lcm}(p,|A|)}{\gcd(p,|a|)}\Big||xA|$. Since p and $|A|$ are relatively prime, $\operatorname{lcm}\left(p,|A|\right) = p|A|$ and $\gcd\left(p,|A|\right) = 1$. Thus $|xA|$ divides $p|A|$ and $p|A|$ divides $|xA|$, so $|xA| = p|A|$. $p|A| > |A|$, which is a contradiction since A had maximal order. Therefore for all $a \in F^*$, the order of a divides $|A|$. □

Definition 1.2.15 (Group Homomorphism)**.** A *group homomorphism* f is a mapping between groups which preserves the group operation; i.e. if $f : G \to \bar{G}$ is a group homomorphism from group G to group $\bar{G}$, then $f(a \circ b) = f(a) \cdot f(b) \quad \forall a, b \in G$ where $\circ$ is the group operation on G and $\cdot$ is the group operation on $\bar{G}$

Definition 1.2.16 (Ring Homomorphism)**.** A *ring homomorphism* f is a mapping between rings which preserves the ring operations; i.e. if $f : R \to \bar{R}$ is a ring homomorphism, then $f(a + b) = f(a) + f(b)$ and $f(ab) = f(a)f(b) \quad \forall a, b \in R$.

Definition 1.2.17 (Isomorphism)**.** An *isomorphism* between groups or rings is a homomorphism which is one-to-one and onto.

1.2.2 *A Modicum of Modular Arithmetic*

Definition 1.2.18. Let $m \geq 1$ be an integer. We say that the integers a and b are *congruent modulo* m if their difference $a - b$ is divisible by m. We write

$$a \equiv b \mod m$$

to indicate that a and b are congruent modulo m. The number m is called the modulus.

Two alternate definitions of modular equivalence are as follows:

(1) (Division algorithm definition) The Division Algorithm guarantees that for all integers a, b, m there exist integers q_1, q_2, r_1, r_2 such that $a = q_1 m + r_1$ and $b = q_2 + r_2$. Then $a \equiv b \mod m$ if and only if $r_1 = r_2$.

(2) (Clock definition) $a \equiv b \mod m$ if and only if a and b are the same time on an m hour clock.

Theorem 1.2.19. *Let $m \geq 1$ be an integer.*

(a) If $a_1 \equiv a_2 \mod m$ and $b_1 \equiv b_2 \mod m$, then

$$a_1 \pm b_1 \equiv a_2 \pm b_2 \mod m \qquad \text{and} \qquad a_1 b_1 \equiv a_2 b_2 \mod m$$

(b) Let a be an integer. Then $ab \equiv 1 \mod m$ for some integer b if and only if $\gcd(a, m) = 1$.

Proof. (a) Let $m \geq 1$, a and b be integers. Notice by the definition of modular arithmetic 1.2.18 $a_1 - a_2 = mx$ and $b_1 - b_2 = my$ for some $x, y \in \mathbb{Z}$. Consider

$$(a_1 - a_2) + (b_1 - b_2) = mx + my$$

$$(a_1 + b_1) - (a_2 + b_2) = m(x + y)$$

which implies $a_1 + b_1 \equiv a_2 + b_2 \mod m$. Similarly for $(a_1 - a_2) - (b_1 - b_2) = mx - my$, $a_1 - b_1 \equiv a_2 - b_2 \mod m$. By definition 1.2.18 and algebraic manipulation, we see $a_1 = mx + a_2$ and $b_1 = my + b_2$.

Now consider

$$a_1 b_1 = (mx + a_2)(my + b_2)$$

$$a_1 b_1 - a_2 b_2 = m(mxy + ya_1 + xb_2)$$

which implies $a_1 b_1 \equiv a_2 b_2 \mod m$.

(b) Consider $ab \equiv 1 \mod m$ which by definition of modular arithmetic 1.2.18 gives $mx = ab - 1$. Now consider d where $d|a$ and $d|m$, so $dl = a$ and $dk = m$ for some $l, k \in \mathbb{Z}$. By substituting, we find $(dl)x = (dk)b - 1$ which simplifies to $d(kb - lx) = 1$. Thus $d|1$, and therefore $d = 1$ so $gcd(a, m) = 1$.

Now suppose $gcd(a, m) = 1$. By 1.1.9, $1 = mx + ab$ for some $x, b \in \mathbb{Z}$. Thus $m(-x) = ab - 1$, so $ab \equiv 1 \mod m$.

□

Definition 1.2.20. The ring of integers $\mathbb{Z}/m\mathbb{Z}$ is the set $\{0, 1, 2, \ldots, m-1\}$ together with modular addition and multiplication.

Theorem 1.2.21. *If $g^a \equiv 1 \mod m$ and $g^b \equiv 1 \mod m$, then $g^{\gcd(a,b)} \equiv 1 \mod m$.*

Proof. Let $g^a \equiv 1 \mod m$ and $g^b \equiv 1 \mod m$. We can consider g as an element in the ring $\mathbb{Z}/m\mathbb{Z}$, such that $|g|$ denotes the multiplicative order, and so by 1.2.3, $|g||a$ and $|g||b$. By 1.1.4, it follows that $|g|| \gcd(a, b)$, and so therefore $g^{\gcd(a,b)} \equiv 1 \mod m$. □

Exercise 1.2.22. Suppose $m \equiv 1 \mod b$. What integer between 1 and $m - 1$ is equal to $b^{-1} \mod m$?

Solution. Supposing $m \equiv 1 \mod b$, by 1.2.19 $gcd(m, b) = 1$ and by 1.1.10 there exist u, v such that $um + vb = 1$. Notice that $1 \equiv um + vb \mod m$ gives $1 \equiv vb \mod m$ so $b^{-1} \equiv v \mod m$.

Theorem 1.2.23. *Let p be an odd prime and a be an integer relatively prime to p. If b is the square root of a modulo p^n, then for some choice of j, the number $b + jp^n$ is a square root of a modulo p^{n+1}.*

Proof. Suppose b is the square root of a modulo p^n, i.e. $b^2 \equiv a \mod p^n$. By definition of modular equivalence, there exists x such that $p^n x = b^2 - a$, and so $b^2 - p^n x = a$. Let $j = -2^{-1}b^{-1}x$, where the inverses are modulo p^{n+1}. We know that 2 has an inverse modulo p^{n+1} since p is an odd prime, thus 2 is relatively prime to p^{n+1} and so it has an inverse by 1.2.19. It is clear that b has an inverse modulo p^n, as a being relatively prime to p^n has an inverse, and so $b * (b * a^{-1}) \equiv 1 \mod p^n$. Thus b has an inverse modulo p^n, is therefore relatively prime to p, and so must have an inverse modulo p^{n+1}. Therefore j exists. Consider $(b + jp^n)^2 = b^2 + 2bjp^n + p^{2n}j^2$. We take this modulo p^{n+1}, which is equivalent to $b^2 + 2bjp^n$, as $p^{n+1} \Big| p^{2n}$. By substitution, we have $b^2 + 2b(-2^{-1}b^{-1}x)p^n \equiv b^2 - xp^n \mod p^{n+1}$. We know that $b^2 - xp^n = a$, so clearly $b^2 - xp^n \equiv a \mod p^{n+1}$. Thus $(b + jp^n)^2 \equiv b^2 - xp^n \equiv a \mod p^{n+1}$. □

Remark 1.2.24. If p is an odd prime and if a has a square root modulo p, then a has a square root modulo p^n for all positive powers of p. This is not true if $p = 2$, as can be seen with , $a = 3$. $3 \equiv 1 \mod 2$, and 1 has a a square root, but 3 has no square root modulo 4.

Theorem 1.2.25 (Lagrange's Other Theorem)**.** *Let p be prime, and*

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

be a polynomial of degree $n \geq 1$ with integer coefficients such that not every coefficient is divisible by p. Then $f(x)$ has at most n distinct roots $\mod p^k$.

Proof. The proof that follows is a proof by induction.

First consider the base case $f(x) = a_1 x + a_0$. We show that $f(x)$ has no more than one root. First, suppose $p|a_1$ and r is a root of $f(x)$, $f(r) = a_1(r) + a_0 = 0$. Consider

$$p^n f(r) \equiv 0 \equiv p^n a_1(r) + p^n a_0 \pmod{p^k} \qquad \text{such that} \qquad k < n.$$

Notice $p^k | p^n$ which implies $p^n a_1(r) \equiv 0 \pmod{p^k}$ and $p^n a_0 \equiv 0 \pmod{p^k}$. This in turn implies that $p|a_0$. However, by the statement of our theorem at least one integer coefficient cannot be divisible by p. Thus, $p \nmid a_1$. Now let r_1 and r_2 be roots of $f(x)$ and suppose a_1 is not divisible by p. We see $f(r_1) = a_1 r_1 + a_0 = 0$ and $f(r_2) = a_1 r_2 + a_0 = 0$, which gives us $a_1 r_1 + a_0 = a_1 r_2 + a_0$. So $a_1 r_1 \equiv a_1 r_2 \mod p^k$, and $a_1^{-1} a_1 r_1 \equiv a_1^{-1} a_1 r_2 \pmod{p^k}$ does not change the integer value of the roots. Thus $r_1 \equiv r_2 \mod p^k$, and $f(x)$ has no more than one root $\mod p^k$.

Now suppose $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ has n or fewer roots $\mod p^k$. Note that $f(x) \in \mathbb{Z}[x] \subseteq \mathbb{Q}[x]$.

Consider $f(x) = a_{n+1}x^{n+1} + a_n x^n + \cdots + a_1 x + a_0$. We want to show that $f(x)$ has no more than $n+1$ roots $\mod p^k$. First consider the case where $f(x)$ has no roots $\mod p^k$. Since $0 < n+1$, there are no more than $n+1$ roots. Now consider the second case where $f(x)$ has at least one root $\mod p^k$. By the Division Algorithm of Polynomials 1.2.12, $f(x) = (x - r_0)q(x) + r(x)$ for $0 < deg(r(x)) < deg(x - r_0)$ where $q(x), r(x) \in \mathbb{Q}[x]$. Note that since $deg(r(x)) < 1$, $f(r_0) = 0 + r(r_0) = 0$ implies $r(x) = 0$. Consider $kf(x) = k(x - r_0)q(x)$, where k is the least common multiple of the denominators of the coefficients in $\mathbb{Q}[x]$. We see $kf(x) \in \mathbb{Z}[x]$ and r_0 is a root of $kf(x)$ if and only if r_0 is root of $f(x)$. Now, notice that $kq(x)$ is a polynomial of degree n with integer coefficients. Thus by our induction hypothesis, $kq(x)$ has no more than n roots $\mod p^k$ and it follows that $f(x) = a_{n+1}x^{n+1} + a_n x^n + \cdots + a_1 x + a_0$ has no more than $n+1$ roots $\mod p^k$. □

Remark 1.2.26. If you have studied Abstract Algebra, you will recognize $\mathbb{Z}/m\mathbb{Z}$ as the quotient ring of $\mathbb{Z}$ mod the principle ideal $m\mathbb{Z}$, and that the numbers $0, 1, \ldots, m-1$ are actually coset representatives for the congruence classes that comprise the elements of $\mathbb{Z}/m\mathbb{Z}$.

Definition 1.2.27. The set of numbers having multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$ is denoted $(\mathbb{Z}/m\mathbb{Z})^*$ and called the *group of units modulo m*.

Example 1.2.28. $(\mathbb{Z}/m\mathbb{Z})^*$ forms a group.

Proof. Note that associativity of $(\mathbb{Z}/m\mathbb{Z})^*$ is inherited from the ring properties of $\mathbb{Z}/m\mathbb{Z}$, as is the multiplicative identity 1. Next we show that all elements of $(\mathbb{Z}/m\mathbb{Z})^*$ have multiplicative inverses. Let $a \in \mathbb{Z}/m\mathbb{Z}^*$. By the definition of this set, we know there exists a^{-1} such that $a * a^{-1} = 1$. Finally, we need to show that there is closure. Let $a, b \in \mathbb{Z}/m\mathbb{Z}^*$. By definition 1.2.27 there exists $a^{-1}, b^{-1} \in \mathbb{Z}/m\mathbb{Z}^*$. Notice

$$ab(b^{-1})(a^{-1}) = ab(b^{-1}a^{-1}) = a(b * b^{-1})(a^{-1}) = a(a^{-1}) = 1$$

Thus ab has a multiplicative inverse and in $\mathbb{Z}/m\mathbb{Z}^*$, which implies that $\mathbb{Z}/m\mathbb{Z}^*$ is closed. Therefore $(\mathbb{Z}/m\mathbb{Z})^*$ forms a group. □

Remark 1.2.29. The ring of integers $\mathbb{Z}/m\mathbb{Z}$ forms a field if and only if m is prime. To form a field, all non-zero elements $\{1, m-1\}$ must have an inverse, meaning they are relatively prime to m. This happens precisely when m is prime.

1.2.3 *A Tour of Euler's Totient Function*

Definition 1.2.30. *Euler's phi function* (also called *Euler's totient function*) is the function

$$\phi(m) = \left|(\mathbb{Z}/m\mathbb{Z})^*\right| = \left|\{0 \leq a < m : \gcd(a, m) = 1\}\right|.$$

Example 1.2.31.

a. $(\mathbb{Z}/12\mathbb{Z}) = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. Note that $1, 5, 7, 11$ are relatively prime to 12. So $\phi(12) = 4$

b. For any prime p, $\phi(p) = p - 1$, because all integers less than p are relatively prime to p.

c. Let p be prime and j be a positive integer, $\phi(p^j) = p^j\left(1 - \frac{1}{p}\right)$.

Proof. Let p be a prime. Note that only integers not relatively prime to p^j are a multiple of p:

$$p, 2p, 3p, \ldots, p^{j-1}p$$

Thus there are p^{j-1} multiples of p less than p^j, so the number of integers not relatively prime to p^j is p^{j-1}. Therefore if we take p^j and subtract the number of integers not relatively prime to p^j we are left with the number of integers relatively prime to p^j. So

$$\phi(p^j) = p^j - p^{j-1} = p^j\left(1 - \frac{1}{p}\right).$$

□

Theorem 1.2.32. *Given integers m and n such that* $\gcd(m, n) = 1$,

$$\phi(mn) = \phi(m)\phi(n).$$

Proof. This result is proved in section 3.5.1 □

Theorem 1.2.33. *Let $p_1, p_2, \ldots, p_r$ be the distinct prime factors of some integer N.*

$$\phi(N) = N \prod_{i=1}^{r} \left(1 - \frac{1}{p_i}\right).$$

Proof. Let $p_1^{k_1} p_2^{k_2} ... p_r^{k_r}$ be the unique prime factorization of N. Consider $\phi(N)$, which by 1.2.32 equals $\phi(p_1^{k_1})\phi(p_2^{k_2})...\phi(p_r^{k_r})$. By 1.2.30, it follows that $\phi(N) = (p_1^{k_1} - p_1^{k_1-1})(p_2^{k_2} - p_2^{k_2-2})...(p_r^{k_r} - p_r^{k_r-1})$. We can factor out each of the $p_i^{k_i}$, giving us $\phi(N) = p_1^{k_1} p_2^{k_2} ... p_r^{k_r} * (1 - 1/p_1)(1 - 1/p_2)...(1 - 1/p_r) = N \prod_{i=1}^{r} \left(1 - \frac{1}{p_i}\right)$. □

Theorem 1.2.34 (Euler's Formula)**.** *For all $a \in \mathbb{Z}$ such that* $\gcd(a, N) = 1$,

$$a^{\phi(N)} \equiv 1 \pmod{N}.$$

Proof. Let a be an arbitrary integer and suppose $gcd(a, N) = 1$. Note $a \in (\mathbb{Z}/N\mathbb{Z})^*$ and by the corollary to Lagrange's Theorem 1.2.8, $|a|$ divides $|(\mathbb{Z}/N\mathbb{Z})^*|$. Since $\phi(N) = |(\mathbb{Z}/N\mathbb{Z})^*|$, by the definition of divides 1.1.1, $\phi(N) = |a| * n$ for some $n \in \mathbb{Z}$. Thus, $a^{\phi(N)} = a^{|a|*n} = 1^n$ so $a^{\phi(N)} \equiv 1 \pmod{N}$. □

1.2.4 *A Frenzy of Finite Fields*

Introduced by Galois in 1830 as part of his proof of the unsolvability of the general quintic equation, finite fields are fields having a finite number of elements.

Theorem 1.2.35. *For each prime p and each positive integer n, there exists (up to isomorphism) a unique finite field of order p^n. This field is denoted $\mathbb{F}_{p^n}$ or $GF(p^n)$.*

Remark 1.2.36 (On Finite Fields). One of the unique properties of finite fields, is that there are no finite fields with composite order. Every finite field has prime power order.

Proof. Let F be a finite field such that $|F| = n$ and n is not a prime power. By the Fundamental Theorem of Arithmatic, we know that n can be written as $n = p_1 p_2 \ldots p_k$ where all the p's are primes and, since n is not a prime power, we know that this can be written so that $p_1 \neq p_2$. Let x be an arbitrary element of F^* and let 1 and 0 be the unity and additive identity of F respectively. By Lagrange's Theorem, we know that the additive order of x divides the order of the additive group, which is to say that ${}^+|x| \Big| n$. Therefore we can write

$$\underbrace{x + x + \cdots + x}_{n\text{-times}} = 0$$

Since multiplication distributes over addition, this is equivalent to saying

$$x(\underbrace{1+1+\cdots+1}_{n\text{-times}}) = 0$$

Now, since F is a field we know that x has an inverse in F and that we can multiply both sides of this equation by x^{-1} to get rid of the x on the left side while keeping the right side the same. We can then use our previous factorization of n to write

$$(\underbrace{1+1+\cdots+1}_{p_1\text{-times}})(\underbrace{1+1+\cdots+1}_{p_2\text{-times}})\ldots(\underbrace{1+1+\cdots+1}_{p_k\text{-times}}) = 0$$

This represents the product of k elements in F. Since F^* is closed under multiplication, and since $0 \notin F^*$ we know that one of those k elements is not in F^*. More specifically, we know that there exists some p_m in the prime factorization of n such that

$$\underbrace{1+1+\cdots+1}_{p_m\text{-times}} = 0$$

By Proposition 1.2.3 we know that ${}^{+}|1|\big|p_m$. Since p_m is prime, its only divisors are one and itself. The additive order of 1 cannot be one since the additive identity of a field is unique and only the additive identity can have an order of one. Therefore, ${}^{+}|1| = p_m$. Now, let y be another arbitrary element of F^*. We can write

$$0 = 0y = y(\underbrace{1+1+\cdots+1}_{p_m\text{-times}}) = \underbrace{y+y+\cdots+y}_{p_m\text{-times}} = 0$$

Again, by Proposition 1.2.3 we know that ${}^{+}|y|\big|p_m$ and furthermore that ${}^{+}|y| = p_m$. Since y was an arbitrary element of F^* we now know that every element of F^* has an additive order of p_m. However, since n is not prime power, the exists a prime divisor of n, p_q, such that $p_q \neq p_m$.

Students that have had instruction in Abstract Algebra that are familiar with the Fundamental Theorem of Finite Abelean Groups and its implications will remember that, because the additive group F is a finite abelean group, for every prime divisor of $|F|$ there must be an element in F with an additive order equal to that prime divisor. This means that every element in F having an additive order of p_m when a distinct p_q is also a divisor of F is a contradiction. Therefore, every finite field has prime power order. □

Example 1.2.37. If p is prime, the ring of integers modulo p, $\mathbb{Z}/p\mathbb{Z}$ forms a finite field. Thus, F_p and $\mathbb{Z}/p\mathbb{Z}$ are really two notations for the same thing.

The application of finite fields in cryptography often involves raising elements of $\mathbb{F}_p$ to high powers. Thus, it is worth investigating exponentiation in $\mathbb{F}_p$ from a purely mathematical point of view.

Theorem 1.2.38 (Fermat's Little Theorem)**.** *Let p be a prime number and let a be an integer. Then $p \nmid a$ implies that $a^{p-1} \equiv 1 \mod p$, and $p|a$ implies that $a^{p-1} \equiv 0 \mod p$.*

Proof. Let p be a prime number and let a be an integer. There are two cases:

Case 1: Suppose that $p \nmid a$. Note that this implies that $\gcd(a, p) = 1$, and consider that $a^{\phi(p)} \equiv 1 \mod p$ by Theorem 1.2.34. Since p is prime, we know that $\phi(p) = p - 1$ by Definition 1.2.30, and therefore, by substitution, $a^{\phi(p)} \equiv a^{p-1} \equiv 1 \mod p$.

Case 2: Suppose that $p|a$. Note that by the definition of divides, there must be some integer n such that $pn = a$, and consider $a^{p-1} \mod p$. By substitution, $a^{p-1} \equiv (pn)^{p-1} \mod p$, since pn is a multiple of p we know that $pn \equiv 0 \mod p$. It follows that $(pn)^{p-1} \equiv 0 \mod p$. Therefore if $p|a$, then $a^{p-1} \equiv 0 \mod p$.

Thus $p \nmid a$ implies that $a^{p-1} \equiv 1 \mod p$, and $p|a$ implies that $a^{p-1} \equiv 0 \mod p$. □

Corollary 1.2.39. *For all $a \in \mathbb{F}_p^*$, if $x \equiv y \mod p - 1$ then $a^x \equiv a^y \mod p$.*

Proof. Let a be arbitrary element of the field $\mathbb{F}_p^*$, where p is a prime number, and suppose that $x \equiv y \mod (p-1)$ for some integers x and y. By the definition of modular arithmetic, $(p-1) \mid (x-y)$, and by the definition of divides, there must be some integer n such that $(p-1)n = x - y$.

Now consider $a^{(p-1)n}$. By Fermat's Little Theorem, we know that $a^{(p-1)n} \equiv 1 \mod p$, and so by substituting in $(p-1)n = x - y$, it follows that $a^{(p-1)n} \equiv a^{x-y} \equiv 1 \mod p$. Applying exponent rules we find that $a^{x-y} \equiv a^x a^{-y} \equiv 1 \mod p$. Because $\mathbb{F}_p$ is a field, we can multiply both sides of the equation by $(a^{-y})^{-1} = a^y$ to find that $a^x \equiv a^y \mod p$. Thus we have shown that for all $a \in \mathbb{F}_p^*$, if $x \equiv y \mod (p-1)$, then $a^x \equiv a^y \mod p$. □

The direct converse of Corollary 1.2.39 is false. There is, however, a slight modification that makes the converse statement true. The revised statement is as follows:

Corollary 1.2.40 (The Revised Converse)**.** *For all $a \in \mathbb{F}_p^*$, if $a^x \equiv a^y \mod p$ then $x \equiv y \mod |a|$.*

Proof. Let a be arbitrary element of the field $\mathbb{F}_p^*$, where p is a prime number, and suppose that $a^x \equiv a^y \mod p$ for some integers x and y.

Since inverses are unique and every element of a field has an inverse, this would also imply that $a^{-x} \equiv a^{-y} \mod p$. We can then write

$$1 \equiv a^x a^{-x} \equiv a^x a^{-y} \equiv a^{x-y} \equiv 1 \mod p$$

By Proposition 1.2.3, we find $|a| \,\big|\, (x-y)$, and by applying the definition of modular arithmetic, 1.2.18, it follows that $x \equiv y \mod |a|$. Therefore, for all $a \in \mathbb{F}_p^*$, if $a^x \equiv a^y \mod p$ then $x \equiv y \mod |a|$.

□

Remark 1.2.41. Fermat's Little Theorem and the fast powering algorithm provide a reasonably efficient method of calculating inverses modulo p, namely

$$a^{-1} \equiv a^{p-2} \mod p.$$

Theorem 1.2.42. *Given a finite field $\mathbb{F}_{p^n}$ of prime power order p^n, there exists an element $g \in \mathbb{F}_{p^n}$ such that*

$$\mathbb{F}_{p^n}^* = \{g^x | x \in \mathbb{Z}/(p^n - 1)\mathbb{Z}\}.$$

Such an element g is called a primitive root, *or* generator, *of $\mathbb{F}_{p^n}$.*

Proof. Let A be an element in $\mathbb{F}_{p^n}^*$, such that $|A|$ is maximal, this means that every element other than A has order less than or equal to $|A|$. Consider the polynomial $x^{|A|} - 1 \mod p^n$. Consider some element $a \in \mathbb{F}_{p^n}^*$. Since A has maximal order, Theorem 1.2.14 tells us that $|a| \Big| |A|$ therefore

$$a^{|A|} - 1 \equiv 1 - 1 \equiv 0 \mod p^n$$

Thus, it follows that every element of F^* is a zero of the polynomial and, since F^* has $p^n - 1$ elements, the polynomial has $p^n - 1$ zeros. Since the degree of this polynomial is $|A|$, Lagrange's Other Theorem 1.2.25 tells us that $p^n - 1 \leq |A|$. Also, since the order of A must be less than the order of the group it is in, we know $|A| \leq p^n - 1$. These two inequalities imply that $|A| = p^n - 1$ and that, since $|F^*| = p^n - 1$, A must generate all of F^*. □

1.3 THE FAST POWERING ALGORITHM

In some cryptosystems we will study, for example RSA and Diffie-Hellman cryptosystems, the encryption process involves computing large powers of a number g modulo another number N. The naive way to compute $g^A \mod N$ is by repeated multiplication, i.e.

$$g_1 = g \mod N, \quad g_2 = gg_1 \mod N, \quad g_3 = gg_2 \mod N, \ldots$$

It is clear that $g_A = g^A \mod N$, but if A is large, this method is computationally infeasible. For example, if $A = 2^{1000}$, then this naive approach would take longer than the estimated age of the universe. Clearly we need a better way of computing $g^A \mod N$.

Algorithm 1.3.1 (Fast Powering Algorithm). To compute $g^A \mod N$:

Step 1. Compute the binary expansion of A,

$$A = 2^{i_1} + 2^{i_2} + \ldots + 2^{i_r}$$

Step 2. Compute the powers $g^{2^i} \mod N$ for $0 \leq i \leq i_r$ by successive squaring, i.e. $a_0 \equiv g \mod N$, $a_1 = a_0^2 \mod N, \ldots, a_{i_r} = a_{i_r-1}^2 = g^{2^{i_r}} \mod N$.

Step 3. Compute $g^A \mod N$ using the formula $g^A = g^{2^{i_1}} g^{2^{i_2}} \ldots g^{2^{i_r}} \mod N$.

Example 1.3.2. We will use the fast powering algorithm to compute $2^{477} \mod 1000$. In this example we will let $A = 477$, $g = 2$, and $N = 1000$.

Step 1. To use the fast powering algorithm to compute 2^{477}, we must first find the binomial expansion of 477. We start by determining the largest power of 2 less than 477. Notice that $2^9 = 512$ and $2^8 = 256$, so the first term of the binomial expansion will be 2^8. Now we compute $477 - 256 = 221$, and find that the largest power of 2 less than 221 is 7, so the first two terms of the binomial expansion are $2^8 + 2^7$. If we continue this pattern we find that $477 = 2^8 + 2^7 + 2^6 + 2^4 + 2^3 + 2^2 + 2^0$.

Step 2. We now will compute the powers $g^{2^i} \mod N$.

$$\begin{aligned}
a_0 &\equiv 2 \mod 1000 \\
a_1 \equiv (a_0)^2 &\equiv 2^2 \equiv 4 \mod 1000 \\
a_2 \equiv (a_1)^2 &\equiv 4^2 \equiv 16 \mod 1000 \\
a_3 \equiv (a_2)^2 &\equiv 16^2 \equiv 256 \mod 1000 \\
a_4 \equiv (a_3)^2 &\equiv 256^2 \equiv 536 \mod 1000 \\
a_5 \equiv (a_4)^2 &\equiv 536^2 \equiv 296 \mod 1000 \\
a_6 \equiv (a_5)^2 &\equiv 296^2 \equiv 616 \mod 1000 \\
a_7 \equiv (a_6)^2 &\equiv 616^2 \equiv 456 \mod 1000 \\
a_8 \equiv (a_7)^2 &\equiv 456^2 \equiv 936 \mod 1000
\end{aligned}$$

Step 3. Now we use the fact that $a_i \equiv 2^{2^i} \mod 1000$ to plug in the relevant terms:

$$\begin{aligned}
2^{477} &\equiv a_8 a_7 a_6 a_4 a_3 a_2 a_0 \mod 1000 \\
&\equiv (2^{2^8})(2^{2^7})(2^{2^6})(2^{2^4})(2^{2^3})(2^{2^2})(2^{2^0}) \mod 1000 \\
&\equiv (936)(456)(616)(536)(256)(16)(2) \mod 1000 \\
&\equiv 272 \mod 1000
\end{aligned}$$

Therefore $2^{477} \equiv 272 \mod 1000$.

2

SYMMETRIC KEY ENCRYPTION

For almost as long as humans have had secrets to keep, there have been cryptographic means of keeping those secrets secret. References to encryption schemes can be found throughout history. Julius Caesar used a simple substitution cipher to communicate with his generals. Mary Queen of Scots was beheaded after her slightly more complex substitution cipher was broken, and her duplicity in a plot to kill the queen of England, revealed. The Kama Sutra lists cryptography as one of the skills men and women should posses. In WWII, American forces used a cryptosystem combining the Navajo language and a set of codewords, similar to the system used by Zimmermann. Today we rely on encryption for many things, including the security of our bank records when accessing accounts via an ATM. Although the encryption methods used in the preceding examples vary strikingly in security, complexity and finesse, they are all examples of symmetric key encryption.

2.1 ABSTRACT MATHEMATICAL FORMULATION

In this section we formulate the notion of a cryptosystem in abstract mathematical terms. There are many reasons why this is desirable. In particular, it allows us to highlight similarities and differences between different systems, while also providing a framework within which we can rigorously analyze the security of a cryptosystem against various attacks.

In symmetric key encryption, communicating parties must agree upon and share a common secret key. This key determines both the encryption and decryption algorithms, and hence the security of the scheme relies on this key being kept secret. Following conventions of cryptology, let Alice and Bob be our communicating parties. Because Bob and Alice both know the secret key, they have equal (or symmetric) knowledge and abilities. Mathematically, a symmetric key cipher uses a key k chosen from a space of possible keys $\mathcal{K}$, to encrypt a plaintext message m chosen from the space of possible plaintext messages $\mathcal{M}$, and the result is a ciphertext c belonging to the space of possible ciphertexts $\mathcal{C}$. Thus, encryption can be viewed as a function

$$e : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$$

whose domain $\mathcal{K} \times \mathcal{M}$ is the set of pairs (k, m) consisting of a key k and a plaintext message m and whose range is the space of ciphertexts $\mathcal{C}$. Similarly, decryption is a function

$$d : \mathcal{K} \times \mathcal{C} \to \mathcal{M},$$

satisfying the property

$$d(k, e(k, m)) = m \qquad \text{for all } m \in \mathcal{M}.$$

It is sometimes convenient to write the dependence on k as a subscript. Then for each $k \in \mathcal{K}$ we get a pair of functions

$$e_k : \mathcal{M} \to \mathcal{C} \qquad \text{and} \qquad d_k : \mathcal{C} \to \mathcal{M},$$

satisfying the property

$$d_k(e_k(m)) = m \qquad \text{for all } m \in \mathcal{M}.$$

Remark 2.1.1. In order for an encryption function to be valid, it must be shown to have a well-defined decryption function. This is equivalent to showing that for any two messages $m_1, m_2 \in M$, if $e(m_1) = e(m_2)$, then it must be that $m_1 = m_2$. In other words, we want to avoid two different messages being encoded as the same cipher text.

Exercise 2.1.2. Let N be a large integer and let $\mathcal{K} = \mathcal{M} = \mathcal{C} = \mathbb{Z}/N\mathbb{Z}$. For each of the functions

$$e : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$$

listed in (a), (b), and (c), determine if the encryption function is valid. If so, find its associated decryption function and show that it works. If not, is there a way to turn it into one by restricting the set of keys to some smaller, yet reasonably large, subset?

(a) $e_k(m) = k - m \mod N$

To see if the encryption function is valid, suppose that for some messages m_1 and m_2, we find that $e_k(m_1) \equiv e_k(m_2) \mod N$. From the definition of our function, this means that $k - m_1 \equiv k - m_2 \mod N$, and by the definition of modular arithmetic, we have that $N \big| \big((k - m_1) - (k - m_2)\big)$. The $k's$ cancel, which leaves us with $N|(m_2 - m_1)$. Using the definition of modular arithmetic again, we find that $m_2 \equiv m_1 \mod N$. Therefore the encryption function is valid.

Now we can define a decryption function. Consider the function defined by $d_k(c) \equiv k - c \mod N$. It is clear that for some arbitrary message m, we can perform $d\big(e_k(m)\big) \equiv d_k(k - m) \equiv k - (k - m) \equiv m \mod N$, which shows that the decryption function works.

(b) $e_k(m) = km \mod N$

Let's take the same approach with our new function. Suppose that for some messages m_1 and m_2, $e_k(m_1) \equiv e_k(m_2) \mod N$. Through similar reasoning, we find

that $N|k(m_1 - m_2)$, but this time, we have reached a dead end. There is no guarantee that these messages are the same. Indeed, if we look for a counterexample, we can choose $N = 1000$, $k = 100$, $m_1 = 10$, and $m_2 = 20$, and it is clear that although $m_1 \not\equiv m_2 \mod 1000$, we find that $e_{100}(10) \equiv 100 \cdot 10 \equiv 0 \equiv 100 \cdot 20 \equiv e_{100}(20) \mod 1000$.

You may have noticed however, that if we could simply argue that $N|(m_1 - m_2)$, then we would have our desired result. So are there certain values of k that will guarantee this? Yes, there are! By Theorem 1.1.13, if $\gcd(k, N) = 1$, then because we already have that $N|k(m_1 - m_2)$, N must divide $m_1 - m_2$, which leads directly to $m_1 \equiv m_2 \mod N$. So when k and N are relatively prime, the encryption function is valid!

We can also define a decryption function for our new encryption function by $d_k(c) \equiv k^{-1}c \mod N$. It is no coincidence that the only time that k^{-1} is guaranteed to exist is when k and N are relatively prime. We can see that this function works by considering $d_k(e_k(m)) \equiv d_k(km) \equiv k^{-1}km \equiv m \mod N$ for some arbitrary message m.

(c) $e_k(m) = (k + m)^2 \mod N$

Once again, suppose that for some messages m_1 and m_2, we find that $e_k(m_1) \equiv e_k(m_2) \mod N$. With our new function, this implies that $N\big|\big((k+m_1)^2 - (k+m_2)^2\big)$. Using the property that $a^2 - b^2 = (a + b)(a - b)$, we find that N divides $(k + m_1)^2 - (k + m_2)^2 = (k + m_1 + k + m_2)(k + m_1 - k - m_2) = (2k + m_1 + m_2)(m_1 - m_2)$. Initially, this looks slightly promising, because of the $m_1 - m_2$ term, however this time, we need to show that N is relatively prime to $2k + m_1 + m_2$. Since this is dependent on both messages, in addition to k, it is hard to imagine a way to only restrict k to make this function work.

To be sure, we can construct a counterexample. Let $N = 1001$, $k = 250$, $m_1 = 250$, and $m_2 = 251$, and it is clear that although $m_1 \not\equiv m_2 \mod 1001$, we find that $e_{250}(250) \equiv (250 + 250)^2 \equiv 751 \equiv (250 + 251)^2 \equiv e_{250}(251) \mod 1001$.

2.2 KERCKHOFF'S PRINCIPLE

If the world were want of adversaries, there would be no need for cryptosystems. Luckily for us, spies abound and the need for secure communication exists. Suppose Eve is just such an adversary of Alice and Bob. It is safest for Alice and Bob to assume that Eve knows everything about the encryption scheme being employed except for the choice of secret key k. This is known as *Kerckhoff's Principle*, which says that the security of a cryptosystem should depend only on the secrecy of the key, and not on the secrecy of the encryption algorithm itself.

Definition 2.2.1. [Desirable Properties of a Cryptosystem] If $(\mathcal{K}, \mathcal{M}, \mathcal{C}, e, d)$ is to be a successful cipher, it must have the following properties.

1. For any key $k \in \mathcal{K}$ and plaintext $m \in \mathcal{M}$, it must be easy to calculate $e_k(m)$
2. For any $k \in \mathcal{K}$ and $c \in \mathcal{C}$, it must be easy to calculate $d_k(c)$
3. Given one or more ciphertexts $c_1, c_2, c_3, \ldots, c_n \in \mathcal{C}$ encrypted using $k \in \mathcal{K}$, it must be difficult to compute any of the plaintexts $d_k(c_1), d_k(c_2), d_k(c_3), \ldots, d_k(c_n)$ without knowledge of k
4. Given one or more pairs of plaintexts and their corresponding ciphertexts, $(m_1, c_1), (m_2, c_2), (m_3, c_3), \ldots, (m_n, c_n)$, it must be difficult to decrypt any ciphertext c not in the list without knowledge of k. This is known as security against a *chosen plaintext attack*.

In our list of desirable properties we have left open the question of what is meant by the words ″easy" and ″hard." We informally take ″easy" to mean computable in less than a second on a typical desktop computer and ″hard" to mean that all of the computing power in the world would take several years (at least) to perform the computation.

2.3 EXAMPLES OF SYMMETRIC KEY CIPHERS

It is convenient to view plaintexts and ciphertexts as numbers, often written in binary form. An easy way to do this is to take binary strings of eight bits which represent numbers from 0 to 255. We can then assign each letter of the alphabet a number between 0 and 255 and represent each letter by the corresponding binary string. The ASCII code on your computer does something much like this. This is an example of an encoding scheme.

Definition 2.3.1. An *encoding scheme* is a method of converting one sort of data into another sort of data, for example, converting text into numbers. The distinction between an encoding scheme and an encryption scheme is one of intent. An encoding scheme is assumed to be public knowledge is employed for convenience, not for secrecy.

2.3.1 *Affine Cipher*

Shift Cipher

Let p be a large prime. Alice and Bob take their key, plaintext, and cipher spaces to all be the group of units in the finite field of order p, i.e.

$$\mathcal{K} = \mathcal{M} = \mathcal{C} = \mathbb{F}_p^*.$$

Alice and Bob randomly select a key $k \in \mathcal{K}$, i.e. an integer satisfying $1 \leq k < p$, and use the encryption function

$$e_k(m) = m + k \mod p \qquad \text{for all } m \in \mathcal{M}.$$

The corresponding decryption function is defined as, $d_k(c) = c - k \mod p$. To show that the function works, we plug in a encrypted message and see if when we decrypt it, the function returns the original plaintext message. Consider $d_k(e_k(m))$, notice that

$$d_k(e_k(m)) = (m + k) - k = m + k - k = m \mod p$$

Since the function returns the original message, the decryption function works. This is known as a *shift cipher* since each element of the plaintext is "shifted" by the same amount to get the ciphertext. Julius Caesar used a shift cipher to communicate with his generals.

Multiplication-Modulo-p Cipher

A slight variation of the shift cipher is the *multiplication-modulo-p* cipher. In this cipher, Alice and Bob again select a key from $\mathbb{F}_p^*$, and use the encryption function

$$e_k(m) = mk \mod p.$$

Since $k \in \mathbb{F}_p^*$, it follows that there exists $k^{-1} \in \mathbb{F}_p^*$. We use k^{-1} to construct a decryption function. The decryption function is defined as $d_k(c) = ck^{-1} \mod p$. We can show that the function works by the same reasoning used for the shift cipher. Consider $d_k(e_k(m))$, notice

$$d_k(e_k(m)) = (mk)k^{-1} = mkk^{-1} = m \mod p$$

Because the function returns the original message, we know that the decryption works.

The question now, is are these ciphers any good? To judge this, we will look at how they stack up against the desirable properties of a cryptosystem defined in Definition 2.2.1. Both ciphers satisfy the first two properties. Computers are more than capable of adding and multiplying integers, which is all that is required to encrypt and decrypt these ciphers. The multiplication-modulo-p cipher technically requires finding k^{-1} in order to decrypt the message, but if you already have k, then you can just use the Extended Euclidean Algorithm.

Properties 3 and 4 are a different story. Both are vulnerable to pattern analysis. Because of the regularity in how the original message is altered, it is very easy to find patterns that give away the encryption. For example, if all you had was an encrypted message, you could make educated guesses by assuming that the most frequent number is the most common letter, e. Through a little trial and error, you could break the entire

cipher. For computers, this is no problem. This will be explored further in Example 2.3.2.

And it gets worse if you have access to encrypted and decrypted pairs. For the shift cipher, if you had access to some original message m, and its encrypted pair $e_k(m)$, you could find the value of k by performing $e_k(m) - m = k + m - m = k \mod p$. From there you could decrypt any message encrypted with that key. Similarly, for the multiplication-modulo-p cipher, you can calculate $m^{-1}e_k(m) = m^{-1}mk = k \mod p$ to learn the value of k. As a result, neither of these are considered particularly strong ciphers.

Affine Cipher

The *affine cipher* is a combination of the shift and multiplication-modulo-p ciphers. The key for an affine cipher consists of two integers $k = (k_1, k_2) \in \mathbb{F}_p^* \times \mathbb{F}_p$. The encryption and decryption functions are defined as follows:

$$e_k(m) = k_1 m + k_2 \mod p, \qquad \text{and}$$

We define the decryption function, once again using inverses.

$$d_k(c) = (c - k_2) * k_1^{-1}$$

Using the same strategy as before, we ensure that the decryption function works by considering $d_k(e_k(m))$.

$$d_k(e_k(m)) = ((k_1 m + k_2) - k_2) * k_1^{-1} = k_1 m k_1^{-1} + k_2 k_1^{-1} - k_2 k_1^{-1} = m * 1 - 0 = m$$

So how does the affine cipher stack up against the desirable properties listed in Definition 2.2.1? The answer is not much better than the shift and multiplication-modulo-p ciphers. Again, properties 1 and 2 are easily satisfied, as all that is required is the addition and multiplication of integers, which is easy for computers.

Also like the previous ciphers, it cannot uphold properties 3 and 4. The affine cipher is just as vulnerable to brute force attacks through pattern recognition. And while it holds up slightly better to property 4, it still fails. For the affine cipher, one pair encrypted/decrypted pair is not enough. But two is. If you have access to the pairs (m_1, c_1) and (m_2, c_2), you can calculate $c_1 - c_2 = k_1 m_1 + k_2 - k_1 m_2 - k_2 = k_1(m_1 - m_2) \mod p$ and solve for k_1 to get $k_1 = (c_1 - c_2)(m_1 - m_2)^{-1} \mod p$. From there you can calculate k_2 the way you would a shift cipher: $k_2 = c_1 - k_1 m_1 \mod p$. So this is not a particularly strong cipher either.

2.3.2 *Cryptanalysis of Simple Substitution Ciphers*

The shift, multiplication-modulo-p, and affine ciphers are all examples of *simple substitution ciphers*. A simple substitution cipher is one in which each letter is replaced by another letter or symbol. Despite the large number of possible simple substitution ciphers, they are quite easy to break. This is because so much is known about the patterns in the English language. Frequency tables of letter usage, common double letters and letter pairings are also well known. This allows a code breaker to make educated guesses as to what some of the encrypted letters are. Since the ciphers explained above can all be cracked with just a couple of known plain text-cipher text pairs, substitution ciphers are quite easy to crack.

Example 2.3.2. A simple substitution cipher to cryptanalyze:

16	5	5	30	27	19	24	14	22	7	25	5	2	2	20	1	5	5	30	26
7	25	5	2	28	10	21	18	16	5	11	21	18	5	14	19	28	15	1	16
21	16	27	19	21	11	26	27	19	28	5	28	10	21	7	27	19	27	11	10
21	4	23	7	14	16	16	19	27	24	10	28	5	14	28	11	27	4	21	9
10	5	17	27	19	11	28	15	25	25	20	2	21	15	11	14	25	21	2	20
22	15	18	30	21	4	7	21	21	16	27	19	24	11	11	18	15	28	28	21
25	26	15	11	28	10	5	14	24	10	15	1	5	8	14	21	28	5	7	17
27	16	4	7	16	5	17	21	25	11	17	21	25	21	14	19	28	27	21	4

To establish a foothold on this problem, we need to know what modulus we are working in. Notice that the highest number that appears in this substitution cipher is 30. It is, therefore, a good guess to assume the prime we are working with is 31, so $p = 31$. Next, we want to know the frequency of each letter in the cipher. Notice the number 21; it appears 18 times. There are 160 characters in total, so the frequency of 21 is $\frac{18}{160} = 11.875\%$. We know E is the most commonly used letter in the English language, with a frequency of 13.11%. It is a fairly safe guess to assume that in this case $E = 21$. We now need to find another letter. The next two frequent in the cipher are 5 and 28, with frequencies of 9.375% and 7.5%, respectively. The next 3 most common letters are T, A, O, with frequencies of 10.47%, 8.15%, and 8.00%. Notice that the pair $5, 5$ appears several times in the message, given that O and T are common double letters, it is likely that $5 = T$ or O. If we try $T = 5$, we find out the the cipher produces nonsense, and so we can eliminate the guess that $T = 5$. Lets try 28.

If we let $T = 28$, we can now try break the cipher once again. Lets assume this was encrypted using an affine cipher. This assumption also would hold strong if a shift or multiplication-modulo-p encryption method had been used as well since both ciphers are affine ciphers with $k_1 = 1$ and $k_2 = 0$ respectively. The general form of an affine cipher is

$$c \equiv k_1 m - k_2 \mod p$$

Table 1.: Frequency Table for Example 2.3.2

Value	Appearance	Frequency
1	3	$\frac{3}{160} = 1.875\%$
2	5	$\frac{5}{160} = 3.125\%$
3	0	0
4	5	$\frac{5}{160} = 3.125\%$
5	15	$\frac{15}{160} = 9.375\%$
6	0	0
7	7	$\frac{7}{160} = 4.375\%$
8	1	$\frac{1}{160} = 0.625\%$
9	1	$\frac{1}{160} = 0.625\%$
10	6	$\frac{6}{160} = 3.75\%$
11	10	$\frac{10}{160} = 6.25\%$
12	0	0
13	0	0
14	8	$\frac{8}{160} = 5.0\%$
15	7	$\frac{7}{160} = 4.375\%$
16	9	$\frac{9}{160} = 5.625\%$
17	4	$\frac{4}{160} = 2.5\%$
18	4	$\frac{4}{160} = 2.5\%$
19	10	$\frac{10}{160} = 6.25\%$
20	3	$\frac{3}{160} = 1.875\%$
21	18	$\frac{18}{160} = 11.875\%$
22	2	$\frac{2}{160} = 1.25\%$
23	1	$\frac{1}{160} = 0.625\%$
24	4	$\frac{4}{160} = 2.5\%$
25	8	$\frac{8}{160} = 5.0\%$
26	3	$\frac{3}{160} = 1.875\%$
27	11	$\frac{11}{160} = 6.875\%$
28	12	$\frac{12}{160} = 7.5\%$
29	0	0
30	3	$\frac{3}{160} = 1.875\%$

Notice the letter E is the 5^{th} letter in the alphabet, and T is the 20^{th}, this means we now the corresponding plaintext values, for T and E. We now how two cipher-plain text pairs and this allows us to create a cystem of equations. For E we have

$$21 \equiv (k_1 * 5 + k_2) \mod 31$$

And for T

$$28 \equiv (k_1 * 20 + k_2)$$

We now have a system of two equations. Starting with the equation for E, we can solve for k_2 in terms of k_1

$$21 - k_1 * 5 \equiv k_2 \mod 31$$

Substituting this value into the equation for T, we find

$$28 \equiv (k_1 * 20 + (21 - k_1 * 5)) \mod 31$$

which reduces to

$$28 \equiv k_1 * 15 + 21 \mod 31$$

simplifying we find

$$7 * 15^{-1} \equiv k_1 \mod 31$$

So we just need to find the inverse of 15 mod 31. Using the extended Euclidean algorithm, we find that $15^{-1} \equiv 29 \mod 31$, and $29 * 7 \equiv 17 \mod 31$. Thus we have found or first key, $k_1 = 17$. To find the second key we only need plug in k_1 and solve for k_2. Thus the equation for E becomes

$$21 \equiv 17 * 5 - k_2 \mod 31$$

Which reduces to

$$k_2 \equiv -2 \equiv 29 \mod 31$$

Thus $k_2 = 29$. Putting our two keys together we produce the following function

$$e_k(m) = 17 * m + 29 \mod 31$$

With the corresponding decoding function

$$d_k(c) = (c - 29) * 11 \mod 31$$

Where 11 is the inverse of 17 mod 31. Using this function we can decrypt the ciphertext to reveal the following message:

L	O	O	K	I	N	G	U	P	F	R	O	M	M	Y	B	O	O	K	26
F	R	O	M	T	H	E	C	L	O	S	E	C	O	U	N	T	A	B	L
E	L	I	N	E	S	26	I	N	T	O	T	H	E	F	I	N	I	S	H
E	D	23	F	U	L	L	N	I	G	H	T	O	U	T	S	I	D	E	9
H	O	W	I	N	S	T	A	R	R	Y	M	E	A	S	U	R	E	M	Y
P	A	C	K	E	D	F	E	E	L	I	N	G	S	S	C	A	T	T	E
R	26	A	S	T	H	O	U	G	H	A	B	O	Q	U	E	T	O	F	W
I	L	D	F	L	O	W	E	R	S	W	E	R	E	U	N	T	I	E	D

Realizing that 26, 23, and 9 represent some kind of punctuation, it is not hard to discover that the final message is[1]

> Looking up from my book, from the close countable lines,
> into the finished-full night outside:
> how in starry measure my packed feelings scatter,
> as though a boquet of wildflowers
> were untied

Exercise 2.3.3. Bob and Alice use a cryptosystem in which their private key is a (large) prime k and their plaintexts and ciphertexts are integers. Bob encrypts a message m by computing the product $c = km$. Eve intercepts the following two ciphertexts:

$$c_1 = 12849217045006222, \qquad c_2 = 6485880443666222.$$

Find Bob and Alice's secret key.

Solution: We know Alice and Bob are using a cipher of the form $c = km \mod p$. We want to find the key, k, that Bob and Alice are using. We know $k|c_1$ and $k|c_2$, and by theorem 1.1.4 it follows that $k|\gcd(c_1, c_2)$. Naturally, our first step is to find the $\gcd(c_1, c_2)$, using the Euclidean algorithm.

Step 1 Finding the $\gcd(c_1, c_2)$, via the Euclidean algorithm. Let $r_o = c_1$, and $r_1 = c_2$.

$$\begin{aligned}
r_o &= 1 * r_1 + 6363336601340000 \\
r_1 &= 1 * r_2 + 12254384232622 \\
r_2 &= 51 * r_3 + 133600642702678 \\
r_3 &= 1 * r_4 + 8943199623544 \\
r_4 &= 12 * r_5 + 6282247220150 \\
r_5 &= 1 * r_6 + 2660952403394 \\
r_6 &= 2 * r_7 + 960342413362 \\
r_7 &= 2 * r_8 + 740267576670 \\
r_8 &= 1 * r_9 + 220074836692 \\
r_9 &= 3 * r_{10} + 80043066594 \\
r_{10} &= 2 * r_{11} + 59988703504 \\
r_{11} &= 1 * r_{12} + 20054363090 \\
r_{12} &= 2 * r_{13} + 19879977324 \\
r_{13} &= 1 * r_{14} + 174385766 \\
r_{14} &= 114 * r_{15} + 0
\end{aligned}$$

1 This is a slightly paraphrased and misspelled version of a poem by Rainer Maria Rilke, written in Paris in 1914. The paraphrasing and misspellings were just to make the cipher a nice rectangle.

Step 2 Finding the key. From the euclidean algorithm, the $\gcd(c_1, c_2) = 174385766$. Since k is relatively prime to 2, $k \Big| \frac{174385766}{2} = 87192883$. By a primality test, we know that this number is prime, and thus $k = 87192883$. And sadly Alice and Bob's encryption is longer safe from prying eyes.

2.4 A HISTORY OF SYMMETRIC KEY CIPHERS (BY JAIME MCCONACHIE)

Once upon a time not so very long ago, cryptology was a game of luck and pattern recognition versus complication. With the introduction of algebra in the construction of cryptosystems, cryptology became markedly more abstract and ciphers became easier to secure. This history of cryptology is very similar to what biologists might call an evolutionary arms race. As popular science author Simon Singh prefaces in his book:

> The development of codes can be viewed as an evolutionary struggle. A code is constantly under attack from codebreakers. When the codebreakers have developed a new weapon that reveals a code's weakness, then the code is no longer useful. It either becomes extinct or it evolves... (*The Code Book* xiii) [2]

This history of cryptology will be presented as the struggle between cryptographers and cryptanalysts, the code makers and code breakers. The narrative ends with the code makers having the upper hand, so before you continue reading, I ask you a crucially important question: do you have a good eraser?

2.4.1 *Early History*

The primary cipher in use in the early years of cryptology was the shift cipher, also known as monoalphabetic substitution (Section 2.3.1). It was easy to use (especially prior to the advent of encryption machines.) But, as we saw in Example 2.3.2, the shift cipher is also easy to break. Accomplished polymath[2] Abu Yusuf Ya-qub ibn Isaq as-Sabbah al Kindi (801-873) of 9th cent. CE Abbasid Caliphate realized two crucial properties in cryptanalysis of monoalphabetic substitution. First, he realized language has an inherent structure that can be quantified. Additionally, shift ciphers preserve that structure. In al Kindi's words:

> One way to solve an encrypted message, if we know its language, is to find a cleartext of the same language long enough to fill one sheet or so and then we count each letter of it. We call the most frequently occuring letter the "first", the next most occuring the "second", the following most occuring the "third" and so on, until we finish all the different letters in the cleartext. Then we

2 and author of 290 texts on subjects ranging from mathematics to astronomy to optics and beyond

> look at the cryptogram we want to solve and we also classify its symbols... until we account for all symbols of the cryptogram we want to solve. [4]

Al Kindi's technique was further refined by Baghdad educated mathematician Ibn Adlan (1127-1268), who determined that the ciphertext should be at least 90 characters long with each letter of the plaintext language represented three times for sufficient frequency analysis.

Definition 2.4.1 (Frequency Analysis)**.** Suppose the language of a ciphertext and the frequency of occurrence of letters of that language are known. Frequency counts of the ciphertext letters can be used to guess corresponding plaintext letters (and possibly decrypt the ciphertext).

Despite the development of frequency analysis, monoalphabetic substitution remained in use for some time. This happened for a couple of reasons. First, monoalphabetic substitution was simply very easy to use. Secondly, Europe was a little behind the times (the Dark Ages does that to a region). In about 1300 the Renaissance kickstarted cryptography into an industry. To combat the increased use of cryptography by foreign diplomats, courts employed cipher secretaries and cryptanalytic techniques developed four centuries previously were 're-discovered' in the West. But the increased traffic, and success of frequency analysis, motivated the pursuit of more secure cipher systems, and eventually cryptographers hit on polyalphabetic substitution [2]. To confuse frequency analysis of individual characters, polyalphabetic substitution introduced multiple encryption alphabets. These multiple encryption alphabets were determined by a keyword, mainly for ease of use. As we will see later, the addition of a keyword compromises the security of polyalphabetic substitution.[3]

Polyalphabetic substitution was developed over about a century by an unwitting collaboration of mathematicians. Sometime in the 1460's, Florentine polymath Leon Battista Alberti recognized the potential of using multiple encryption alphabets for confusing frequency analysis. In 1563, Italian scientist Giombattista della Porta introduced the idea of using a keyword to determine those multiple encryption alphabets to make the system more user friendly. Finally, in 1586, French diplomat Blaise de Vigenere compiled the information, polished it up a bit, and got credit for inventing the polyalphabetic substitution cipher (which is now commonly referred to as the Vigenere cipher). Polyalphabetic substitution was such a successful security upgrade that the system was considered unbreakable for about 300 years.[4]

All good things come to an end, however. A huge flaw in the Vigenere cipher was that if the key was shorter than the plaintext, the encryption alphabets were used cyclically.

3 Kahn, David. *Kahn on Codes: Secrets of the New Cryptology*. Macmillan, 1983.

4 Dooley, John. *A Brief History of Cryptology and Cryptographic Algorithms.* Springer, 2013.

Any pattern that provided a foothold for cryptanalysis was eagerly sussed out, and in 1854 Englishman Charles Babbage found that foothold. He was challenged by a dentist from Bristol to decrypt ciphertext encrypted using a Vigenere cipher. Babbage noticed repetitive letter sequences that were likely produced from using the same encryption alphabets, and could be used to guess at the length of the keyword. Much as Leibniz did to Newton, Friedrich Wilhelm Kasiski independently discovered and published the same method in 1863 [2].

Definition 2.4.2 (Kasiski Test 1863)**.** If a string of characters appears repeatedly in a polyalphabetic ciphertext message, it's possible (not certain) that the distance between occurrences is a multiple of the length of the keyword.

During WWI, William Friedman (father of the NSA and husband of the brilliant Elizebeth Friedman) needed more precision than the Kasiski Test offers. The Kasiski Test, though useful, does not result in the actual length of the keyword, nor does it help distinguish between a mono- and polyalphabetic cipher. But suppose we recall al Kindi's observation on the inherent structure of languages. Comparing the probability of two randomly selected letters from a n-letter text being the same (call it the Index of Coincidence) to the Index of Coincidence (IC) of the English language can tell us about the structure of a text. For example, if $IC_{\text{ciphertext}} \approx IC_{\text{English}}$, this suggests that the ciphertext is monoalphabetic [5].

First we let n_i denote the number of ith letters of the alphabet in the n-letter text. For example a is the $1st$ letter of the English alphabet, so n_1 would denote the number of a's in the n-letter text. We see there are

$$\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$$

ways to choose two letters at random from our n-letter text. Similarly, there are

$$\binom{n_i}{2} = \frac{n_i!}{2!(n_i-2)!} = \frac{n_i(n_i-1)}{2}$$

ways to choose two letters at random from the ith letters of our text (of which there are n_i). Thus, the probability of choosing two of the ith letters at random from our n letter text is

$$\frac{n_i(n_i-1)}{n(n-1)}$$

Supposing our n-letter text is written using a 26 character alphabet, the probability that two randomly chosen letters are the same is simply the sum of the probabilities of choosing any one of the letters

$$IC = \sum_{i=1}^{26} \frac{n_i(n_i-1)}{n(n-1)}$$

In texts where n is very large, we can assume that

$$IC \approx \sum_{i=1}^{26} \left(\frac{n_i}{n}\right)^2$$

For a standard English language text (with a large n) we can use the standard relative frequencies of the letters to calculate $IC_{\text{English}} \approx 0.065$ [5]. Recall that ciphertexts with a similar IC to a typical English language text are usually monoalphabetic. A text with an equal distribution of letter frequencies (where each of the 26 letters appears with equal frequency, i.e. $\frac{1}{26}$) has an IC closer to 0.038 [5]. Polyalphabetic substitution typically results in a ciphertext with a more equal distribution of letter frequencies and a cryptanalyst would expect the IC of a polyalphabetic ciphertext to be less than 0.065 [5].

We can use this information to re-derive the Index of Coincidence for use on polyalphabetic ciphertexts to infer the probable length of the keyword. Consider a n-letter polyalphabetic ciphertext that was encrypted with a keyword of length r. Suppose we arrange our ciphertext in a table, as below,

$$\begin{array}{c|c|c|c} c_1 & c_2 & \dots & c_r \\ c_{r+1} & c_{r+2} & \dots & c_{2r} \\ \vdots & \vdots & \vdots & \vdots \end{array}$$

Since our ciphertext was encrypted using a polyalphabetic cipher, we know that each column (encrypted with the same encryption alphabet) is a monoalphabetic substitution cipher. We can choose two letters at random from the same column where first we choose a letter from our text

$$\binom{n}{1}$$

and then a letter from the same column

$$\binom{\frac{n}{r}}{1}$$

Without regard for order, we see we are able to choose two letters from one column

$$(0.065)\frac{n(\frac{n}{r}-1)}{2}$$

ways. Similarly, if we were to choose two letters at random *not* from the same column, there are

$$(0.038)\frac{n(n-\frac{n}{r})}{2}$$

ways to do that. Thus, the probability of two letters being chosen at random from our text being the same is

$$\frac{1}{n(n-1)}\left((0.065)\frac{n(\frac{n}{r}-1)}{2} + (0.038)\frac{n(n-\frac{n}{r})}{2}\right)$$

which can be rearranged as

$$\frac{1}{r(n-1)}(0.027n + r(0.038n - 0.065))$$

With this definition of the Index of Coincidence, we are able to infer the probable length r of our keyword from our n-letter intercepted ciphertext.

Definition 2.4.3 (Index of Coincidence)**.** Let IC be the probability that two letters randomly selected from an n-letter text are the same, and r be the length of a keyword of distinct letters. Then

$$IC \approx \frac{1}{r(n-1)}(0.027n + r(0.038n - 0.065))$$

With the Kasiski Test and the Index of Coincidence, an eavesdropper can find the length of a keyword with ease. For example, if Kasiski Test gives $7|r$ and Index of Coincidence gives $r \approx 15.3$, $r = 14$ (probably) [5]. Between Kasiski and Friedman, the security of polyalphabetic substitution was well and truly shattered.

2.4.2 *WWII (An almost wholly historical anecdote)*

The cryptology of WWII was characterized by machines and depth. Depth is achieved when a large amount of intercepted ciphertexts from the same sender using the same cipher is collected, and allows cryptanalysts to more easily discover patterns and footholds. WWII was where symmetric key encryption reached its peak - numerous histories of the period pin the successes and failures of the war on participants' abilities to read encrypted messages.[5]

The words "WWII" and "code-breaking" typically inspire thoughts of Bletchley Park, Turing, and the Enigma. However, the cryptanalysis of WWII was much, much bigger than the four-rotor Enigma machine and British attempts at decrypting this admittedly complicated polyalphabetic substitution cipher.[6]

The bigger story of cryptanalysis during WWII is also the story of women cryptanalysts. In the United States, especially, women mathematicians, linguists, schoolteachers, and librarians were in high demand to fill analyst positions. With most able-bodied young men fighting in the Pacific or in Europe, the Army and Navy had no choice but to recruit all the young women they could (it was a bit of a pissing contest, as it turns out; the Navy got to ivy-league women's colleges first and so the Army aggressively recruited schoolteachers, librarians, and state college students). The combined force

5 Mundy *Code Girls*; Fagone *Woman Who Smashed Codes*
6 Mundy *Code Girls*

of often-brilliant, ever-diligant women analysts turned the tide of the war for the U.S. Achievements that effectively turned the tide of the war included breaking Purple (the Japanese diplomatic encryption machine), the Japanese Army code, the Japanese Navy code, and sifting through mountains of four-rotor Enigma decryptions from European analysts' bombes machines (you can applaud Turing here). These women, whose work was only recently declassified, are amazingly inspirational (and were a big part of the inspiration for this review of the history of symmetric key encryption).[7]

The increased traffic, depth, and nations' ability to employ enormous numbers of effective cryptanalysts motivated the development of algebraic cryptography. However, it took some time for the world to catch up with mathematical advances. Many polyalphabetic and transposition ciphers were still utilized - they were complicated by age old techniques like using codebooks (a collection of words and corresponding codewords) or frequently changing encryption keys [2].

2.4.3 *Hill Ciphers*

Cryptography underwent its next big adaptation with the introduction of abstract algebra structures. Lester Sanders Hill, a mathematician interested in error-detection and cryptography, was, quite probably, the first to suggest using abstract algebra to construct cryptosystems [6]. He published a 1929 paper outlining the use of the ring $\mathbb{Z}_{26}$ to set up his cryptosystem. Hill constructed a polygraphic cipher - a cipher that encrypts blocks of plaintext letters - using matrices [6]. To understand his cryptosystem more fully, we need to review some linear algebra.

Selection of Linear Algebra

Let A be a $n \times n$ square matrix such that $A = [a_{ij}]$.

Definition 2.4.4. A $n \times n$ matrix A is *nonsingular* if there is a $n \times n$ matrix B such that $AB = BA = I$ where I is the identity matrix. We say $B = A^{-1}$.

To give a general definition of the determinant of a square matrix, which will be important later, we need to define some terms. An *elementary product* from an $n \times n$ matrix A is a product of n entries of A such that no two come from the same row or same column. The elementary product can be expressed in the form $a_{1j_1}a_{2j_2}\dots a_{nj_n}$ where the column indices $j_1, j_2, \dots, j_n$ form a permutation of the integers from 1 to n and the row indices are in natural order (i.e. $1, 2, \dots, n$). A *signed elementary product* $\pm a_{1j_1}a_{2j_2}\dots a_{nj_n}$ is $+$ if the permutation of the column indices is even and $-$ if the permutation is odd.

7 Mundy *Code Girls*; Fagone *Woman Who Smashed Codes*

Definition 2.4.5. The *determinant* of a $n \times n$ matrix A is denoted $|A|$ and is defined to be the sum of all signed elementary products from A, or

$$|A| = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = \sum \pm a_{1j_1} a_{2j_2} \dots a_{nj_n}$$

We call this the nth order determinant.

Theorem 2.4.6. *A is invertible if and only if $|A| \neq 0$.*

Theorem 2.4.7. *If B is the matrix that results when a multiple of one row of A is added to another row, then $|B| = |A|$. The same is true for columns.*

A system of linear equations in matrix form $AX = 0$ is said to be homogeneous, where $AX = H$ such that $H \neq 0$ is nonhomogeneous. Note that the homogeneous system of linear equations in matrix form $AX = 0$ will always have a trivial solution $X = 0$.

Theorem 2.4.8. *Every system of linear equations has zero, one, or infinitely many solutions.*

Lemma 2.4.9. *A homogeneous system of linear equations has a unique trivial solution if and only if its determinant is nonzero. Otherwise the system has an infinite number of solutions.*

Proof. Let $AX = 0$ be the matrix form of a homogeneous system of linear equations. Suppose $|A| \neq 0$ and there are two solutions X_1 and X_2 such that $X_1 \neq X_2$. So $AX_1 = 0$ and $AX_2 = 0$. Since $|A|$ is nonzero, by Theorem 2.4.6 there exists an inverse matrix A^{-1}. Thus

$$X_1 = A^{-1}(AX_1) = A^{-1}(0) = 0 = A^{-1}(AX_2) = X_2$$

and there exists a unique, trivial solution. Now suppose X_1 and X_2 such that $X_1 \neq X_2$ and $|A| = 0$. By Theorem 2.4.6, A is not invertible. Notice any linear combination of the solutions $aX_1 + bX_2$ where $a, b \in \mathbb{Z}$ is also a solution

$$A(aX_1 + bX_2) = A(X_1a + X_2b) = AX_1a + AX_2b = 0a + 0b = 0$$

Thus if $|A| = 0$, $AX = 0$ has infinitely many solutions. □

Theorem 2.4.10. *Let X_p be some solution to $AX = H$. Then any solution X of $AX = H$ can be expressed in the form $X = X_p + X_h$ where X_h is a solution of the associated homogeneous equation $AX = 0$. Further, any vector $X = X_p + X_h$, where $AX_h = 0$, is a solution of $AX = H$.*

Proof. Let X_p be some solution to $AX = H$ and X_h be a solution to $AX = 0$. Consider $X = X_p + X_h$. Notice

$$AX = A(X_p + X_h) = AX_p + AX_h = H + 0 = H$$

so X is also a solution to $AX = H$. Now suppose $X_h = X + (-1)X_p$. We see X_h is a solution of $AX = 0$ since $AX_h = AX + (-1)AX_p = H + (-1)H = 0$. Then $X = X_p + X_h$ and it follows that the solution set of $AX = H$ is $\{X_p + X_h | AX_h = 0\}$. □

Definition 2.4.11. Let U and V be subsets of vectors of the commutative ring with unity R. A function $T : U \to V$ is called a linear transformation if and only if

a) $T(X + Y) = T(X) + T(Y)$ for all $X, Y \in U$

b) $T(rX) = rT(X)$ for all $X \in U$, $r \in R$

Theorem 2.4.12. *Let $T : U \to V$ be a linear transformation. If $e_1, e_2, \ldots, e_n$ are standard unit vectors in U, and if X is any vector in U, then $T(X)$ can be expressed as*

$$T(X) = AX$$

where

$$A = [T] = [\ T(e_1) \quad T(e_2) \quad \ldots \quad T(e_n)\ ]$$

The matrix A is called the *standard matrix for T* and we say T is the transformation A.

Theorem 2.4.13. *The standard matrix A of the linear transformation T is nonsingular (Definition 2.4.4) if and only if there is a linear transformation $T^{-1} : V \to U$ such that*

$$T^{-1} \circ T(X) = T \circ T^{-1}(X) = X.$$

We call T^{-1} the inverse of T.

Theorem 2.4.14. *The inverse of a linear transformation is unique.*

For more linear algebra see Anton and Busby's *Contemporary Linear Algebra* [7].

Hill's substitution cipher

Now that we've remembered (enough) linear algebra, let's return to Hill's cipher [6]. Hill's cryptosystem used the integers mod 26 where $a_0, a_1, a_2, \ldots, a_{25} \in \mathbb{Z}_{26}$ corresponded to letters of the English alphabet. This one-to-one number-to-letter correspondence between $\mathbb{Z}_{26}$ and an alphabet of 26 characters (like English) allowed for cryptology in ring algebra.

Hill's cipher is similar to a multiplication mod p cipher that uses matrix multiplication to encrypt blocks of plaintext. Letting R be a commutative ring with unity, the key matrix K is from the keyspace of $n \times n$ matrices such that for $k \in \mathcal{K}$, $|K|$ is a unit in R. The plaintext $m \in \mathcal{M}$ takes the form of a vector of length n. The encryption function is as follows

$$e_k(m) = K_{n \times n} m$$

The corresponding decryption function is

$$d_k(c) = K^{-1}c$$

This kind of cipher - that encrypts the plaintext in blocks of n - is called a polygraphic cipher.[8] Hill claims that for $n > 4$ the cipher is both easy to use and difficult to break [6]. However, performing matrix operations of higher n's by hand is prone to human error. The use of machines readily solves this issue [6].

To construct an invertible key matrix, Hill suggests that a cryptographer take a unit element b of R and construct K such that

$$K = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & b \end{bmatrix}_{n \times n}$$

Since the key matrix was constructed by multiplying a row of the $n \times n$ identity matrix, $K = R_i(b)I_n$, it follows by properties of linear algebra that $|K| = b|I_n| = b$. To complicate the elements of K, the cryptographer can add multiples of rows to other rows without changing the determinant (and thus the invertibility) of the matrix.

Suppose Alice's colleague Alex is trying to encrypt a plaintext message

$$m = [m_1, m_2, \dots, m_n, m_{n+1}, \dots, m_{rn}]$$

to send to Bobbi (Bob's cousin.)[9] An encryption using Hill's cipher first partitions the plaintext into blocks of length n:

$$m_1, m_2, \dots, m_n \Big| m_{n+1}, m_{n+2}, \dots, m_{2n} \Big| m_{2n+1}, \dots$$

and so on. Then the plaintext blocks are multiplied by the key matrix, as below, to get the corresponding ciphertext:

$$\begin{bmatrix} k_{11} & \dots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{n1} & \dots & k_{nn} \end{bmatrix} \begin{bmatrix} m_1 \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$$

$$\begin{bmatrix} k_{11} & \dots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{n1} & \dots & k_{nn} \end{bmatrix} \begin{bmatrix} m_{n+1} \\ \vdots \\ m_{2n} \end{bmatrix} = \begin{bmatrix} c_{n+1} \\ \vdots \\ c_{2n} \end{bmatrix}$$

$$\vdots$$

8 Another example of a polygraphic cipher is the Playfair cipher invented in 1854 by Charles Wheatstone

9 the whole family is in the business of secrets

$$\begin{bmatrix} k_{11} & \dots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{n1} & \dots & k_{nn} \end{bmatrix} \begin{bmatrix} m_{(r-1)n+1} \\ \vdots \\ m_{rn} \end{bmatrix} = \begin{bmatrix} c_{(r-1)+1} \\ \vdots \\ c_{rn} \end{bmatrix}$$

In the case that the length of the plaintext message is not divisble by n, Alex need only pad the message with nonsense to increase the length.

Hill's polygraphic substitution cipher has a couple of advantages over the mono- or polyalphabetic substitution ciphers we've investigated previously. For one, encryption of blocks of plaintext enables the sender to encrypt longer messages and the use of machines renders the encryption process easy. Additionally, frequency analysis and the Index of Coincidence can only be used intermittently. On digraphic ciphers, where the encryption matrix is 2×2 and plaintext is encrypted in blocks of 2 letters, an eavesdropper may be able to decrypt a ciphertext using the relative frequency or IC of digraphs.[10] However, as the dimension of the key matrix and plaintext blocks increases, so too does the difficulty of this cryptanalytic technique. To successfully use frequency analysis of n-graphs, the cryptanalyst would need a very large amount of ciphertext to work from (something on the scale of a $90n$ character ciphertext, if we're following the sufficient sample size proposed by Ibn Adlan).

Despite its apparent advantages, the Hill substitution cipher fails to satisfy Kerchoff's Desirable Properties of a Cryptosystem (Definition 2.2.1). It is easy to calculate both $e_k(m)$, given the plaintext and key, and $d_k(m)$, given the ciphertext and key. Unfortunately for the desirability of the Hill cipher, though, it is susceptible to both known-plaintext and ciphertext-only attacks; Levine (1961) [8] outlined an algorithm for a successful known-plaintext attack and Khazaei and Ahmadi (2017) [9] proved a ciphertext-only attack of reduced complexity.

Hill's affine cipher

Following his successful construction of a simple substitution cipher using a square matrix key, Lester Hill published an affine cipher [10] that also utilizes properties of commutative rings with unity and matrix operations.

Similar to his substitution cipher, Hill's affine cipher is constructed using a commutative ring with unity R where K is from the keyspace of $n \times n$ matrices such that for $k \in \mathcal{K}$, $|K|$ is a unit in R, and the plaintext $m \in \mathcal{M}$ is a vector of length n. There is an additional key k that takes the form of a n-length vector of elements in R. The encryption function for Hill's affine cipher is

$$e_k(m) = Km + k$$

10 Called "Sinkov's Digram Roughness Ratio"

Hill initially defined his affine cipher in terms of a linear transformation

$$c = T_k([m:1]) \qquad \text{where} \qquad [T_k] = \left[\begin{array}{c|c} K & k \end{array}\right]_{n\times(n+1)}$$

By properties of linear algebra (Theorem 2.4.12) we see that Hill's linear transformation $c = T_k(m)$ is equivalent to the encryption function $e_k(m)$. In the above transformation T_k, the *frame matrix* is the $n \times n$ matrix K.

Lemma 2.4.15. *Let T_a, T_b, and T_c be linear transformations and A, B, and C be associated frame matrices. If $T_c = T_a T_b$ then $C = AB$.*

Theorem 2.4.16. *Let T_a be some transformation that has a nonsingular frame matrix, then there exists a unique transformation T_b such that $T_b = T_a^{-1}$*

Proof. Let T_a be a homogeneous linear transformation with a nonsingular standard matrix A such that $Ax = 0$. Since there exists only the trivial solution to $Ax = 0$, there also exists an inverse transformation $T_a^{-1}(x) = A^{-1}x$. Now let T_a be any linear transformation of the form $T_a(x) = Ax + c$ such that the standard matrix is $[T_a] = \left[\begin{array}{c|c} A & c \end{array}\right]$ where the frame matrix A is nonsingular. Consider $T_a(x) = Ax + c$. This can be rewritten as $T_a(x) = Ax + A(A^{-1}c)$ which in turn, if we let $a = A^{-1}c$, can be rewritten $T_a(x) = Ax + Aa$. Now consider the associated homogeneous transformation T_c with a nonsingular standard matrix A such that $T_c(x) = Ax$. By the first part of this proof, there exists a T_c^{-1} such that $T_c^{-1}(x) = A^{-1}x$. Notice that

$$T_c(x) = Ax = A(x-a) + Aa = T_a(x-a).$$

So, since we have T_c^{-1}, we see $T_c^{-1}(x) = T_a^{-1}(x-a)$. Thus

$$T_c^{-1}(x+a) = A^{-1}(x+a) = T_a^{-1}(x)$$

and there exists an inverse transformation of any linear transformation

$$T_a(x) = Ax + c.$$

□

Thus the corresponding decryption function for Hill's affine cipher is

$$d_k(c) = K^{-1}(c-k)$$

Hill's motivation for representing his cipher as a linear transformation comes from using machines. He argues that a polygraphic cipher that utilizes an involutary linear transformation (T is its own inverse) is fast, accurate, and cheap [10].

Modified Hill cipher

Though Hill published his work on polygraphic ciphers in rings in 1929, research on Hill ciphers is still active. Many researchers are attempting to improve the system's security, which is a primary concern for cryptographers. Using operations like Arnold transforms to increase the key space[11] or elliptic curve encryption[12] (for more on the ECC see Chapter 7.1) to convert the Hill cipher into an asymmetric system, researchers were able to increase security and (in some cases) efficiency. Some of these modified Hill ciphers have applications in fields like biometric privacy protection[13] and image encryption.

Definition 2.4.17. A $n \times n$ matrix whose rows are cyclically shifted versions of a length n list is called a *circulant matrix*. The length n list is called the *row vector*.

$$circ(c_1, c_2, \ldots, c_n) = \begin{bmatrix} c_1 & c_2 & \ldots & c_n \\ c_n & c_1 & \ldots & c_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ c_2 & c_n & \ldots & c_1 \end{bmatrix}_{n \times n}$$

Definition 2.4.18. A circulant matrix is *prime* if the gcd of the row vector is 1,

$$gcd(c_1, c_2, ..., c_n) = 1.$$

In 2012, Adinarayana et al.[14] proposed a modified Hill cipher system that utilizes a secret, shared circulant matrix key and an public, invertible matrix key. The public key matrix was chosen using what the authors termed a *coefficient matrix*. Given a matrix G, the coefficient matrix G_c is defined as the circulant matrix of the circulant matrices of the rows of G. For example, if

$$G = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \text{then} \qquad G_c = circ(circ(a,b), circ(c,d)) = \begin{bmatrix} a & b & c & d \\ b & a & d & c \\ c & d & a & b \\ d & c & b & a \end{bmatrix}$$

The algorithm for constructing the proposed modified Hill cryptosystem is as follows:

11 Mishra, D.C., Sharma, R.K., Ranjan, R., Hanmandlu, M. "Security of RGB image data by affine hill cipher over $SL_n(\mathbb{F}_q)$ and $M_n(\mathbb{F}_q)$ domains with Arnold transform." *Optik*, vol. 126, 2015, pp. 3812-3822.

12 Dawahdeh, Z.E., Yaakob, S.N., and bin Othman, R.R. "A new image encryption technique combining Elliptic Curve Cryptosystem with Hill Cipher." *Journal of King Saud University - Computer and Information Sciences*, 2017.

13 Acharya, B., Sharma, M.D., Tiwari, S., and Minz, V.K. "Privacy Protection of Biometric Traits using Modified Hill Cipher with Involutary Key and Robust Cryptosystem." *Procedia Computer Science*, vol. 2, 2010, pp. 242-247.

14 Adinarayana, R.K., Vishnuvardhan B., Madhuviswanatham, Krishna A.V.N. "A Modified Hill Cipher Based on Circulant Matrices." *Procedia Technology*, vol. 4, 2012, pp. 114-118.

- Alex selects $n \times n$ non-singular matrix G from $\mathbb{F}_p$ (where p is prime) such that $|G_c| = 0$ as a public key
- Alex also selects $n \times n$ prime circulant matrix $A \in \mathbb{F}_p$ as a secret key
- then Alex calculates the substitution key $K = AGA^{-1} \pmod{p}$

The encryption function is a modification of Hill's affine cipher using the *ith* row of the prime circulant matrix as the shift key. Let m_i be the *ith* n-length block of plaintext and c_i denote the corresponding *ith* ciphertext block. The encryption function is

$$e(m_i) = Km_i + v_i \pmod{p}$$

where v_i is the *ith* row of the prime circulant matrix A. Each time Alex wants to encrypt a new block of plaintext, a new v_i is selected from a row of A. After receiving the secret key (transmitted using a secure key exchange algorithm, see Section 3.2.1 for an example) and finding the public key, Bobbi is able to calculate the inverse substitution key $K^{-1} = AG^{-1}A^{-1} \pmod{p}$. The corresponding decryption function is

$$d(c_i) = K^{-1}(c_i - v_i) \pmod{p}$$

The key part of the security of this system is that the vector v_i changes for *each* encryption of a n-length plaintext block. (As of 2012) this technique defeated cryptanalysis by both a known-plaintext attack and ciphertext-only attack. One reason cryptanalysis of the modified Hill cryptosystem is difficult is because the substitution key $K = AGA^{-1} \pmod{p}$ forms a system of multivariate polynomial equations. This is an NP-hard problem (see Chapter 10) and is difficult to solve in a large prime modulus. An added benefit of this modified cryptosystem comes from the structure of the circulant key matrix; since the matrix is specified by the first row, key storage requirements are significantly reduced from n^2 elements in $\mathbb{F}_p$ to n elements

Cryptanalysis of Hill's substitution cipher

As mentioned previously, Hill's substitution cipher is susceptible to what is called a known-plaintext attack. Suppose Eve got bored with Alice and Bob's correspondence and wanted to snoop on Alex and Bobbi. Alex is using Hill's substitution cipher, $c = K_{n\times n}m$, to send messages to Bobbi. Eve has intercepted a message from Alex to Bobbi. Eve has also been quite lucky - she has managed to guess part of the plaintext. She assumes that [8]

- the cryptosystem language is English
- the number-to-letter correspondence of alphabet known (i.e. 1 corresponds to a, 17 corresponds to q, etc.)
- the key is involutary (its own inverse): $K = K^{-1}$

To find a solution Eve needs to determine the ciphertext block that corresponds to the known plaintext *and* determine the key matrix K. She recalls that Hill's cipher is of the form $c = Km$ where m and c are length n. Since Eve is assuming $K = K^{-1}$, she also knows that $m = Kc$. To determine the ciphertext block that corresponds to the known plaintext - the probable location of the plaintext - Eve constructs some equalities using properties of linear algebra. Using matrix multiplication, Eve knows the *ith* row of plaintext is

$$m_i = k_{i1}c_1 + \ldots + k_{in}c_n$$

and, similarly, the *ith* row of ciphertext is

$$c_i = k_{i1}m_1 + \ldots + k_{in}m_n.$$

With this knowledge she can construct matrices whose determinants are 0

$$\begin{bmatrix} m_1 & c_1 & \ldots & c_n \\ m_2 & c_1 & \ldots & c_n \\ \vdots & \vdots & \ddots & \vdots \\ m_n & c_1 & \ldots & c_n \\ m_{n+1} & c_{n+1} & \ldots & c_{2n} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} c_1 & m_1 & \ldots & m_n \\ c_2 & m_1 & \ldots & m_n \\ \vdots & \vdots & \ddots & \vdots \\ c_n & m_1 & \ldots & m_n \\ c_{n+1} & m_{n+1} & \ldots & m_{2n} \end{bmatrix}$$

Since the first column of the matrices is a linear combination of the other columns, the determinants of the matrices are 0 (by an equivalent property to Theorem 2.4.6). Eve can now text her known plaintext against blocks of ciphertext to determine its probable location. Once Eve finds the probable position, she can construct a system of linear equations to solve for the elements of the key matrix. If Eve's known plaintext is $m_1, m_2, \ldots, m_n, m_{n+1}, \ldots, m_{2n}$, she sets up the following system

$$\begin{aligned} m_1 &= k_{11}c_1 + \ldots + k_{1n}c_n \\ m_2 &= k_{21}c_1 + \ldots + k_{2n}c_n \\ &\vdots \\ m_n &= k_{n1}c_1 + \ldots + k_{nn}c_n \\ m_{n+1} &= k_{11}c_{n+1} + \ldots + k_{1n}c_{2n} \\ &\vdots \\ m_{2n} &= k_{n1}c_{n+1} + \ldots + k_{nn}c_{2n} \end{aligned}$$

in which the elements k_{ij} of the key matrix are unknowns. Eve realizes that if the coefficient matrix of her system of linear equations (which is not necessarily square)

$$\begin{bmatrix} c_1 & \ldots & c_n \\ c_1 & \ldots & c_n \\ \vdots & \ddots & \vdots \\ c_{n+1} & \ldots & c_{2n} \end{bmatrix}$$

contains an $n \times n$ determinant that is prime to 26, she can find a unique solution and the key matrix K follows. Otherwise, Eve will find multiple possible solutions and she

has to test each solution against her known plaintext and corresponding ciphertext to determine the correct key matrix K [8].

The above example of cryptanalysis on Hill's substitution cipher takes advantage of several assumptions, most importantly that a piece of the plaintext is known. This is not always the case. Khazaei and Ahmadi (2017) [9] performed a successful ciphertext-only attack on a $d \times d$ Hill system in $\mathcal{O}(d13^d)$ (for more on polynomial and non-polynomial time see Section 3.3.1). Starting with the trivial brute force attack that checks all possible secret keys, whose complexity is $\mathcal{O}(d^3 26^{d^2})$, Khazaei and Ahmadi (2017) use a divide-and-conquer technique that searches each column of the inverse key matrix separately. They use the Chinese Remainder Theorem (Theorem 3.5.3) to perform their divide-and-conquer attack modulo 13 and 2 (recall Hill's substitution cipher is in $\mathbb{Z}_{26}$). They then are able to obtain a key in $\mathcal{O}(d13^d)$ steps [9]. Khazaei and Ahmadi (2017) amiably conclude that "beating [their] results or proving its optimality based on some reasonable computational complexity assumptions remains an open problem" [9].

Despite the many advances in symmetric key encryption, and the altogether more utilized asymmetric key encryption, the old mottos of the cryptanalyst still stand: "let's suppose," and "the most important aid in cipher solution is a good eraser." (*Code Girls* 139-140)

3

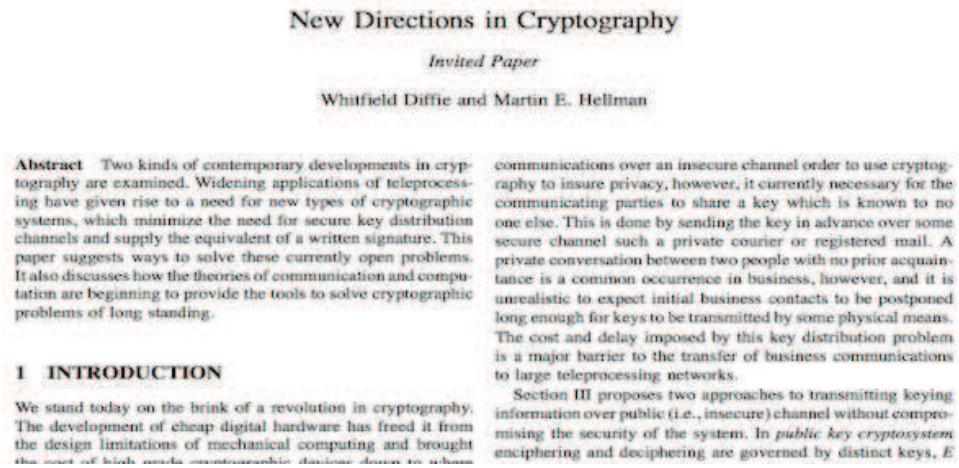
THE DISCRETE LOGARITHM PROBLEM

We stand today on the brink of a revolution in cryptography.

Thus begins Whitfield Diffie's and Martin Hellman's ground breaking 1976 paper *New Directions in Cryptography*. Indeed, the ideas set forth in this paper revolutionized not only cryptography, but computing, commerce and communications. *New Directions in Cryptography* introduced the basic definitions and goals of a new field of mathematics and computer science, a field whose existence was dependent on the then emerging age of the digital computer. Every time we make purchases over the internet, e-mail, or use social networking sites, we are relying on security measures based on ideas set forth in this pioneering paper.

The most important contribution of *New Directions in Cryptography* was the definition of a *Public Key Cryptosystem* and its associated components - trapdoor information and one-way functions. A *one-way function* is an invertible function which is easy to compute, but whose inverse is difficult to compute. What does "difficult to compute" mean? An inverse function is difficult to compute if any algorithm that attempts to compute it in a "reasonable" amount of time (i.e. less than the age of the universe), will fail almost certainly. (The phrase "almost certainly" means that the set of values for which the algorithm doesn't fail has measure zero is the space of all possible values.) Secure public key cryptosystems are based on one-way functions having a *trapdoor*. The trapdoor is some additional information which makes calculation of the inverse feasible. This trapdoor information is often called the private key.

Despite years of research, it is still not known whether one-way functions exist. In fact, a proof of the existence of one-way functions would simultaneously solve the famous $\mathcal{P} = \mathcal{NP}$ problem in complexity theory. Various candidates for one-way functions have been proposed, and some of them are used by modern public key encryption algorithms, but the security of these cryptosystems rests on the *assumption* that inverting the underlying function (or solving for the private key from the public key) is a hard problem. The situation is somewhat analogous to theories in Physics which gain credibility over time, as they fail to be disproved and continue to explain natural phenomena.



New Directions in Cryptography

Invited Paper

Whitfield Diffie and Martin E. Hellman

Abstract Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

1 INTRODUCTION

We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought

communications over an insecure channel order to use cryptography to insure privacy, however, it currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such a private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channel without compromising the security of the system. In *public key cryptosystem* enciphering and deciphering are governed by distinct keys, *E*

Figure 2.: Whitfield Diffie and Martin Hellman's ground breaking paper

Prior to the publication of *New Directions in Cryptography*, encryption research in the United States was the domain of the National Security Agency, and all information in this area was classified. Until the 1990's, the US government treated cryptographic algorithms as munitions, making their export an act of treason. Eventually the government conceded what had been obvious to mathematicians and computer scientists for 20 years; namely that trying to restrict free and open discussion about abstract cryptographic algorithms is futile. However, in order to maintain some control, the government continued to restrict export of "machine readable" cryptographic algorithms. While the goal of preventing global dissemination of sophisticated cryptographic techniques to potential enemies of the United States might be reasonable, the government's efforts were ultimately thwarted by two difficulties.

First, the existence of optical scanners makes nearly everything "machine readable." To protest the government's policy, people had a three line version of the RSA algorithm printed on t-shirts, thereby making these products munitions. Wearing such a t-shirt on an international flight subjected the wearer to a large fine and up to 10 years in prison. Tattoos of the RSA algorithm made people's bodies into non-exportable munitions.

Second, public key cryptosystems are based on underlying "hard" mathematical problems. The algorithms themselves are fairly simple and straightforward. The underlying mathematical problems are part of a historical and open discussion on mathematics. Thus, once the concept of public key cryptography was introduced, it was almost inevitable that mathematicians and computer scientists across the world would rediscover the particulars of a given classified cryptosystem.

Figure 3.: Nonexportable munitions: three-line implementation of RSA in Perl

3.0.1 *Public Key Exchange*

Just as Caesar had to deliver his key to the members of the military, so to do banks have to deliver PIN numbers to their account holders. To share the secret key, communicating parties must either meet, or use a trusted courier. This is one of the fundamental challenges of symmetric key cryptography. Trusted couriers are hard to come by and it is often impractical for parties to meet. This need to establish and share a secret key precludes secure communication between entities who have not had previous contact. Moreover, key management becomes an issue since a different secret key is needed for each party with whom one wants to communicate. For example, 171 secret keys would be needed in order to have secure communications between each of the 19 member nations of NATO.

It wasn't until the publication of *New Directions in Cryptography* that a solution to this key dissemination and management problem arose[1]. Whitfield Diffie and Martin Hellman's publication introduced the idea of public key exchange. Like all public key cryptosystems, the security of the Diffie-Hellman public key exchange protocol is based on an underlying *hard* mathematical problem. In particular, the Diffie-Hellman key exchange is based on the assumption that the discrete logarithm problem (DLP) is difficult to solve.

1 It is unclear exactly when PKC was developed. Recently declassified documents show that the British government's GCHQ (Government Communications Headquarters) was working on PKC as early as 1970. There is some suggestion that the NSA may have been developing similar algorithms as early as 1962, as part of its nuclear weapon control research. What is clear is that PKC was not *publicly* known until the Martin Hellman, and his graduate students Whitfield Diffie and Ralph Merkle, began publishing their work.

3.1 THE DISCRETE LOGARITHM PROBLEM

The discrete logarithm problem is a mathematical problem which arises in a number of different settings, including finite fields and elliptic curves. The first published public key construction, due to Diffie and Hellman, is based on the the discrete logarithm problem in a finite field $\mathbb{F}_p$.

3.1.1 *Generic Discrete Logarithm Problem*

We can take any finite group and use the group law to pose the discrete logarithm problem as follows:

Definition 3.1.1. Let G be a group whose group law we denote by the symbol $*$. *The Discrete Logarithm Problem* for G is to determine for any two given elements g and h in G, an integer x such that g composed with itself x times gives h, i.e.

$$g * g * \cdots * g = h.$$

3.1.2 *Discrete Logarithm Problem in* $\mathbb{F}_p$

Given the field structure of $\mathbb{F}_p$, there are two ways to pose the discrete logarithm problem – one using the additive operation of the field, and the other using the multiplicative operation.

Definition 3.1.2. *The Discrete Logarithm Problem for addition* in $\mathbb{F}_p$ is to determine, for any two elements g and h in $\mathbb{F}_p$, an integer x such that $x \cdot g \equiv h \pmod{p}$, i.e.

$$\underbrace{g + g + \cdots + g}_{x\text{-times}} \equiv h \pmod{p}.$$

Definition 3.1.3. *The Discrete Logarithm Problem for multiplication* in $\mathbb{F}_p$ is to determine, for any two elements g and h in $\mathbb{F}_p$, an integer x such that $g^x \equiv h \pmod{p}$, i.e.

$$\underbrace{g * g * \cdots * g}_{x\text{-times}} \equiv h \pmod{p}.$$

Remark 3.1.4. The multiplicative discrete logarithm problem is not well-posed for all values of g in $\mathbb{F}_p^*$. To make it a well-posed problem, g must be a generator of $\mathbb{F}_p^*$ so that every element $a \in \mathbb{F}_p^*$ can be expressed as $a = g^x$ for an integer x. Even for values of g which make the DLP well-posed, the corresponding discrete logarithm function $\log_g$ is not well defined from $\mathbb{F}_p \to \mathbb{Z}$.

To show this, consider $\log_g(h) = a$ such that $g^a = h$ where $a \in \mathbb{Z}$ and $g \in \mathbb{F}_p$. Further consider $g^{(a+|g|n)}$ where $n \in \mathbb{Z}$. It follows that $g^{(a+|g|n)} = g^a * g^{|g|n} = g^a * 1 = g^a = h$. Thus $\log_g$ is not well defined since $\log_g(h) = a + |g|n$ and $\log_g(h) = a$. In other words, the integers contain infinite solutions that are equivalent $\pmod{|g|}$.

Theorem 3.1.5. *Given $g \in \mathbb{F}_p^*$ where g is a generator,*

$$\log_g : \mathbb{F}_p^* \to \mathbb{Z}/(p-1)\mathbb{Z}$$

is a well-defined function. Furthermore, it has the familiar properties of the continuous logarithm function,

i. $\log_g(h_1h_2) = \log_g(h_1) + \log_g(h_2) \qquad \forall h_1, h_2 \in \mathbb{F}_p^*$,

ii. and $\log_g(h^n) = n\log_g(h) \qquad \forall h \in \mathbb{F}_p^*$ *and* $n \in \mathbb{Z}$.

Proof. To prove that the function is well defined, let g be an element of $\mathbb{F}_p^*$ such that g is a generator and let $h_1, h_2 \in \mathbb{F}_p^*$. Suppose $h_1 \equiv h_2 \pmod{p}$ and consider $\log_g(h_1) \equiv x_1 \pmod{p-1}$ and $\log_g(h_2) \equiv x_2 \pmod{p-1}$ for $x_1, x_2 \in \mathbb{Z}/(p-1)\mathbb{Z}$. Thus, $g^{(x_1)} \equiv h_1 \equiv h_2 \equiv g^{(x_2)} \pmod{p}$ by definition of the logarithm function. By 1.2.40, since $g^{x_1} \equiv g^{x_2} \pmod{p}$, it follows that $x_1 \equiv x_2 \pmod{|g|}$. Since g is a generator, $|g| = p-1$, and thus $x_1 \equiv x_2 \pmod{p-1}$. Therefore $\log_g : \mathbb{F}_p^* \to \mathbb{Z}/(p-1)\mathbb{Z}$ is well defined.

To prove property *i*, consider $\log_g(h_1) \equiv x_1 \pmod{p-1}$ and $\log_g(h_2) \equiv x_2 \pmod{p-1}$. By definition of logarithm and equivalence in $\mathbb{F}_p^*$, $g^{x_1} \equiv h_1 \pmod{p}$ and $g^{x_2} \equiv h_2 \pmod{p}$. Consider $x \equiv \log_g(h_1h_2) \pmod{p-1}$. By substitution, $x \equiv \log_g(h_1h_2) \equiv \log_g(g^{x_1}g^{x_2}) \equiv \log_g(g^{x_1+x_2}) \equiv x_1 + x_2 \equiv \log_g(h_1) + \log_g(h_2) \pmod{p-1}$. Therefore, $\log_g(h_1h_2) = \log_g(h_1) + \log_g(h_2)$.

To prove property *ii*, consider $\log_g(h^n)$. This is equivalent to $\log_g \underbrace{(h * h * \cdots * h)}_{n\text{–times}}$. By property *i*, this is equivalent to $\underbrace{\log_g(h) + \cdots + \log_g(h)}_{n\text{-times}}$. Simplifying this term gives $\log_g(h^n) = n\log_g(h)$. □

Theorem 3.1.6. *Let p be an odd prime and $g \in \mathbb{F}_p^*$ be a generator. For all $a \in \mathbb{F}_p^*$, a has a square root modulo p if and only if its discrete logarithm $\log_g(a) \pmod{p-1}$ is even.*

Proof. Let p be an odd prime and $g \in \mathbb{F}_p^*$ be a generator.

($\Rightarrow$) Let a and x be elements of $\mathbb{F}_p^*$ such that $x^2 \equiv a \pmod{p}$. Since g generates $\mathbb{F}_p^*$, we know that $x = g^n$ for some $n \in \mathbb{Z}$. Now, let $\log_g(a) \equiv b \pmod{p-1}$ for some $b \in \mathbb{Z}$. By definition of logarithm, it follows that $g^b \equiv a \pmod{p}$. Thus, we have $(g^n)^2 \equiv (x)^2 \equiv a \equiv g^b \pmod{p}$. This implies that $g^b \equiv g^{2n} \pmod{p}$, and we see by

1.2.40 that $b \equiv 2n \pmod{|g|}$ where $|g| = p-1$. Since $b \equiv 2n \pmod{p-1}$, and an even number in an even modulus is even, it follows that $b = \log_g(a) \pmod{p-1}$ is even.

($\Leftarrow$) Let a be an element of $\mathbb{F}_p^*$, and suppose that $\log_g(a) \pmod{p-1}$ is even. By definition of even number, $\log_g(a) \equiv 2n \pmod{p-1}$ for some integer n. By definition of logarithm, we know that $a \equiv g^{2n} \pmod{p}$. This implies that $a \equiv (g^n)^2 \pmod{p}$. Thus, a has a square root modulo p.

Thus for all $a \in \mathbb{F}_p^*$, a has a square root modulo p if and only if its discrete logarithm $\log_g(a) \pmod{p-1}$ is even. □

3.2 DIFFIE-HELLMAN & ELGAMAL PUBLIC KEY CRYPTOSYSTEMS

In this section we will define two major cryptosystems: the Diffie-Hellman Public Key Exchange, and the ElGamal Public Key Cryptography.

3.2.1 *Diffie-Hellman Key Exchange*

In public key exchange, parties transmit all information over insecure channels and use this information to set up a shared secret key known only to the communicating parties. This shared secret key can then be used in any traditional symmetric key crypto-system. Although adversaries have full access to the information being transmitted, they cannot calculate the shared secret key from this information. Diffie-Hellman key exchange has enabled the boom in e-commerce over the last 20 years. Sending information over the insecure Internet, companies and potential buyers are able to establish a common secret key. This shared key can then be used to encrypt financial and personal information (such as credit card numbers and social security numbers).

Definition 3.2.1 (Diffie-Hellman Public Key Exchange Protocol)**.** The following protocol allows Alice and Bob to establish a secure symmetric key while conducting all communication over insecure channels.

Step 1. Alice and Bob agree on a large prime p and an integer $g \in \mathbb{F}_p^*$. These values are considered public knowledge, since Alice and Bob agree upon them using insecure channels.

Step 2. Alice picks a secret integer (also known as her private key) a, while Bob picks a secret integer (his private key) b. Only Alice knows a, and only Bob knows b. Alice computes $A = g^a \pmod{p}$, while Bob computes $B = g^b \pmod{p}$. Alice then sends Bob her value A, and Bob sends Alice his value B. Thus, while a and b

are secret, since Alice and Bob communicate over insecure channels, A and B are public knowledge.

Step 3. Upon receiving Bob's value B, Alice calculates the shared secret key $K = B^a \pmod{p}$. Similarly, Bob calculates $K' = A^b \pmod{p}$. Since

$$K = B^a = (g^b)^a = g^{ba} = g^{ab} = (g^a)^b = A^b = K' \pmod{p},$$

Alice and Bob now share a secret key K.

Example 3.2.2. We will demonstrate the Diffie-Hellman Public Key Exchange Protocol with an example.

Step 1. Suppose that Alice and Bob agree on the large prime $p = 307$ and the integer $g = 425 \equiv 118 \in \mathbb{F}_{307}^*$.

Step 2. Next, suppose that Alice chooses the private key $a = 323$ and Bob chooses the private key $b = 547$. Alice computes

$$A \equiv g^a \equiv 118^{323} \equiv 53 \pmod{307},$$

while Bob computes

$$B \equiv g^b \equiv 118^{547} \equiv 94 \pmod{307}.$$

Step 3. After exchanging their values for A and B, Alice calculates

$$K \equiv B^a \equiv 94^{323} \equiv 287 \pmod{307},$$

and Bob calculates

$$K' \equiv A^b \equiv 53^{547} \equiv 287 \pmod{307}.$$

Now, Alice and Bob share the secret key $K \equiv 287 \equiv K' \pmod{307}$.

An adversary Eve has access to the values of A, B, g, and p. If she can solve the DLP, then she can find a and b and calculate the shared key K. But is this the only way of calculating K from A, B, g, and p? In fact, an adversary Eve need only solve what is known as the Diffie-Hellman Problem.

Definition 3.2.3 (The Diffie-Hellman Problem (DHP))**.** Let p be a prime number and g an integer. The Diffie-Hellman problem (DHP) is the problem of computing the value $g^{ab} \pmod{p}$ from the known values of $g^a \pmod{p}$, and $g^b \pmod{p}$.

It is clear that the DHP is no harder than the DLP, but the converse is less clear. In fact, it is not known whether any method of solving the DHP would necessarily lead to a solution of the DLP.

3.2.2 *ElGamal Public Key Cryptography*

While the Diffie-Hellman key exchange algorithm provides a method of publicly sharing a random secret key, it falls short of being a public key cryptosystem (PKC). To be a cryptosystem, it must be possible to transmit a broad range of information, not just a secret key. The first public key cryptosystem was the RSA cryptosystem of Rivest, Shamir and Adleman. While RSA was historically first, the ElGamal cryptosystem described by Taher ElGamal in 1985 is a more natural extension of the ideas presented by Diffie and Hellman. Like Diffie-Hellman key exchange, the security of the ElGamal cryptosystem is based on the difficulty of the discrete logarithm problem. In this section we will describe the version of the ElGamal PKC set in $\mathbb{F}_p^*$, but using the generic discrete logarithm problem, we can adapt the algorithm to any finite group G. For example, as we'll see later, we can define an ElGamal PKC based on elliptic curve groups.

In public key cryptography, rather than one symmetric key used in both decryption and encryption, there is a public key used for encryption and a private key used for decryption. Anyone can encrypt a message for Alice using her public key, but only Alice can decrypt the message since decryption involves a private key known only to Alice. If Bob and Alice need secure communications in both directions, i.e. Bob needs to encrypt messages for Alice and Alice needs to encrypt messages for Bob, then they will each need a public/private key pair.

Definition 3.2.4 (ElGamal PKC in $\mathbb{F}_p^*$)**.** In the $\mathbb{F}_p^*$ ElGamal PKC, Alice establishes her public/private key pair as follows. First she selects a large prime p for which the discrete logarithm problem in $\mathbb{F}_p^*$ is difficult. She publishes this prime p along with an element $g \pmod p$ of large prime order. She then chooses a private key a and calculates her public key A as follows:

$$A = g^a \pmod p.$$

Alice publishes her public key A and keeps her private key a secret.

- Encryption: Suppose Bob wants to encrypt a message $m \in \mathbb{F}_p^*$, where $m \neq 1$, for Alice. Bob first chooses a random number $k \pmod p$. Bob uses k to encrypt one, and only one, message, and then he discards it. The number k is called the *ephemeral key*. For each message Bob encrypts for Alice, he will need to select another random ephemeral key. Bob computes the following quantities using his message, his ephemeral key, and Alice's public information (namely A, g, and p):

  $$c_1 = g^k \pmod p \qquad \text{and} \qquad c_2 = mA^k \pmod p$$

  Bob's ciphertext is the tuple (c_1, c_2).

- Decryption: $d_A(c_1, c_2) = c_2((c_1)^a)^{p-2} \equiv m \pmod p$

Step 1. Alice calculates $(A^k \pmod p)$ using the public knowledge, c_1, and her private knowledge a as follows: $(c_1)^a \equiv (g^k)^a \equiv (g^a)^k \equiv A^k \pmod p$.

Step 2. She then finds $(A^k)^{-1}$ using 1.2.41 as follows: $(A^k)^{-1} \equiv (A^k)^{p-2} \pmod{p}$.

Step 3. Alice then can calculate Bob's message, m, using the public information c_2 as follows: $c_2(A^k)^{-1} \equiv mA^k * (A^k)^{-1} \equiv m \pmod{p}$.

Remark 3.2.5. Using the abstract mathematical formulation, the ElGamal PKC is given by the set of functions

$$e_A : \mathcal{K} \times \mathcal{M} \to \mathcal{C} \times \mathcal{C} \text{ and } d_a : \mathcal{C} \times \mathcal{C} \to \mathcal{M}, \text{ where } \mathcal{K} = \mathbb{Z}, \mathcal{M} = \mathbb{F}_p^*, \text{ and } \mathcal{C} = \mathbb{F}_p^*.$$

Is the ElGamal system as hard for an adversary Eve to attack as the Diffie-Hellman problem? Or, by introducing a clever way of encrypting messages, have we unwittingly opened a back door that makes it easy to decrypt messages without solving the DHP? One of the goals of modern cryptography is to identify an underlying hard problem like the DHP and to *prove* that a given cryptographic scheme like ElGamal is at least as hard to attack as the underlying problem. In this case, we would like to prove that anyone who can decrypt arbitrary ElGamal ciphertexts without knowledge of the private key, must be able to solve the DHP.

Definition 3.2.6. In cryptography an *oracle* is a hypothetical machine which solves a particular problem. It takes in certain inputs and returns a specific answer, but gives no other information.

Theorem 3.2.7. *Fix a prime p and a base g to use for ElGamal encryption. Suppose that Eve has access to an oracle that decrypts arbitrary ElGamal ciphertexts encrypted using arbitrary ElGamal public keys (i.e. you would enter A, c_1 and c_2 into the oracle, and the oracle would return m). Eve can use this oracle to solve the Diffie-Hellman problem. Similarly, if Eve has access to an oracle which solves the Diffie-Hellman problem, she can use it to decrypt messages that have been encrypted using the ElGamal PKC.*

Proof. Let p be a prime and let g be an element of large prime order in $\mathbb{F}_p^*$.

First, suppose Eve has access to an oracle that decrypts arbitrary ElGamal ciphertexts. In other words, $O(A, c_1, c_2) = m$. Notice that Eve has access to the following information in ElGamal PKC: $A \equiv g^a \pmod{p}$, $c_1 \equiv g^k \pmod{p}$, and $c_2 \equiv mA^k \pmod{p}$. Using 1.2.41, and the oracle's output, m, Eve can find m^{-1}. Now, Eve can calculate $m^{-1}c_2 \equiv m^{-1}mA^k \equiv A^k \equiv g^{ak} \pmod{p}$. Thus, using the oracle, Eve is able to compute $g^{ak} \pmod{p}$ after inputting $A \equiv g^a \pmod{p}$ and $c_1 \equiv g^k \pmod{p}$. Therefore, Eve can solve the Diffie-Hellman problem using the oracle.

Now, suppose Eve has access to an oracle that solves the Diffie-Hellman problem. In other words, $O(g^a \pmod{p}, g^k \pmod{p})) = g^{ak} \pmod{p}$. Notice that Eve has access to the following information in ElGamal PKC: $A \equiv g^a \pmod{p}$, $c_1 \equiv g^k \pmod{p}$, and $c_2 \equiv mA^k \pmod{p}$. Eve can use the oracle to calculate $g^{ak} \equiv A^k \pmod{p}$. Since $A^k \in \mathbb{F}_p^*$, she

can find $(A^k)^{-1}$. Now, Eve can calculate $c_2(A^k)^{-1} \equiv mA^k(A^k)^{-1} \equiv m \pmod{p}$. Thus, using the oracle that can solve the Diffie-Hellman problem, Eve can decrypt messages that have been encrypted using the ElGamal PKC. □

An attack in which Eve has access to an oracle that decrypts arbitrary ciphertexts is known as a *chosen ciphertext attack*. Theorem 3.2.7 shows that the ElGamal system is secure against chosen ciphertext attacks, or to be more precise, it is secure provided the Diffie-Hellman problem is hard.

3.3 HOW HARD IS THE DLP? BIG-$\mathcal{O}$ NOTATION

Given a group G and two elements $g, h \in G$, the discrete logarithm problem asks for an exponent x such that $g^x = h \in G$. What does it mean to talk about the difficulty of this problem? How can we quantify "difficult"? A natural measure of difficulty is the approximate number of operations necessary for a person or a computer to solve the problem using the most efficient method currently known. For example, suppose we count the process of computing g^x as a single operation. Then the trial-and-error approach to solving the DLP would be to calculate g^x and compare it to h for $x = 1, 2, \ldots$. This process is guaranteed to find a solution in at most $|g|$ operations. If g has large enough order, for example $|g| > 2^{80}$, then this is not a practical algorithm with the computing power available today.

Order Notation was developed to make this notion of difficulty precise. It is prevalent in mathematics and computer science and provides a useful tool for measuring the magnitude of quantities.

Definition 3.3.1 (Order Notation)**.** Let $f(x)$ and $g(x)$ be functions of x. We say "f is big-$\mathcal{O}$ of g" and write

$$f(x) = \mathcal{O}(g(x))$$

if there are positive constants c and C such that

$$\big|f(x)\big| \leq c\big|g(x)\big| \qquad \forall x \geq C.$$

Definition 3.3.2 (Limit)**.** Let $f(x)$ be a function of x. We say that the limit of $f(x)$ as x approaches infinity is L and write

$$\lim_{x\to\infty} f(x) = L$$

if and only if for all $\epsilon > 0$, there exists a real number $N > 0$ such that for all $x > N, |f(x) - L| < \epsilon$.

Theorem 3.3.3. *If the limit*

$$\lim_{x\to\infty} \frac{f(x)}{g(x)}$$

exists (and is finite), then $f(x) = \mathcal{O}(g(x))$.

Proof. Assume that

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = L \qquad \text{for some } L \in \mathbb{R}.$$

Choose $\epsilon > |L|$. By definition of limit, there exists a positive $C \in \mathbb{R}$ such that for all $x > C$,

$$\left|\frac{f(x)}{g(x)} - L\right| < \epsilon.$$

Thus, we can see that

$$\left|\frac{f(x)}{g(x)}\right| < \epsilon + L,$$

which implies that

$$|f(x)| < (\epsilon + L)|g(x)|.$$

Now, let $c = \epsilon + L$, and notice that c is positive. By substitution, we have $|f(x)| < c|g(x)|$ for all $x > C$. Thus, by definition of big-$\mathcal{O}$, it follows that $f(x) = \mathcal{O}(g(x))$. □

Example 3.3.4. $x^2 = \mathcal{O}(2^x)$

First, notice that

$$\lim_{x\to\infty} \frac{x^2}{2^x} = \frac{\infty}{\infty} \qquad \text{and, therefore, the limit is indeterminate.}$$

Thus, we proceed by using L'Hôpital's Rule, which can be found in any standard calculus textbook. After applying the rule twice, we have

$$\lim_{x\to\infty} \frac{x^2}{2^x} = \lim_{x\to\infty} \frac{\frac{d}{dx}(x^2)}{\frac{d}{dx}(2^x)} = \lim_{x\to\infty} \frac{2x}{2^x \ln(2)} = \lim_{x\to\infty} \frac{\frac{d}{dx}(2x)}{\frac{d}{dx}(2^x \ln(2))} = \lim_{x\to\infty} \frac{2}{2^x (\ln(2))^2} = \frac{2}{\infty} = 0.$$

Since 0 exists and is finite, we have shown by 3.3.3 that $x^2 = \mathcal{O}(2^x)$.

3.3.1 *Polynomial-, Exponential-, and Subexponential-Time*

Order notation allows us to define several fundamental concepts that are used to get a rough handle on the computational complexity of mathematical problems. Suppose that we are trying to solve a certain type of mathematical problem, where the input is a number whose size can vary. We are interested in knowing how long it takes to solve the problem in terms of the size of the input. Typically, we use the number of bits as a measurement of the size of the input. Suppose that there is a constant $A \geq 0$, independent of the size of the input, such that if the input is $\mathcal{O}(k)$ bits long, then it takes

$\mathcal{O}(k^A)$ steps to solve the problem. Such a problem is said to be solvable in *polynomial-time*. If we take $A = 1$, then the problem is solvable in linear time; if $A = 2$ it is solvable in quadratic time. Polynomial-time algorithms are considered to be fast algorithms.

On the other hand, if there is a constant $c > 0$ such that for inputs of size $\mathcal{O}(k)$ bits, the problem takes $\mathcal{O}(e^{ck})$ steps to solve, then the problem is said to be solvable in *exponential-time*. Exponential-time algorithms are considered fast.

In between polynomial-time and exponential-time are *subexponential-time* algorithms. These have the property that for every $\epsilon > 0$, they solve the problem in $\mathcal{O}_\epsilon(e^{\epsilon k})$ steps. This notation means that the constants c and C from Definition 3.3.1 are allowed to depend on ϵ.

As a general rule of thumb in cryptography, problems solvable in polynomial-time are considered "easy" and problems that require exponential-time are considered "hard," with subexponential-time algorithms lying somewhere in between. It is important to remember that these are asymptotic descriptions. Depending on the big-$\mathcal{O}$ constants and the size of the input, an exponential problem may be easier to compute than a polynomial problem.

To illustrate these concepts, consider the Discrete Log Problem:

- Let $G = \mathbb{F}_p^*$ and consider the multiplicative DLP ($g^x = h \pmod p$). If the prime p is chosen between 2^k and 2^{k+1}, then g, h, and p all require at most k bits, so the problem can be stated in $\mathcal{O}(k)$-bits. Note, $\mathcal{O}(k) = \mathcal{O}(\log_2 p)$. If we try to solve the DLP using the trial-and-error method discussed previously, it takes at most $\mathcal{O}(p) = \mathcal{O}(2^k)$ steps to solve the problem. Thus, the trial-and-error algorithm is exponential-time. There are faster ways to solve the DLP in $\mathbb{F}_p^*$, some of which are very fast but work only for some primes, while others are less fast but work for all primes. For example, the Pohlig-Hellman algorithm described in section 3.5.2 solves the DLP in $\mathcal{O}(\sqrt{p}\log p)$ steps.
- Let $G = \mathbb{F}_p$ and consider the additive DLP ($x \cdot g = h \pmod p$). We can use the extended Euclidean Algorithm (Theorem 1.1.10) to calculate g^{-1} and then solve for x by setting $x = g^{-1}h \pmod p$. This takes $\mathcal{O}(\log p)$ steps, and thus is a linear-time algorithm. This is such a fast algorithm that the DLP in $\mathbb{F}_p$ under addition is not a good candidate for a one-way function in cryptography.
- Let $G = E(\mathbb{F}_p)$, i.e. the set of point of an elliptic curve over a finite field (as discussed in Chapter 7). The best known algorithms to solve the DLP in this setting requires $\mathcal{O}(\sqrt{N})$ steps, where $N = |G|$. Thus, it currently takes exponential-time to solve the elliptic curve discrete log problem (ECDLP).

3.4 SOLVING THE DISCRETE LOGARITHM PROBLEM (BY DAPHNE JACOBSEN)

The discrete logarithm problem not only is a mathematically interesting problem, but it is a measure of security for many encryption schemes. After the Diffie-Hellman paper, *New Directions in Cryptography* came out in 1976, the discreet logarithm problem had wide-spread cryptographic importance and is still relevant today.

Recall the Discrete Logarithm Problem

In common words, the discrete logarithm problem in a multiplicative setting is the search for x such that $g^x = h$. For the exact definition recall definition 3.1.1.

Definition Let G be a group whose group law we denote by the symbol *. The *Discrete Logarithm Problem* for G generated by g, is to determine for any element h in G, an integer x such that g composed with itself x times gives h, i.e.

$$\underbrace{g * g * \ldots * g}_{\text{x-times}} = h$$

Through out this chapter we will refer to the order of g as n such that $|g| = n$. Also, remember that we solve for $\log_g h$ modulo n.

Square Root Attacks

In the section we will discuss four different ways to attack the discrete logarithm problem in a arbitrary cyclic group. They vary in sophistication, but all use the same strategy of guess and check in order to solve the discrete logarithm problem. These methods all take time $\mathcal{O}(\sqrt{n})$. This is where these methods get their name, "square root attacks". The reason for this time estimation rests in probability theory and largely we will not go into that aspect of these methods. For further context, please refer to section 5.1. It has been shown that when solving the discrete logarithm problem by only calculating inverses and applying the group operation, that $\mathcal{O}(\sqrt{n})$ is the fastest way to solve the discrete logarithm problem. However, these methods are a relatively general and simplistic attack on the discrete logarithm problem. If you know more information about a specific discrete logarithm problem such as the group that the discrete logarithm problem is set within or the factorization of the order of the group there are more efficient ways to solve the discrete logarithm problem. However, square root attacks are still very useful. Since they are such a rudimentary attack on encryption schemes that rely on the discrete logarithm problem, we can assume that the time these attacks take to work will be the lower security bound. In other words, the encryption scheme is vulnerable to attacks that take, at the longest time $\mathcal{O}(\sqrt{n})$.

Another consideration when preforming complex calculations required to break an encryption scheme is space. The amount of space required to complete the calculations in each of the four methods varies. This is the factor that people use to distinguish which method is the best. It is also a component in each method that leaves room for improvement through further mathematical exploration.

3.4.1 *Enumeration*

The first way one might propose to solve the DLP would be to calculate positive powers of g until the result is congruent to h modulo n.

Method

To preform this method, calculate $g^i \equiv h \pmod{n}$ for some i i.e.

$$g^0, g^1, \ldots, g^{n-1}.$$

Since g generates the group, this method will eventually return h.

Evaluation

This method is clearly brute-force trial and error. It is definitely effective, however for values of n that are large, this will take much too much time[2]. It is only effective to use this method when n is small or prime factors of n are small. It clear why n has to be small. As for n's prime factors, it suffices to compute the discrete logarithm modulo each of the prime powers that divide n, and then use the Chinese Remainder Theorem to solve for the discrete logarithm of the main problem. This explained more in depth in section 3.5. On average this method takes takes $n/2$ group operations. The main advantage of this method is that it takes no storage.

3.4.2 *Shank's Baby-Step, Giant-Step Method*

In 1973 Mathematician Daniel Shank published a method known as Shank's Baby-Step, Giant-Step Method. His method was published prior to the Diffie-Hellman paper and thus purely for mathematical exploration.

The method is slightly more sophisticated than enumeration. It is created by designing and generating two lists of elements of G. Once an element appears on both lists, we

2 But what does too much time mean? For an example, consider the largest academic computation, 2^{65} bits operation. The heat dispersal from this calculation could heat two Olympic sized pools filled with water until boiling point.

can use information about that element to solve for x, the index of the discrete logarithm problem. This method has broad applications because it works for any cyclic group. It requires $\mathcal{O}(\sqrt{n})$ group calculations and uses $\mathcal{O}(\sqrt{n})$ space.

Method

To begin constructing the lists, notice that by the division algorithm we can write

$$x = a + (bm)$$

when $m = \lceil \sqrt{n} \rceil$ and $a, b \in \mathbb{Z}$, $a, b < m$.

Then we can substitute this new form of x into the DLP, such that $h = g^x = g^{a+(bm)}$. After rearranging this equation, we arrive at the inspiration for the two lists:

$$g^a = h\left(g^{-bm}\right)$$

The lists are generated by varying the values of a and b as follows:

Baby Step:

$$\left\{g^i \;\middle|\; 0 \leq i < m\right\} = \left\{1, g, g^2, \ldots, g^{m-1}\right\}$$

Giant Step:

$$\left\{h\left(g^{-jm}\right) \;\middle|\; 0 \leq j < m\right\} = \left\{h, hg^{-m}, hg^{-2m}, \ldots, hg^{-(m-1)m}\right\}$$

You will notice that both lists have m elements in them. The first list, 'Baby Step', moves through the powers of g in an orderly fashion. It passes by each power of g sequentially. This is that same as the enumeration method, except the list terminates after m group elements have been generated. The 'Giant Step' list, in contrast, makes much larger bounds through the group. One can make sense of this visualization because the number of group elements between $h\left(g^{-0m}\right)$ and $h\left(g^{-1m}\right)$ should be more than 1 group element in most cases. This method is a slightly more chaotic version of enumeration that will terminate after m group elements have been generated.

If a group element appears on both lists we call it a collision. A collision is when

$$g^i = h\left(g^{-jm}\right)$$

for some i, j. In this case, i does not have to equal j because we are only interested in the fact a collision occurred, not if it happened in the same position in the list. Using the information from the collision, we can rearrange to find $h = g^i\left(g^{jm}\right) = g^{i+jm}$. Thus,

$$\log_g h = x = i + jm.$$

Therefore when a collision occurs, we can solve the discrete logarithm problem.

Even though the lists are only m elements long, the information that they provide will lead us to every group element at least once. When calculating $i + jm$ for each pair of i and j we will produce every group element of G. We can organize these calculations in the following table in which the entries are $i + jm$ evaluated at the corresponding value of i and j.

	$i = 0$	$i = 1$	$\dots$	$i = m - 1$
$j = 0$	0	1	$\dots$	$m - 1$
$j = 1$	m	$m + 1$	$\dots$	$2m - 1$
$j = 2$	$2m$	$2m + 1$	$\dots$	$3m - 1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$j = m - 1$	$m^2 - m$	$m^2 - m + 1$	$\dots$	$(m^2 - m) + m - 1$

We can see that $(m^2 - m) + m - 1 = n - 1$. Remember that n is the order of g. The table entries range from 0 to $n - 1$, and thus when g is raised to each entry we will generate the whole of G. Therefore we will be able to solve the discrete logarithm problem for any value of h.

For this method to work, it is essential there is a collision. We can guarantee a collision because h will come from G. Therefore there must be some a_h and b_h such that $h = g^{a_h + b_h m}$ where $a_h, b_h < m$. By the construction of our lists we will eventually have g^{a_h} from the Baby Steps list and $hg^{-b_h m}$ from the Giant Steps list. As we have shown, these are equal.

Example 3.4.1. We will demonstrate this method with an example. Our example is set in $\mathbb{F}_{13}^*$ because of its cyclic nature.

Find the index of the discrete logarithm problem

$$2^x = 3 \in \mathbb{F}_{13}^*.$$

We notice that 2 is a generator of $\mathbb{F}_{13}^*$, thus this is a valid discrete logarithm problem. We then calculate $\lceil \sqrt{n} \rceil = \left\lceil \sqrt{12} \right\rceil = 4 = m$. The lists will be generated as follows:

Baby Steps:

$$\left\{ g^i \;\middle|\; 0 \le i < 4 \right\} = \left\{ 2^i \pmod{13} \;\middle|\; 0 \le a < 4 \right\} = \{1, 2, 4, 8\}.$$

Giant Steps:

$$\left\{ h\left(g^{-jm}\right) \;\middle|\; 0 \le j < m \right\} = \left\{ 3\left(2^{-j4}\right) \pmod{13} \;\middle|\; 0 \le j < 4 \right\}$$

$$= \left\{ 3\left(7^{j4}\right) \pmod{13} \;\middle|\; 0 \le j < 4 \right\} = \{3, 1, 9, 3\}.$$

When calculating the giant steps, we found the inverse of the group element 2 in $\mathbb{F}_{13}^*$.

In 'Baby Steps', when $i = 0$, the formula returns 1. Similarly, in 'Giant Steps', when $j = 1$, the formula returns 1. Thus $x \equiv 0 + 1(4) \equiv 4 \pmod{13}$. Indeed, $2^4 = 3 \in \mathbb{F}_{13}^*$

In practice people do not store both the Baby-Steps and the Giant-Steps list. They will generate and store elements from one list then check elements from the other list against that information. Thus they store $m + 1$ elements.

Evaluation

Similar to enumeration, this method also relies on a brute force calculations. The maximum number of calculations will be $\lceil \sqrt{n} \rceil \times 2 < n$. Thus we are likely to find the index of the discreet logarithm problem faster with this method than enumeration. However, compared to the Pollard Rho Method, Shank's Baby-Step, Giant-Step Method uses a massive amount of space and thus is rarely used in practice.

3.4.3 *Pollard Rho Method*

In 1978 John Pollard came out with his paper *Monte Carlo Methods for Index Computations*. This paper provided a new way of solving the discreet logarithm problem. The method relies on creating a pseudo-random, deterministic sequence of group elements. The sequence will eventually repeat a group element, and from that point on, every group element in the sequence will follow the same pattern as before, forming a cycle. This method is often visualized with a rho, ρ, with the circular part as the cycle. The tail of the ρ corresponds to the group elements that are generated prior to finding the start of the cycle.

To save computational energy and valuable time, it would be advantageous that the cycle forms relatively quickly. Pollard was able to ensure this. He recognized how the Birthday Problem relates to our context. The Birthday Problem is as follows:

In a group of 23 people, there is a 50% chance of 2 people having the same birthday.

The birthday problem informed Pollard that after $\sqrt{2n \ln 2}$ group elements are generated there is a 50% chance of a repeat.

This is because the general form of the birthday problem, adjusted to our variables, states:

$$P(x, n) \approx 1 - e^{\frac{-x(x-1)}{2n}} \quad \text{when } x \ll n.$$

When we desire the probability $P(x, n) > \frac{1}{2}$, we find

$$e^{\frac{-x(x-1)}{2n}} < \frac{1}{2}$$

$$\frac{-x(x-1)}{2n} < \ln\frac{1}{2}$$

$$x < \sqrt{2n\ln 2}$$

Since $\sqrt{2n\ln 2}$ is $\mathcal{O}(\sqrt{n})$, by definition of big O notation, this method takes time $\mathcal{O}(\sqrt{n})$.

Pollard's Rho Method is not ensured to work in any group. To guarantee success the group must be of prime order. Later in the section, we will reveal specifically why the method can fail.

Method

This method is reliant on creating a pseudo-random, deterministic sequence of group elements. There are many different functions that will result in such a sequence. This is an area that has been explored by mathematicians since Pollard released his paper. We will use the function that he presented in his paper.

To create the sequence, partition G into three subsets, G_1, G_2, G_3. Note that these are partitions, not subgroups. Then, select a random first element of the sequence by choosing $x_0 \in 0, \ldots, n-1$ and computing $v_0 = g^{x_0}$. We then can define $f : G \to G$ such that

$$f(v_{i+1}) = \begin{cases} gv_i & \text{if } v_i \in G_1 \\ v_i^2 & \text{if } v_i \in G_2 \\ hv_i & \text{if } v_i \in G_3 \end{cases}$$

This function works quickly and efficiently by generating the subsequent element of the sequence based on the previous element. However, to calculate x, we need to have information about the group elements.

To understand how to solve the discrete logarithm problem with this method, observe that we can write all elements of G as $v_i = g^a h^b$ for $v_i \in G$, $a, b \in \mathbb{Z}$ since G is a cyclic group generated by g.

When a collision occurs in the sequence, and $v_i = v_j$, then we write the equality as

$$g^{a_i} h^{b_i} = g^{a_j} h^{b_j}$$

for $a_i, b_i, a_j, b_j \in \mathbb{Z}$.

This can be rearranged to be

$$h = g^{(a_i - a_j)(b_j - b_i)^{-1}}.$$

Thus we can solve for the index of the discreet logarithm problem,

$$x = \log_g h = (a_i - a_j)(b_j - b_i)^{-1} \pmod{n}.$$

To ensure this is valid equality, $(b_j - b_i)^{-1} \pmod{n}$ must exist. This is ensured when n is prime. In all other cases, it suffices to show that $\gcd\left((b_j - b_i)^{-1}, n\right) = 1$.

We are guaranteed a collision, because the sequence will eventually exhaust every group element in G and then will be forced to repeat. In this case we could visualize the sequence with a very big circle instead of a rho.

Thus we can create two additional functions that will keep track of the powers of g and h, a and b, respectively for each group element generated by the original function. We note that by construction of v_0, $a_0 = x_i$ and $b_0 = 0$.

$$g(a_{i+1}) = \begin{cases} a_i + 1 \pmod{n} & \text{if } v_i \in G_1 \\ 2a_i \pmod{n} & \text{if } v_i \in G_2 \\ a_i \pmod{n} & \text{if } v_i \in G_3 \end{cases}$$

$$k(b_{i+1}) = \begin{cases} b_i \pmod{n} & \text{if } v_i \in G_1 \\ 2b_i \pmod{n} & \text{if } v_i \in G_2 \\ b_i + 1 \pmod{n} & \text{if } v_i \in G_3 \end{cases}$$

The true benefit from this method comes from the ability to store simply two group elements! Without this advantage, Pollard's Rho Method would require more storage than Shank's Baby-Step, Giant-Step because we keep track of three pieces of information for each group element generated. This advantage help decrease the amount of storage required, which can get quite hefty when n is large. However, with modern computing this is less of an issue.

Pollard took advantage of the cyclical form of the sequence and used Floyd's cycle finding algorithm to justify why we can store just two elements and their associated values of the functions g and k. Floyd's cycle finding algorithm informs us that instead of storing every group element until we find a repeating element, we only need to store two.

Lemma 3.4.2. *Let l_c denote the cycle length and l_t denote the tail length of a rho-shaped sequence. $v_i = v_{2i}$ if and only if $l_c | i$ and $i \geq l_t$. Further, there exists some i such that $l_t \leq i \leq l_t + l_c$ and $v_{2i} = v_i$.*

Proof. Let l_c denote the cycle length and l_t denote the tail length of a rho-shaped sequence. Let $v_i = v_{2i}$. Since v_i and v_{2i} are distinct points along the sequence this implies that v_i and v_{2i} are one full cycle apart from each other and thus $l_c|(2i - i)$. It follows that $l_c|(i)$. By the definition of l_t, $i \geq l_t$. This logic works in reverse, so $v_i = v_{2i}$ if and only if $l_c|i$ and $i \geq l_t$.

Furthermore, between l_t and $l_t + l_c$, there is some multiple of l_c, namely $(1)l_c$. Thus, there exists some i such that $l_t \leq i \leq l_t + l_c$ and $v_{2i} = v_i$. □

Example 3.4.3. We will again use $\mathbb{F}_{13}^*$ to demonstrate how Pollard's Rho method will be used to solve the discrete logarithm problem.

Find the index of the discrete logarithm problem

$$2^x = 8 \in \mathbb{F}_{13}^*.$$

In this case, 2 generates $\mathbb{F}_{13}^*$, thus this is a valid DLP.

We will partition G in the following way:

$$G_1 = \{1, 2, 3, 4\}$$
$$G_2 = \{5, 6, 7, 8\}$$
$$G_3 = \{9, 10, 11, 12\}$$

The first element of the sequence is selected at random. Let $x_0 = 2$, and thus $v_0 \equiv 2^2 \equiv 4 \pmod{13}$.

We will use the functions previously stated in this chapter to generate the sequence and the representation of group elements in the $g^a h^b$ form. As a reminder they are:

$$f(v_{i+1}) = \begin{cases} gv_i & \text{if } v_i \in G_1 \\ v_i^2 & \text{if } v_i \in G_2 \\ hv_i & \text{if } v_i \in G_3 \end{cases}$$

$$g(a_{i+1}) = \begin{cases} a_i + 1 \pmod{n} & \text{if } v_i \in G_1 \\ 2a_i \pmod{n} & \text{if } v_i \in G_2 \\ a_i \pmod{n} & \text{if } v_i \in G_3 \end{cases}$$

$$k(b_{i+1}) = \begin{cases} b_i \pmod{n} & \text{if } v_i \in G_1 \\ 2b_i \pmod{n} & \text{if } v_i \in G_2 \\ b_i + 1 \pmod{n} & \text{if } v_i \in G_3 \end{cases}$$

We are now prepared to generate the sequence and look for a cycle. We will elect to not use the advantage of Floyd's cycle finding technique. The way in which the cycle generates requires us to calculate each subsequent element. Since we are not a computer, and storage is not a concern, there is no advantage to Floyd's method.

We begin and generate the following information:

$$\begin{array}{lll} v_0 = 4 & a_0 = 2 & b_0 = 0 \\ v_1 = 2(4) \equiv 8 \pmod{13} & a_1 = 3 & b_1 = 0 \\ v_2 = 8^2 \equiv 12 \pmod{13} & a_2 = 6 & b_2 = 0 \\ v_3 = 8(12) \equiv 5 \pmod{13} & a_3 = 6 & b_3 = 1 \\ v_4 = 5^2 \equiv 12 \pmod{13} & a_4 = 12 = 0 \pmod{12} & b_4 = 2 \end{array}$$

We have a collision at $v_2 = v_4 = 12$. Thus we have the following information

$$g^{a_2} h^{b_2} \equiv g^{a_4} h^{b_4} \pmod{n}$$

$$2^6 8^0 \equiv 2^{12} 8^2 \pmod{12}$$

Thus we can solve for the index of the discreet logarithm problem,

$$x = \log_g h = (a_2 - a_4)(b_4 - b_2)^{-1} \pmod{n}.$$

$$x = \log_g h = (6 - 0)(2 - 0)^{-1} \pmod{12}.$$

We have been quite unlucky and reached a problem. $2^{-1} \pmod{12}$ does not exist because $\gcd(2, 12) \neq 1$. This is an example of how Pollards Rho Method can fail.

Evaluation

Pollard's Rho method requires dramatically less space than Shank's Baby-Step, Giant-Step Method and works in the same amount of time. It cleverly uses the birthday problem to expedite the brute-force calculation method in a different way than Shank's Baby-Step, Giant-Step Method. This is a powerful method and has been explored further in order to continue to improve performance through function selection, group partitioning, and cycle detection criteria.

3.4.4 *Pollard's Kangaroo Method*

In the same paper, *Monte Carlo Methods for Index Computations*, Pollard introduced the 'Kangaroo' or 'Lambda' method. This method is primarily referred to as the Lambda Method, however, Pollard's dedication to the Kangaroo visualization was clear in his

paper. It is riddled with many references on ways to catch a kangaroo in a well disguised trap.

This method is visualized by two paths that become one. A tame kangaroo jumps along a path of group elements, and a wild kangaroos jumps along a separate path of group elements. Once the kangaroos jump on the same element, but not necessarily at the same time, we have the information needed to solve the discrete logarithm problem. The hopping pattern is generated deterministically, so once the kangaroos start to land on the same group elements, they will continue to jump together. In my words, the kangaroos become inseparable. In Pollard's words, the wild kangaroo has been caught.

This method lends itself to being used when we have a known range for the value of x. We will say that $a < x < b$ for some integers a, b modulo n. This variation runs in time $\mathcal{O}\big(\sqrt{b-a}\big)$ and requires $\mathcal{O}\big(\log(b-a)\log n\big)$ space.

Method

To begin the process of solving discrete logarithm problem with the Kangaroo method, we must meet the kangaroos.

We name the tame Kangaroo T. This kangaroo is tame because we know the discreet logarithm for its starting point. This starting point can be random or it could be strategically picked to be within the interval between a and b. For our purposes we will select a random x_0 from 0 to $n-1$ then calculate $g^{(x_0)} = v_0$. This will be the tame kangaroo's starting point. Alternatively, x_0 could equal $\frac{a-b}{2}$, but we will not choose that option for this explanation. The wild kangaroo is called W, naturally. We assign the wild kangaroo to start at h, and thus we do not know the discrete logarithm of the starting point of the wild kangaroo. Their walks are defined as such:

- T hops on group elements denoted t_i
  - T starts their walk at $t_0 = v_0$
  - $t_{i+1} = t_i g^r$ for some $r \in \mathbb{Z}$
- W hops on group elements denoted w_i
  - W starts their walk at $w_0 = h$.
  - $w_{i+1} = w_i g^s$ for some $s \in \mathbb{Z}$

The values of r and s determine step sizes and are generated through a pseudo-random function, $f : G \to \mathbb{Z}_n$. Thus the jump length for the kangaroos is function of the state of the group element that the kangaroo is jumping from. There are many ways to optimize the function, however, a simple example would be analogous to the one shown in Pollard's Rho method. We will demonstrate an even more simplistic function in the example of this method.

When a kangaroo lands on an element v_i, the next element it lands on will be $v_i g^{f(v_i)}$. We say the kangaroo has traveled a distance of $f(v_i)$. A visualization of this is provided in figure 4.

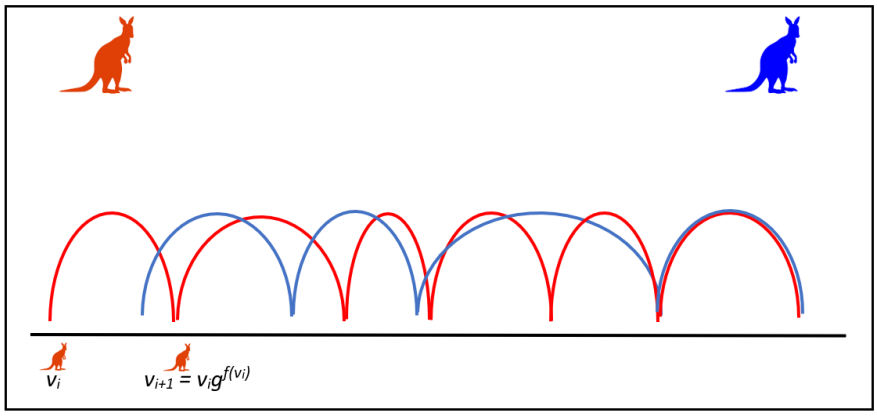


Figure 4.: Visualization of Pollard's Kangaroo Method typical hopping pattern

It is easy to confuse the word distance with the difference between group elements. This is incorrect; distance is a power of g. Notice that after k hops the kangaroo will have traveled a distance of

$$\sum_{i}^{i+k} f(v_i)$$

Using this information we will know exactly how far each kangaroo has travelled from their starting point until they meet - and where love happens at first sight. Once the kangaroos meet, we can work backwards to find the index of the discrete logarithm problem.

If the the wild and tame kangaroo meet up, we will have $t_i = w_j$ for some i, j where i does not have to equal j. When $t_i = w_i$ we can calculate t_{i+1} and w_{i+1} to be $t_i g^r$ and $w_i g^s$ respectively. We note that r and s are generated by the same function that is dependent on the group element the kangaroo is currently on. So it is true that when $t_i = w_i$, then $t_{i+1} = w_{i+1}$.

Substituting in the information from the walk to that point they meet we have:

$$t_i = w_j g^{x_i} g^{d_t} = h g^{d_w}$$

where $d_t = \sum_{i=t_0}^{t_i} f(v_i)$ and $d_w = \sum_{j=w_0}^{w_j} f(v_j)$. Therefore we can solve for

$$x = \log_g h = x_i + d_t - d_w.$$

There is the unfortunate case that the kangaroos do not meet up. This is highly unlikely due to the pseudo-random nature of f, and one solution to this problem is changing the function. When the kangaroo method fails, and we have been quite unlucky, it could look something like the visualization in figure 5.

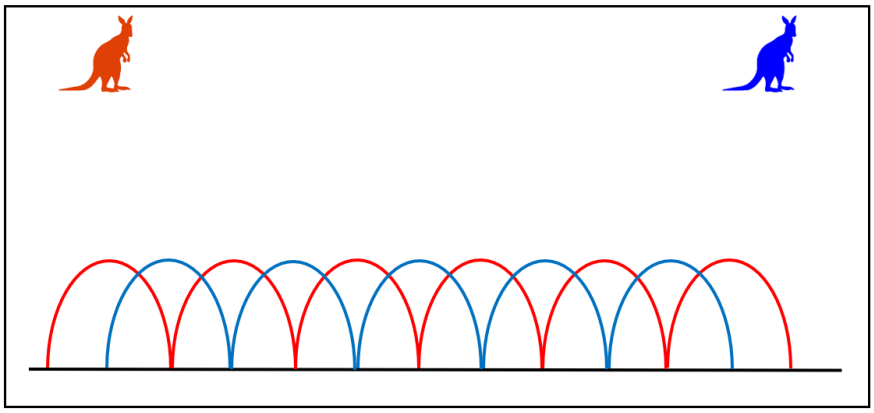


Figure 5.: Visualization of Pollard's Kangaroo Method hopping pattern in the case we have been quite unlucky

Example 3.4.4. To demonstrate Pollard's Kangaroo method, we will continue to use the simple group $\mathbb{F}_{13}^*$.

Find the index of the discrete logarithm problem

$$2^x = 5 \in \mathbb{F}_{13}^*.$$

This is a valid DLP since 2 us a generator of $\mathbb{F}_{13}^*$. We then select a random integer between 0 and $n-1$ to be x_0. Let $x_0 = 3$, and thus $v_0 = 2^3 = 8$.

We now need a pseudo-random function that maps from G to $\mathbb{Z}_n$ to calculate the hop distance. Let

$$f(v_{i+1}) = v_i - 2 \pmod{13}.$$

Note that this is an appropriate mapping in our group setting, however it will not work with group elements that are not integers.

We now can calculate our kangaroos' walks. We will do this simultaneously to minimize the amount of time it takes to complete this method.

By construction $t_0 = 8$ and $w_0 = 5$. We find

$$t_1 \equiv t_0 g^r \equiv t_0 g^{f(v_0)} \equiv 8(2^{8-2}) \equiv 8(2^6) \equiv 5 \pmod{13}.$$

We continue in this fashion, using f to calculate the hopping distance for both the tame and wild kangaroo until the kangaroos meet.

We generate the following lists:

$$T : t_0 = 8, t_1 = 5, t_2 = 1$$

$$W : w_0 = 5, w_1 = 1$$

The kangaroos have met at $t_1 = w_0$, and are now inseparable!

We have kept track of the hopping distance for the tame and wild kangaroos and found that

$$d_t = \sum_{i=t_0}^{t_1} f(v_i) = 6$$

and

$$d_w = \sum_{j=w_0}^{w_0} f(v_j) = 0$$

We can see that this is a surprisingly simplistic example since the kangaroos met so quickly. This will not happen in every case.

Therefore with this information, we can solve for

$$x = \log_g h = x_i + d_t - d_w.$$

$$x \equiv 3 + 6 - 0 \pmod{12}$$

$$x = 9$$

Indeed $2^9 = 5 \in \mathbb{F}_{13}^*$.

Evaluation

Similar to Pollard's Rho Method, Pollard's Kangaroo Method is an interesting exploration into solving the discrete logarithm problem but is not used that often. We improve the brute force guess and check method by finding two group elements that match rather than just the single one that equals h. However, we note that this method is not guaranteed to work every time and it could take up a lot of space.

Pollard's Kangaroo Method leaves room for improvement on number of kangaroos and jumping function. These methods have been explored extensively since Pollard's initial paper.

3.5 THE CHINESE REMAINDER THEOREM

How do you find the solution to a system of simultaneous linear congruences. The first recorded instance of a problem of this type appears in a Chinese mathematical work by Sun Tzu from the late third of fourth century. It was later republished in a 1247 book by Qin Jiushao, the Shushu Jiuzhang (Mathematical Treatise in Nine Sections).

> We have a number of things, but we do not know exactly how many. If we count them by threes, we have two left over. If we count them by fives, we have three left over. If we count them by sevens, we have two left over.
>
> -Sun Tzu Suan Ching (Master Sun's Mathematical Manual, volume 3, problem 26)

Figure 6.: Mathematical Treatise in Nine Sections

A similar question appears in Brahmagupta's Brahma-Sphuta-Siddhanta (Brahma's Correct System), from the sixth century:

> An old woman goes to market and a horse steps on her basket and crashes the eggs. The rider offers to pay for the damages and asks her how many eggs she had brought. She does not remember the exact number, but when she had taken them out two at a time, there was one egg left. The same happened when she picked them out three, four, five, and six at a time, but

when she took them seven at a time they came out even. What is the smallest number of eggs she could have had?

Problems of this kind are all examples of what universally became known as the Chinese Remainder Theorem. The Chinese Remainder Theorem and its generalizations have many applications in number theory and other areas of mathematics.

Theorem 3.5.1 (Chinese Remainder Theorem)**.** *Let $m_1, m_2, \ldots, m_k$ be a collection of pairwise relatively prime integers (i.e.* $\gcd(m_i, m_j) = 1 \quad \forall i \neq j$*). Let $a_1, a_2, \ldots, a_k$ be arbitrary integers. The system of simultaneous congruences*

$$x \equiv a_1 \pmod{m_1}, \quad x \equiv a_2 \pmod{m_2}, \quad \ldots, \quad x \equiv a_k \pmod{m_k}$$

has a solution. Furthermore, if $x = c$ is one solution and $x = c'$ is another, then

$$c \equiv c' \pmod{m_1 m_2 \cdots m_k}.$$

The following example demonstrates how to find the solution of a system of simultaneous congruences.

Example 3.5.2. We solve the three simultaneous congruences

$$x \equiv 2 \pmod{3}, \quad x \equiv 3 \pmod{7}, \quad x \equiv 4 \pmod{16}.$$

The Chinese Remainder Theorem tells us that there is a unique solution modulo 336. We start with the solution $x_1 = 2$ to the first congruence $x \equiv 2 \pmod{3}$. We note $x_2 = 2 + 3y$ is the general solution to the first congruence. Substituting x_2 into the second congruence, and solving for y we find

$$\begin{aligned} 2 + 3y &\equiv 3 \mod 7 \\ 3y &\equiv 1 \mod 7 \\ y &\equiv 3^{-1} \cdot 1 \mod 7 \\ y &\equiv 5 \mod 7 \end{aligned}$$

Note that we know by 1.2.19 that 3^{-1} exists since $\gcd(3,7) = 1$. Thus $x_2 = 2 + (3 \cdot 5) = 17$ is a solution to the first two congruences. Furthermore, by the Chinese Remainder Theorem, every solution of these first two congruences has the general form $x_3 = 17 + 21z$. Substituting into the third congruence and solving for z, we find

$$\begin{aligned} 17 + 21z &\equiv 4 \mod 16 \\ 1 + 5z &\equiv 4 \mod 16 \\ 5z &\equiv 3 \mod 16 \\ z &\equiv 5^{-1} \cdot 3 \mod 16 \\ z &\equiv 13 \cdot 3 \mod 16 \\ z &\equiv 7 \mod 16 \end{aligned}$$

Note that we know by 1.2.19 that 5^{-1} exists since $\gcd(5, 16) = 1$.

Substituting back into x_3, we find the unique solution $x_3 = 17 + (21 \cdot 7) = 164 \mod 336$.

The following theorem is a algebraic reformulation of the Chinese Remainder Theorem, and is often called the *Modern Chinese Remainder Theorem*.

Theorem 3.5.3 (Modern Chinese Remainder Theorem). *Let $m_1, m_2, \ldots, m_k$ be a collection of pairwise relatively prime integers. There exists a ring isomorphism*

$$f : \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \times \ldots \times \mathbb{Z}/m_k\mathbb{Z} \to \mathbb{Z}/m_1m_2 \ldots m_k\mathbb{Z}$$

with the property that if $f(a_1, a_2, \ldots, a_k) = x$ then

$$x \equiv a_1 \pmod{m_1}, \quad x \equiv a_2 \pmod{m_2}, \quad \ldots, \quad x \equiv a_k \pmod{m_k}.$$

Proof. Let $m_1, m_2, \ldots, m_k$ be a collection of pairwise relatively prime integers and consider the function $f : \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \times \ldots \times \mathbb{Z}/m_k\mathbb{Z} \to \mathbb{Z}/m_1m_2 \ldots m_k\mathbb{Z}$ with the property that if $f(a_1, a_2, \ldots, a_k) = x$ then $x \equiv a_1 \pmod{m_1}$, $x \equiv a_2 \pmod{m_2}$, $\ldots$, $x \equiv a_k \pmod{m_k}$.

First, we must show that the function is well-defined. Consider $f(a_1, a_2, \ldots, a_k) = x$ and $f(a_1, a_2, \ldots, a_k) = y$ for some $x, y \in \mathbb{Z}/m_1m_2 \cdots m_k\mathbb{Z}$. By way of contradiction suppose $x \not\equiv y \pmod{m_1m_2 \ldots m_k}$. By the definition of modular congruence, there exists some $q_x, q_y, r_x, r_y \in \mathbb{Z}$ such that $x = q_x(m_1m_2 \cdots m_k) + r_x$ and $y = q_y(m_1m_2 \cdots m_k) + r_y$ where $r_x \neq r_y$ and $0 \leq r_x, r_y < m_1m_2 \cdots m_k$. By construction, $x \equiv a_i \equiv y \pmod{m_i}$ for all $1 \leq i \leq k$, and so $r_x \equiv r_y \pmod{m_i}$. By definition of modular congruence therefore $m_i | (r_x - r_y)$. Since $m_1, m_2, \ldots, m_k$ are pairwise relatively prime, by Remark 1.1.14 it follows that $m_1m_2 \cdots m_k | (r_x - r_y)$. Since $r_x < m_1m_2 \cdots m_k$ and $r_y < m_1m_2 \cdots m_k$, clearly $r_x - r_y < m_1m_2 \cdots m_k$. Since $m_1m_2 \cdots m_k | (r_x - r_y)$ and $r_x - r_y < m_1m_2 \cdots m_k$, then $r_x - r_y$ must equal zero. This implies that $r_x = r_y$, and thus we have reached a contradiction. Therefore $x \equiv y \pmod{m_1m_2 \ldots m_k}$ and so the function is well defined.

To prove that the function is operation preserving, consider $f(a_1, \ldots, a_k) + f(b_1, \ldots, b_k)$ in $\mathbb{Z}/m_1 \cdots m_k\mathbb{Z}$. Where $f(a_1, \ldots, a_k) = x$ and $f(b_1, \ldots, b_k) = y$. Notice that the function is well-defined. By construction of the function, $x \equiv a_i \pmod{m_i}$ and $y \equiv b_i \pmod{m_i}$ for all $1 \leq i \leq k$. By 1.2.19, $x + y \equiv a_i + b_i \pmod{m_i}$. Thus $f(a_1, \ldots, a_k) + f(b_1, \ldots, b_k) = f\big((a_1, \ldots, a_k) + (b_1, \ldots, b_k)\big)$. Similarly, consider $f(a_1, \ldots, a_k)f(b_1, \ldots, b_k)$ in $\mathbb{Z}/m_1 \cdots m_k\mathbb{Z}$. Where $f(a_1, \ldots, a_k) = x$ and $f(b_1, \ldots, b_k) = y$. By construction of the function, $x \equiv a_i \pmod{m_i}$ and $y \equiv b_i \pmod{m_i}$ for all $1 \leq i \leq k$. By 1.2.19, $x * y \equiv a_i * b_i \pmod{m_i}$. Thus $f(a_1, \ldots, a_k)f(b_1, \ldots, b_k) = f\big((a_1, \ldots, a_k)(b_1, \ldots, b_k)\big)$.

To prove the function is injective consider $f(a_1, a_2, \ldots, a_k) \equiv x \equiv f(b_1, b_2, \ldots, b_k) \pmod{m_1m_2 \cdots m_k}$ where $x \in \mathbb{Z}/m_1m_2 \cdots m_k\mathbb{Z}$. By construction, $a_i \equiv x \equiv b_i \pmod{m_i}$

for all $1 \leq i \leq k$. Since $(a_1, a_2, \ldots, a_k)$ and $(b_1, b_2, \ldots, b_k)$ are component-wise congruent, they are equal in $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}$ and so it follows that the function is injective.

To prove the function is surjective, let x be an arbitrary element of $\mathbb{Z}/m_1m_2 \cdots m_k\mathbb{Z}$. If we consider $x \pmod{m_i}$, clearly there exists $a_1, a_2, \ldots, a_k$ such that $x \equiv a_i \pmod{m_i}$ for all $1 \leq i \leq k$. By construction, therefore $f(a_1, a_2, \ldots, a_k) = x$, and so it follows that the function is surjective.

Thus it has been shown that the function is a ring isomorphism. □

We now have the tools necessary to prove Theorem 1.2.32.

Theorem 3.5.4. *(1.2.32) Given integers m and n such that* $\gcd(m, n) = 1$,

$$\phi(mn) = \phi(m)\phi(n).$$

Proof. Let m, n be integers such that $\gcd(m, n) = 1$. Since m and n are relatively prime, $\mathbb{Z}/mn\mathbb{Z} \cong \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ by the Modern Chinese Remainder Theorem. Notice that since these rings are isomorphic, and isomorphisms preserve inverses, they have the same number of elements with inverses. Since there are $\phi(mn)$ elements with inverses in $\mathbb{Z}/mn\mathbb{Z}$, there are $\phi(mn)$ elements with inverses in $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$.

Let (a, b) be an arbitrary element $\in \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ and suppose (a, b) has an inverse (c, d). By definition of inverse,

$$(a, b)(c, d) = (ac, bd) = (e_m, e_n)$$

and

$$(c, d)(a, b) = (ca, db) = (e_m, e_n)$$

Thus $c \equiv a^{-1} \pmod{m}$ and $d \equiv b^{-1} \pmod{n}$. In other words, (a, b) has an inverse in $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ if and only if a has an inverse in $\mathbb{Z}/m\mathbb{Z}$ and b has an inverse in $\mathbb{Z}/n\mathbb{Z}$

There are $\phi(m)$ elements in $\mathbb{Z}/m\mathbb{Z}$ with inverses and $\phi(n)$ elements in $\mathbb{Z}/n\mathbb{Z}$ with inverses by definition of the Euler Totient function. Therefore, there are $\phi(m)\phi(n)$ element pairs in $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ with inverses. Since we already stated that the number of elements with inverses in $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ was equal to $\phi(mn)$, it follows that $\phi(mn) = \phi(m)\phi(n)$. □

3.5.1 *Solving Congruences with Composite Moduli*

It is usually easiest to solve congruences with a composite modulus by first solving several congruences modulo primes (or prime powers) and then stitching together a solution to the composite modulus problem using the Chinese remainder theorem. We

will illustrate this process by examining the problem of finding square roots modulo m. For primes congruent to 3 modulo 4, it is extremely easy to find square roots.

Theorem 3.5.5. *Let p be a prime satisfying $p \equiv 3 \pmod 4$. Let a be an integer such that a has a square root modulo p, then*

$$b \equiv a^{(p+1)/4} \pmod p$$

is a solution to the equation $x^2 \equiv a \pmod p$. Note: this formula is only valid if a has a square root modulo p.

Proof. Let p be a prime such that $p \equiv 3 \mod 4$, and $a \in \mathbb{Z}$ such that a has a square root mod p. Consider $b \equiv a^{(p+1)/4} \mod p$, and note that $p \equiv 3 \mod 4$ ensures that the exponent $(p+1)/4$ is an integer. Now, notice that

$$b^2 \equiv (a^{(p+1)/4})^2 \equiv a^{(p+1)\frac{1}{2}} \equiv a^{1+(p-1)\frac{1}{2}} \equiv aa^{(p-1)\frac{1}{2}} \pmod p$$

If it is the case that $p|a$, then $a \equiv 0 \pmod p$ and by Theorem 1.2.38 $aa^{(p-1)\frac{1}{2}} \equiv a(0)^{\frac{1}{2}} \equiv 0 \equiv a \pmod p$. If $p \nmid a$, again by Theorem 1.2.38 $aa^{(p-1)\frac{1}{2}} \equiv a(1)^{\frac{1}{2}} \equiv a \pmod p$. Therefore, in all cases $b \equiv a^{(p+1)/4} \pmod p$ is a solution to the equation $x^2 \equiv a \pmod p$. □

Theorem 3.5.6. *Let $n = pq$ where p and q are both odd primes. If the $\gcd(a, pq) = 1$, then if the equation $x^2 \equiv a \pmod n$ has any solutions, it must have four distinct solutions modulo n.*

Proof. Let $n = pq$, where p and q are both primes. Suppose that $\gcd(a, pq) = 1$ and that the equation $x^2 \equiv a \pmod n$ has solutions. Using the Modern Chinese Remainder Theorem, it follows that there exists $f : \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ where f is a ring isomorphism. It follows that there exists $a_1 \in \mathbb{Z}/p\mathbb{Z}$ and $a_2 \in \mathbb{Z}/q\mathbb{Z}$ such that $f(a_1, a_2) = (a)$. Let b be a solution to the equation $x^2 \equiv a \pmod n$. Again, there exists $b_1 \in \mathbb{Z}/p\mathbb{Z}$ and $b_2 \in \mathbb{Z}/q\mathbb{Z}$ where $f(b_1, b_2) = b$. Since f preserves operations, it also follows that $(b_1, b_2)^2 = (a_1, a_2)$ and so $b_1^2 \equiv a_1 \pmod p$ and $b_2^2 \equiv a_2 \pmod q$. From this, it also follows that $(-b_1)^2 \equiv a_1 \pmod p$ and $(-b_2)^2 \equiv a_2 \pmod q$.

Notice then that $(b_1, b_2), (-b_1, -b_2), (b_1, -b_2), (-b_1, b_2)$ all square to (a_1, a_2), and so $f(b_1, b_2), f(-b_1, -b_2), f(b_1, -b_2), f(-b_1, b_2)$ must all square to a. By way of contradiction, suppose $b_1 \equiv -b_1 \pmod p$. This implies that $b_1 + b_1 \equiv 0 \pmod p$, and so the additive order of b_1 divides 2. This means the additive order of b_1 is either one or two. If it is one, this implies that $b_1 = 0$ and that $b_1^2 = a_1 = 0$. Since, by how we defined our isomporphism, $a_1 \equiv a$(mod p), this implies that $a \equiv 0$ (mod p). Since that would imply that the $gcd(a, pq) \neq 1$ the additive order of b_1 cannot be one. If it were two, by Lagranges Theorem, $2|p$. Since p was an odd prime, this is a contradiction and so b_1 cannot be congruent to $-b_1 \pmod p$. By similar process with b_2, we can conclude that each pair $(b_1, b_2), (-b_1, -b_2), (b_1, -b_2), (-b_1, b_2)$ is therefore distinct, and so we have found 4 distinct square roots of a. It remains to show that these are the only solutions.

Since p is prime, we know by Theorem 1.2.25 that the polynomial $x^2 - a_1 \pmod{p}$ has at most two solutions. We repeat the process with $x^2 - a_2 \pmod{q}$, and can thus conclude that the only square roots of a_1 are b_1 and $-b_1$, and the only square roots of a_2 are b_2 and $-b_2$. Thus the only square roots of (a_1, a_2) are our four ordered pairs, and so the only square roots of a are $f(b_1, b_2), f(-b_1, -b_2), f(b_1, -b_2), f(-b_1, b_2)$. Therefore, we have shown that $x^2 \equiv a \mod n$ has precisely four distinct solutions modulo n. □

The following example demonstrates how Theorem 3.5.5 and the Chinese remainder theorem can be used to find square roots modulo a composite number.

Example 3.5.7. We find four distinct solutions to the congruence

$$x^2 \equiv 2833 \pmod{4189}.$$

We begin by noting the prime factorization of 4189 is $59 \cdot 71$. Also, notice that $59 \equiv 3 \pmod 4$ and $71 \equiv 3 \pmod 4$. Thus, we know by Theorem 3.5.5 that $b \equiv 2833^{(59+1)/4} \equiv 1 \pmod{59}$ is a solution to $x^2 \equiv 2833 \pmod{59}$, which implies that $x \equiv \pm 1 \pmod{59}$ are both solutions to this equation. Similarly, we have that $c \equiv 2833^{(71+1)/4} \equiv 8 \pmod{71}$ is a solution to $x^2 \equiv 2833 \pmod{71}$, which implies that $x \equiv \pm 8 \pmod{71}$ are both solutions to this equation.

Now, we can use the Chinese Remainder Theorem to find the four distinct solutions to $x^2 \equiv 2833 \pmod{4189}$. First, consider the pair of congruences $x \equiv 1 \pmod{59}$ and $x \equiv 8 \pmod{71}$. We start with $x_1 = 1$ being a solution to the first congruence. Then, $x_2 = 1 + 59y$ is the general solution to the first congruence. Substituting x_2 into the second congruence, we find

$$\begin{aligned} 1 + 59y &\equiv 8 \pmod{71} \\ 59y &\equiv 7 \pmod{71} \\ y &\equiv 59^{-1} \cdot 7 \pmod{71} \\ y &\equiv 65 \cdot 7 \pmod{71} \\ y &\equiv 29 \pmod{71} \end{aligned}$$

Note that by 1.2.19, 59^{-1} exists since $\gcd(59, 71) = 1$. Substituting back into x_2, we know by the Chinese Remainder Theorem that $x_2 = 1 + (59 \cdot 29) = 1712 \pmod{4189}$ is the unique solution to our pair of simultaneous congruences.

By similar method we solve the other three paired congruences to find that the four distinct solutions to the equation $x^2 \equiv 2833 \pmod{4189}$ are: $1712 \pmod{4189}$, $2477 \pmod{4189}$, $1002 \pmod{4189}$, and $3187 \pmod{4189}$. For full details, see Appendix B.

Remark 3.5.8. If you had a machine that could find all four solutions for some given square a modulo n, you could use it to factor n.

To prove this, suppose we have a machine which, given some square a modulo n, outputs x_1, x_2, x_3, x_4 such that $x_i^2 \equiv a \mod n$ for all $1 \leq i \leq 4$. We know, by the Modern Chinese Remainder Theorem that there exists some $b_1 \in \mathbb{Z}_p$ and $b_2 \in \mathbb{Z}_q$ such that

$$x_1 = f(b_1, b_2) \quad x_2 = f(-b_1, b_2) \quad x_3 = f(-b_1, -b_2) \quad x_4 = f(b_1, -b_2).$$

If we choose two of the possible four solutions x_j, x_k which are not additive inverses (we can easily check whether or not they are inverses simply by adding them modulo n), clearly either $x_j \equiv x_k \pmod{p}$ or $x_j \equiv x_k \pmod{q}$. It follows that $p|(x_i - x_j)$ or $q|(x_i - x_j)$ by the definition of modular congruence. Since $p|n$ and $q|n$, the $\gcd(x_i - x_j, n) = p$ or $\gcd(x_i - x_j, n) = q$. Therefore by taking the $\gcd(x_i - x_j, n)$, we factor n.

3.5.2 *The Pohlig-Hellman Algorithm*

In addition to being a theorem and an algorithm, the Chinese Remainder Theorem is a state of mind. If $m = m_1 \cdot m_2 \cdots m_t$ is a product of relatively prime integers, then the Chinese Remainder Theorem says that solving an equation modulo m is equivalent to solving an equation modulo m_i for each i. By the Chinese Remainder Theorem, we can take these solutions modulo the m_i and knit them together to find the solution to the original equation.

In the multiplicative version of the discrete log problem for finite fields of order p, we need to solve the equation

$$g^x \equiv h \pmod{p}.$$

In this case, the modulus is prime, which suggests the Chinese remainder theorem is irrelevant. However, by definition of the discrete log problem, the solution x is determined modulo $p - 1$. This hints that the factorization of $p - 1$ into primes may play a role in determining the difficult of the DLP in $\mathbb{F}_p^*$. More generally, if G is any group and $g \in G$ is an element of order n, then solutions to $g^x = h$ in G are determined modulo n. This is the idea at the core of the Pohlig-Hellman algorithm.

Theorem 3.5.9 (Pohlig-Hellman Algorithm)**.** *Let G be a group, and suppose we have an algorithm to solve the DLP in G for any element whose order is a power of a prime. To be concrete, if $y \in G$ has order q^n, suppose we can solve $y^x = h$ in $\mathcal{O}(S_{q^n})$ steps. Now let $g \in G$ be an element of order N, where N has prime factorization*

$$N = q_1^{n_1} \cdot q_2^{n_2} \cdots q_t^{n_t}.$$

The discrete log problem $g^x = h$ can be solved in

$$\mathcal{O}\left(\sum_{i=1}^{t} S_{q_i^{n_i}} + \log N\right) \text{ steps}$$

using the following procedure:

- *For each* $1 \leq i \leq t$, *let*

$$g_i = g^{N/q_i^{n_i}} \qquad \text{and} \qquad h_i = h^{N/q_i^{n_i}}.$$

Notice that g_i *has prime power order* $q_i^{n_i}$, *so the given algorithm can solve the DLP*

$$g_i^{x_i} = h_i.$$

- *Use the Chinese remainder theorem to solve the simultaneous system of linear congruences*

$$x \equiv x_1 \pmod{q_1^{n_1}}, \quad x \equiv x_2 \pmod{q_2^{n_2}}, \quad \ldots, \quad x \equiv x_t \pmod{q_t^{n_t}}.$$

The Pohlig-Hellman algorithm reduces the discrete log problem for elements of arbitrary order to the discrete log problem for elements of prime power order. The following theorem provides a refinement which further reduces the problem to elements of prime order. More precisely, the running time $\mathcal{O}(S_{q^n})$ for elements of order q^n can be reduced to $\mathcal{O}(nS_q)$. Thus, given the Pohlig-Hellman algorithm, the DLP is not secure if the order of the group is a product of powers of small primes. More generally, $g^x = h$ is easy to solve if the order of the element g is a product of powers of small primes. This applies, in particular, to the discrete log problem in $\mathbb{F}_p$ if $p-1$ factors into powers of small primes.

Theorem 3.5.10. *Let* G *be a group. Suppose that* q *is a prime, and suppose that we know an algorithm that takes* S_q *steps to solve the discrete logarithm problem* $y^x = h$ *in* G *whenever* y *has order* q. *Let* $g \in G$ *be an element of order* q^n *with* $n \geq 1$. *We can solve the discrete logarithm problem* $g^x = h$ *in* $\mathcal{O}(nS_q)$ *steps.*

Proof. Let q be a prime and suppose we have an algorithm that solves the DLP in G in S_q steps for elements of order q. Let g be an element of order q^n with $n \geq 1$, and consider $g^x = h$. We begin by considering the base q expansion of the unknown exponent x, i.e.

$$x = x_0 + x_1 q + x_2 q^2 + \cdots + x_{n-1} q^{n-1} \quad \text{with } 0 \leq x_i < q.$$

To determine the unknowns $x_0, x_1, \ldots x_{n-1}$ we raise both sides of the equation $h = g^x$ to the power q^{n-1}. We find

$$\begin{aligned} h^{q^{n-1}} &= (g^x)^{q^{n-1}} \\ &= \left(g^{x_0 + x_1 q + x_2 q^2 + \cdots + x_{n-1} q^{n-1}}\right)^{q^{n-1}} \\ &= g^{x_0 q^{n-1}} \cdot \left(g^{q^n}\right)^{x_1 + x_2 q + \cdots + x_{n-1} q^{n-2}} \\ &= \left(g^{q^{n-1}}\right)^{x_0} \cdot (1)^{x_1 + x_2 q + \cdots + x_{n-1} q^{n-2}} \\ &= \left(g^{q^{n-1}}\right)^{x_0} \end{aligned}$$

Since $g^{q^{n-1}}$ is an element of order q in G, the equation $\left(g^{q^{n-1}}\right)^{x_0} = h^{q^{n-1}}$ is a discrete log problem whose base is an element of order q. By assumption, we have an algorithm to solve this problem (i.e. find x_0) in S_q steps.

Now that we know x_0, we can solve for x_1 in a similar fashion. We start by raising both sides of $h = g^x$ to the power q^{n-2}. Then, we find

$$\begin{aligned} h^{q^{n-2}} &= (g^x)^{q^{n-2}} \\ &= \left(g^{x_0+x_1q+x_2q^2+\cdots+x_{n-1}q^{n-1}}\right)^{q^{n-2}} \\ &= g^{x_0q^{n-2}} \cdot g^{x_1q^{n-1}} \cdot \left(g^{q^n}\right)^{x_2+x_3q+\cdots+x_{n-1}q^{n-3}} \\ &= g^{x_0q^{n-2}} \cdot g^{x_1q^{n-1}} \end{aligned}$$

Since we already know the value of x_0, we can find g^{-x_0}. Multiplying both sides of equation 1 by $g^{-x_0q^{n-2}}$, we find

$$\left(g^{q^{n-1}}\right)^{x_1} = g^{-x_0q^{n-2}} \cdot h^{q^{n-2}}$$

Solving this discrete log problem gives us x_1. Again, this can be done in S_q steps. Thus, in $\mathcal{O}(2S_q)$ steps, we have found values for x_0 and x_1 satisfying

$$g^{(x_0+x_1q)q^{n-2}} = h^{q^{n-2}} \qquad \text{in } G.$$

Similarly, we find x_2 by solving the DLP

$$\left(g^{q^{n-1}}\right)^{x_2} = g^{-x_1q^{n-2}} \cdot g^{-x_0q^{n-3}} \cdot h^{q^{n-3}}$$

In general, after having found $x_0, x_1, \ldots, x_{i-1}$, we obtain x_i by solving the DLP

$$\left(g^{q^{n-1}}\right)^{x_i} = h^{q^{n-1-i}} \cdot g^{-q^{n-1-i}\left(x_0+x_1q+x_2q^2+\ldots\right)}$$

Each of these discrete logarithm problems has a base of order q, so each can be solved in S_q steps. Thus after $\mathcal{O}(nS_q)$ steps, we have found $x = x_0 + x_1q + x_2q^2 + \cdots + x_{n-1}q^{n-1}$ satisfying $g^x = h$, thereby solving the original discrete logarithm problem. □

Example 3.5.11. Using the Pohlig-Hellman algorithm and the method described in the proof of Theorem 3.5.10, we solve the discrete logarithm problem

$$10^x = 243278 \quad \text{in } \mathbb{F}_{746497}.$$

Beginning by applying the Pohlig-Hellman algorithm, consider $g = 10$ and $h = 243278$, and assume g is a generator. The factorization of the multiplicative order of g is $746497 - 1 = 2^{10}3^6$. Next, we define

$$g_1 = g^{\frac{2^{10}3^6}{3^6}} \quad h_1 = h^{\frac{2^{10}3^6}{3^6}} \quad g_2 = g^{\frac{2^{10}3^6}{2^{10}}} \quad h_2 = h^{\frac{2^{10}3^6}{2^{10}}}$$

Note that the order of $g_1 = 3^6$ and $g_2 = 2^{10}$, as given by the Pohlig-Hellman algorithm. Now applying the methods of 3.5.10, consider the DLP $g_1^{x_\alpha} = h_1$ where

$$x_\alpha = x_0 + x_1(3) + x_2(3^2) + x_3(3^3) + x_4(3^4) + x_5(3^5)$$

Continuing with the 3.5.10 method, we can calculate $x_0, x_1, x_2, x_3, x_4, x_5$ and use these values to calculate x_α. To find x_0, we consider:

$$\left(g_1^{3^5}\right)^{x_0} = h_1^{3^5}$$

However, since $\left|g^{3^5}\right| = 3$, x_0 must equal 0, 1, or 2. In this case, we see that $\left(10^{\left(\frac{2^{10}3^6}{3^6}\right)^{3^5}}\right)^{x_0} = \left(243278^{\frac{2^{10}3^6}{3^6}}\right)^{3^5}$ gives $x_0 = 0$.

Similarly, we can find x_1 by considering:

$$\left(g_1^{3^5}\right)^{x_1} = h_1^{3^4} g_1^{-x_0 3^4}$$

However, this computation differs slightly because it relies on the identifying $g_1^{-x_0 3^4}$. We can identify the inverse of x_0 by using Theorem 1.2.41 such that $\left(g_1^{x_0 3^4}\right)^{746497-2} \equiv g_1^{-x_0 3^4} \pmod{746497}$.

Thus, proceeding in the same fashion, we can find x_1 by plugging in $0, 1$ and 2 for x_1, and determining which results in $h_1^{3^4} g_1^{-x_0 3^4}$. We find that $x_1 = 2$.

Continuing this pattern set out in Theorem 3.5.10, we find the following equations:

$$\left(g_1^{3^5}\right)^{x_2} = h_1^{3^3} g_1^{-x_1 3^4} g_1^{-x_0 3^3}$$

$$\left(g_1^{3^5}\right)^{x_3} = h_1^{3^2} g_1^{-x_2 3^4} g_1^{-x_1 3^3} g_1^{-x_0 3^2}$$

$$\left(g_1^{3^5}\right)^{x_4} = h_1^{3^1} g_1^{-x_3 3^4} g_1^{-x_2 3^3} g_1^{-x_1 3^2} g_1^{-x_0 3^1}$$

$$\left(g_1^{3^5}\right)^{x_5} = h_1^{3^0} g_1^{-x_4 3^4} g_1^{-x_3 3^3} g_1^{-x_2 3^2} g_1^{-x_1 3} g_1^{-x_0 3^0}$$

These equations can be solved after calculating the inverses in the same way we did while solving for x_1, by relying on Theorem 1.2.41. Using this method, we find that

$$x_0 = 0,\ x_1 = 2,\ x_2 = 0,\ x_3 = 1,\ x_4 = 2,\ x_5 = 2$$

Plugging these values in to calculate x_α, we find

$$x_\alpha = 0(3^0) + 2(3^1) + 0(3^2) + 1(3^3) + 2(3^4) + 2(3^5) = 681$$

We start at the beginning of this process again, and consider the DLP $g_2^{x_\beta} = h_2$ where $x_\beta = y_0(2^0) + y_1(2^1) + y_2(2^2) + \cdots + y_9(2^9)$.

Again, continuing with the 3.5.10 method, we can calculate $y_0, \ldots, y_9$ and use these values to calculate x_β. To find y_0, we consider:

$$\left(g_2^{2^9}\right)^{y_0} = h_2^{2^9}.$$

However, since $\left|g_2^{2^9}\right| = 2$, this limits y_0 to be 0 or 1. In this case, we see that $\left(10^{\left(\frac{2^{10}3^6}{3^6}\right)^{2^9}}\right)^{y_0} = \left(243278^{\frac{2^{10}3^6}{3^6}}\right)^{2^9}$ gives $y_0 = 1$.

We go through the same process as we did for the base-3 expansion, using the following equations:

$$\left(g_2^{2^9}\right)^{y_1} = h_2^{2^8} g_2^{-y_0 2^8}$$

$$\left(g_2^{2^9}\right)^{y_2} = h_2^{2^7} g_2^{-y_1 2^8} g_2^{-y_0 2^7}$$

$$\left(g_2^{2^9}\right)^{y_3} = h_2^{2^6} g_2^{-y_2 2^8} g_2^{-y_1 2^7} g_2^{-y_0 2^6}$$

$$\left(g_2^{2^9}\right)^{y_4} = h_2^{2^5} g_2^{-y_3 2^8} g_2^{-y_2 2^7} g_2^{-y_1 2^6} g_2^{-y_0 2^5}$$

$$\left(g_2^{2^9}\right)^{y_5} = h_2^{2^4} g_2^{-y_4 2^8} g_2^{-y_3 2^7} g_2^{-y_2 2^6} g_2^{-y_1 2^5} g_2^{-y_0 2^4}$$

$$\left(g_2^{2^9}\right)^{y_6} = h_2^{2^3} g_2^{-y_5 2^8} g_2^{-y_4 2^7} g_2^{-y_3 2^6} g_2^{-y_2 2^5} g_2^{-y_1 2^4} g_2^{-y_0 2^3}$$

$$\left(g_2^{2^9}\right)^{y_7} = h_2^{2^2} g_2^{-y_6 2^8} g_2^{-y_5 2^7} g_2^{-y_4 2^6} g_2^{-y_3 2^5} g_2^{-y_2 2^4} g_2^{-y_1 2^3} g_2^{-y_0 2^2}$$

$$\left(g_2^{2^9}\right)^{y_8} = h_2^{2^1} g_2^{-y_7 2^8} g_2^{-y_6 2^7} g_2^{-y_5 2^6} g_2^{-y_4 2^5} g_2^{-y_3 2^4} g_2^{-y_2 2^3} g_2^{-y_1 2^2} g_2^{-y_0 2^1}$$

$$\left(g_2^{2^9}\right)^{y_9} = h_2^{2^0} g_2^{-y_8 2^8} g_2^{-y_7 2^7} g_2^{-y_6 2^6} g_2^{-y_5 2^5} g_2^{-y_4 2^4} g_2^{-y_3 2^3} g_2^{-y_2 2^2} g_2^{-y_1 2^1} g_2^{-y_0 2^0}$$

To use these equations to find $y_1, \cdots, y_9$, we must use 1.2.41 to find the inverses. This method gives that

$$\begin{array}{llllll} y_0 = 1, & y_1 = 1, & y_2 = 0, & y_3 = 1, & y_4 = 0, \\ y_5 = 0, & y_6 = 0, & y_7 = 0, & y_8 = 0, & y_9 = 1 \end{array}$$

Plugging these values in to calculate x_β, we find

$$x_\beta = 1 + 1(2) + 0(2^2) + 1(2^3) + 1(2^4) + 0(2^5) + 0(2^6) + 0(2^7) + 0(2^8) + 1(2^9) = 523.$$

Continuing in the Pohlig-Hellman algorithm, we can now use the Chinese Remainder Theorem to solve the simultaneous system of linear congruences

$$x \equiv 681 \pmod{3^6}, \quad x \equiv 523 \pmod{2^{10}}$$

To use the Chinese Remainder Theorem, we start with the general solution to the first congruence, $x = 681 + 3^6 y$. Substituting into our second congruence, we have $681 + 3^6 y \equiv 523 \pmod{2^{10}}$. Thus $y = (523 - 681)3^{-6} \equiv 306 \pmod{2^10}$. Plugging this value of y back into the general solution to the first congruence, we have $x = 681 + 3^6(306) = 223755$. Thus $10^{223755} = 243278$ in $\mathbb{F}_{746497}$.

4

PRIMES AND FACTORIZATION

As discussed in Chapter 3, the goal of public key encryption is to enable the transmission of sensitive information over insecure channels. The security of public key cryptosystems rests in the difficulty of computing the inverse of some underlying one-way function, without some additional (trapdoor) information. In the case of Diffie-Hellman key exchange and ElGamal public key cryptography, the underlying one-way function is $f(x) = g^x$, and security rests in the difficulty of the discrete logarithm problem. In the case of RSA (the first published public key encryption scheme), the underlying one-way function is $f(x) = x^n$, and security rests in the difficulty of computing n^{th} roots in a finite field.

4.1 RSA ENCRYPTION

There are three main components to the RSA public key cryptosystem: key creation, encryption, and decryption. To illustrate these three components, consider once again our intrepid covert agents Bob and Alice.

- **Key Creation**: In order for Alice to receive secure communications from Bob, she must first set-up her public/private key pair. Her private key consists of two large primes p and q. For her public key, she chooses some encryption exponent e such that $\gcd(e, (p-1)(q-1)) = 1$. She then publishes e and the product N of p and q.
- **Encryption**: To encrypt a message m for Alice, Bob computes the ciphertext

  $$c = m^e \pmod{N}.$$

  Note: we assume $\gcd(m, N) = 1$. If this is not the case, Bob can always pad his message with irrelevant information to make it the case.
- **Decryption**: Given the trapdoor knowledge of p and q, it is easy for Alice to decrypt Bob's message. Alice proceeds with her decryption by calculating $d \equiv e^{-1} \pmod{N}$, and then by raising Bob's cipher text to to the power of d:

  $$c^d = (m^e)^d$$

Note that Alice can calculate e^{-1} because $\gcd(e, \phi(N)) = 1$. Since Alice is very comfortable with modular arithmetic, she notices that $ed \equiv 1 \pmod{\phi(N)}$ which, by the definition of modular arithmetic, is $k\phi(N) = ed - 1$ or $ed = k\phi(N) + 1$, for some $k \in \mathbb{Z}$. This is useful for Alice's decryption:

$$\begin{aligned} c^d &\equiv (m^e)^d \equiv m^{ed} \pmod{N} \\ &\equiv m^{k\phi(N)+1} \pmod{N} \\ &\equiv m^{k\phi(N)} \cdot m \pmod{N} \\ &\equiv \left(m^{\phi(N)}\right)^k \cdot m \pmod{N} \end{aligned}$$

By Theorem 1.2.34 we know that $m^{\phi(N)} \equiv 1 \pmod{N}$. It therefore follows that

$$\begin{aligned} c^d &\equiv 1^k \cdot m \pmod{N} \\ &\equiv m \pmod{N} \end{aligned}$$

Remark 4.1.1. By the following theorem, the decryption process can be made more efficient by instead calculating $d = e^{-1} \pmod{\operatorname{lcm}\big((p-1),(q-1)\big)}$.

Theorem 4.1.2 (Euler's Formula for p and q)**.** *Let p and q be distinct primes. For all a such that $\gcd(a, pq) = 1$,*

$$a^{\operatorname{lcm}(\phi(p),\phi(q))} \equiv 1 \pmod{pq}$$

Proof. Let q and p be distinct odd primes, and let $a \in \mathbb{Z}$, such that $\gcd(a, pq) = 1$. Suppose a is an arbitrary element in $\mathbb{Z}_{pq}$. By the Modern Chinese Remainder Theorem, there exists an isomorphism that maps $a \in \mathbb{Z}_{pq}$ to some $(a_1, a_2) \in \mathbb{Z}_p \times \mathbb{Z}_q$, such that $a_1 \in \mathbb{Z}_p$ and $a_2 \in \mathbb{Z}_q$.

Consider $(a_1, a_2)^{\operatorname{lcm}(\phi(p),\phi(q))} = \left(a_1^{\operatorname{lcm}(\phi(p),\phi(q))}, a_2^{\operatorname{lcm}(\phi(p),\phi(q))}\right)$. By the definition of the least common multiple, we can say that $\operatorname{lcm}\big(\phi(p), \phi(q)\big) = n\phi(p) = m\phi(q)$ for some $n, m \in \mathbb{Z}$, and so by substitution and Theorem 1.2.34, it follows that

$$\left(a_1^{\operatorname{lcm}(\phi(p),\phi(q))}, a_2^{\operatorname{lcm}(\phi(p),\phi(q))}\right) = \left(a_1^{n\phi(p)}, a_2^{m\phi(q)}\right) = (1, 1)$$

Therefore $\big|(a_1, a_2)\big| \operatorname{lcm}\big((\phi(p), \phi(q)\big)$. By the CRT, this implies that $|a|$ divides the $\operatorname{lcm}\big((\phi(p), \phi(q)\big)$, and so there exists a $k \in \mathbb{Z}$, such that $|a|k = \operatorname{lcm}\big(\phi(p), \phi(q)\big)$. Thus $a^{|a|k} \equiv a^{\operatorname{lcm}(\phi(p),\phi(q))} \equiv 1 \pmod{pq}$. □

4.2 PRIMALITY TESTING

In order to create her RSA public/private key pair, Alice has to choose two large prime numbers p and q. It is not enough for her to chose two large, possibly composite, numbers p and q. For one thing, she will need to know how to factor p and q to compute the trapdoor information $\phi(N)$. Secondly, if p or q contain small factors, an adversary Eve may be able to factor N and thus decrypt messages intended for Alice. Thus, Alice is faced with the task of distinguishing large prime numbers from large composite numbers. In this section, we'll examine probabilistic tests and factorization methods for determining whether a given number is prime.

4.2.1 *Probabilistic Tests: Miller-Rabin Witnesses*

By Fermat's Little Theorem (Theorem 1.2.38), we know that if p is a prime number, then

$$a^p \equiv a \pmod{p}$$

for all integers a. Thus, Fermat's Little Theorem provides a method for proving something is **not** prime. For example, $2^m = 2^{15485207} \not\equiv 2 \mod 15485207$, thus m is not prime. However, the converse is not true: if $a^m \equiv a \pmod{m}$ then m may or may not be prime.

Definition 4.2.1 (Witnesses)**.** For a given integer n, an integer a is called a *witness (for the compositeness) of* n if

$$a^n \not\equiv a \pmod{n}.$$

This suggests the following probabilistic test for primality:

Example 4.2.2 (Possible probabilistic test for primality). For a given potentially prime, large number n,

- chose a large set of numbers $a_1, a_2, \ldots, a_k$ and check whether any of these a_i are witnesses for n.
- If you find a witness, then you know n is not prime. If you do not find a witness, then it seems likely that n is probably prime.

How many a_i should we check to be relatively certain n is prime? Can we ever be certain of the primeness of n using this testing method? To answer these questions we need some idea of how many witnesses there are for a given integer n. For all composite numbers, does there exist a witness? Unfortunately, the answer to this last question is no.

Definition 4.2.3 (Carmichael Numbers)**.** Composite numbers having no witnesses are called *Carmichael numbers*, after R.D. Carmichael who published a list of 15 such numbers in 1910.

Example 4.2.4. Show that 561 is a Carmichael number. Notice that $561 = 3 \cdot 11 \cdot 17$. Let $a \in \mathbb{Z}/561\mathbb{Z}$. First, consider $a^{560} \pmod 3$. We find that

$$a^{561} \equiv a^{560}a \equiv (a^2)^{280}a \equiv \left(a^{\phi(3)}\right)^{280} a \equiv 1 \cdot a \equiv a \pmod 3$$

Checking, $a^{560} \pmod{11}$ we have

$$a^{561} \equiv a^{560}a \equiv (a^{10})^{56}a \equiv \left(a^{\phi(11)}\right)^{56} a \equiv 1 \cdot a \equiv a \pmod{11}$$

Finally, $a^{560} \pmod{17}$ we see

$$a^{561} \equiv a^{560}a \equiv (a^{16})^{35}a \equiv \left(a^{\phi(17)}\right)^{35} a \equiv 1 \cdot a \equiv a \pmod{17}$$

Notice that $2 = \phi(3)$, $10 = \phi(11)$, and $16 = \phi(17)$, this is because $3, 11$ and 17 are all prime and $\phi(p) = p - 1$ by Example 1.2.31. By the CRT, it follows that $a^{561} \equiv a \pmod{3 \cdot 11 \cdot 17}$, and thus by the definition of Carmichael number, 561 is a Carmichael number.

Theorem 4.2.5. *Carmichael numbers must be odd*

Proof. Let c be a Carmichael number, and by way of contradiction, suppose c is even. By the definition of an even number, $c = 2k$ for some $k \in \mathbb{Z}$. Now consider $(c-1)^c$. By the definition of a Carmichael number, $(c-1)^c \equiv c - 1 \pmod c$, and by substitution it follows that $(2k-1)^{2k} \equiv 2k - 1 \pmod{2k}$. Simplifying, we see that $(-1)^{2k} \equiv -1 \pmod{2k}$, and therefore $(-1)^{2k} \equiv (-1^2)^k \equiv 1^k \equiv 1 \equiv -1 \pmod{2k}$.

This is only true if $c = 2$, however by the definition of a Carmichael number, c must not be prime. This is a contradiction, and thus every Carmichael number must be odd. □

Theorem 4.2.6. *Carmichael numbers must be the product of distinct primes*

Proof. Let c be a Carmichael number, and factor out all powers of a prime q such that $c = q^n m$. Consider $q^c = q^{q^n m}$, and since c is a Carmichael number, we know that $q^{q^n m} \equiv q \pmod{q^n m}$. Since $\gcd(q, m) = 1$, we can use the CRT to say that $q^{q^n m} \equiv q \pmod{q^n}$. Since $q > 1$, we know that $n < q^n m$, and therefore $q^n | q^{q^n m}$. This implies that $q^{q^n m} \equiv 0 \pmod{q^n}$. By substituting this into $q^{q^n m} \equiv q \pmod{q^n}$, we see that $q \equiv 0 \pmod{q^n}$, and by the definition of modular congruence, $q^n | q$. This is only possible if $n = 0$ or $n = 1$, and so a prime q can appear at most once in the prime factorization of c. Thus a Carmichael number is the product of distinct primes. □

While Carmichael numbers are relatively rare, Alford, Granville, and Pomerance [1] proved that there are infinitely many of them. So Alice needs something stronger than Fermat's Little Theorem to test whether a number is *probably* prime.

Lemma 4.2.7. *Up to modular equivalence, there are only two square roots of* $1 \pmod{p}$*, namely* 1 *and* $p-1$*, where* p *is an odd prime.*

Proof. Let p be an odd prime. Consider the polynomial $f(x) \equiv x^2 - 1 \pmod{p}$, and note that any square roots of 1 modulo p will be a zero of this polynomial. By Lagrange's Other Theorem (Theorem 1.2.25), $f(x)$ has at most two roots. Notice that

$$f(1) \equiv (1^2) - 1 \equiv 0 \pmod{p}$$

and

$$f(p-1) \equiv (p-1)^2 - 1 \equiv p^2 - 2p + 1 - 1 \equiv 1 - 1 \equiv 0 \pmod{p}$$

are both roots of $f(x)$. Since p is an odd prime, we know that these roots are distinct, and thus we have found the only roots of $1 \pmod{p}$. □

Theorem 4.2.8. *Let* p *be an odd prime, and let* a *be any number not divisible by* p*. Consider*

$$p - 1 = 2^k q \qquad \text{with } q \text{ odd.}$$

One of the following two conditions must be true:

i. $a^q \equiv 1 \pmod{p}$

ii. One of $a^q, a^{2q}, a^{4q}, \ldots, a^{2^{k-1}q}$ *is congruent to* $-1 \pmod{p}$

Proof. Let p be an odd prime, and let a be a number not divisible by p. Suppose $p - 1 = 2^k q$ for an odd number q. Consider $(a^{2^{k-1}q})^2 \equiv a^{2^k q} = a^{p-1} \pmod{p}$, and by Theorem 1.2.38 it follows that this is equivalent to 1 modulo p. By Lemma 4.2.7, it follows that either $a^{2^{k-1}q} \equiv 1 \pmod{p}$ or $a^{2^{k-1}q} \equiv -1 \pmod{p}$.

If it is the latter, then condition *ii* is met, and so suppose that $a^{2^{k-1}q} \equiv 1 \pmod{p}$, and we see that $a^{2^{k-1}q} \equiv (a^{2^{k-2}q})^2 \equiv 1 \pmod{p}$. Now, by Lemma 4.2.7, it follows that $a^{2^{k-2}q} \equiv 1 \pmod{p}$ or $a^{2^{k-2}q} \equiv -1 \pmod{p}$.

It is clear that this process must terminate either when , $a^{2^{\ell}q} \equiv -1 \pmod{p}$ or when we reach $a2^1 q \equiv 1 \pmod{p}$. Thus, if none of $a^q, a^{2q}, a^{4q}, \ldots, a^{2^{k-1}q}$ are congruent to $-1 \pmod{p}$, then it follows that $a^{2^0 q} \equiv a^q \equiv 1 \pmod{p}$, in which case condition *i* is true.

Therefore, condition *i* or condition *ii* must be true. □

1 W. R. Alford, A. Granville, and C. Pomerance. There are infinitely many Carmichael numbers. *Ann. of Math. (2)*, 139(3):703-722, 1994

The previous theorem suggests the following Miller-Rabin test for compositeness.

Definition 4.2.9 (Miller-Rabin Witness)**.** Let n be an odd number and write $n-1 = 2^k q$ where q is odd. An integer a, relatively prime to n, is called a *Miller-Rabin witness for (the compositeness of) n* if both the following conditions are true.

i. $a^q \not\equiv 1 \pmod{n}$

ii. $a^{2^i q} \not\equiv -1 \pmod{n}$ for all $i = 0, 1, 2, \ldots, k-1$.

Definition 4.2.10 (Miller-Rabin Test)**.** For a given potentially prime, large number n,

- chose a large set of numbers $a_1, a_2, \ldots, a_k$ and check whether any of these a_i are Miller-Rabin witnesses for n.
- If you find a witness, then you know n is not prime. If you do not find a witness, then it seems likely that n is probably prime.

Is this really an improvement over our initial probabilistic test for primality? We need to address the same questions as before: How many a_i should we check to be relatively certain n is prime? Can we ever be certain of the primeness of n using this testing method? For every composite number, does there exist a Miller-Rabin witness.

The advantage of the Miller-Rabin test over our initial probabilistic primality test is that for every composite number n, there does exist a Miller-Rabin witness. The following two theorems, stated without proof[2], reveal the power of the Miller-Rabin test.

Theorem 4.2.11. *Let n be an odd composite number, then at least 75% of the numbers between 1 and $n-1$ are Miller-Rabin witnesses for n.*

Theorem 4.2.12. *If a generalized version of the Riemann hypothesis is true, then every composite number n has a Miller-Rabin witness a satisfying*

$$a \leq 2(\ln n)^2.$$

4.2.2 *Factorization*

The composite number N at the heart of the RSA cryptosystem must strike a delicate balance. On one hand, it needs to be large enough that it can not be easily factored. On the other had, it can not be too large, or computations modulo N will be too time consuming. Determining which values of N strike this balance of security and efficiency has been a motivating force behind the modern study of factorization.

2 For proofs, see V. Shoup. *A Computational Introduction to Number Theory and Algebra*, and E. Bach, J. Shallit. *Algorithmic Number Theory, Vol. 1*.

The study of factorization dates back to at least ancient Greece, however it wasn't until the advent of computers that people started developing algorithms capable of factoring very large numbers. There is still no (publicly) known algorithm for factoring *all* large numbers. The modern study of factorization involves topics from elliptic curves, algebraic number theory, and quantum computing.

Not all numbers of a given length are equally hard to factor. Many factorization algorithms are only efficient for certain classes of composite numbers. In this section we examine one such factorization algorithm, *Pollards p-1 method*. Although not useful for all numbers, there are certain types of numbers for which it is quite efficient. Pollard's method demonstrates that there are insecure RSA moduli that at first glance appear secure. In addition, the $p-1$ method provides the inspiration for Lenstra's elliptic curve factorization method, discussed in 7.3.

4.2.3 *Pollard's $p-1$ factorization algorithm*

Here is the problem: Given a composite number $N = pq$, find the prime factors p and q. Suppose that we had an integer L, such that

$$p-1|L \qquad \text{and} \qquad q-1 \nmid L.$$

Where did this L come from? For the moment, we'll assume it was a gift from the factorization fairies.

Theorem 4.2.13. *Let a be a randomly chosen integer. For most choices of a,*

$$p = \gcd\left(a^L - 1, N\right)$$

Proof. Suppose $N = pq$, where p and q are primes. Let L be an integer such that $p-1|L$ and $q-1 \nmid L$, and let a be any integer such that $p \nmid a$ and $q \nmid a$. Note that many integers are not divisible by relatively large primes and thus most choices of a satisfy these restrictions. By the definition of divides, $(p-1)m = L$ for some $m \in \mathbb{Z}$. Consider a^L. By Theorem 1.2.38,

$$a^L \equiv a^{(p-1)m} \equiv 1 \pmod{p}.$$

Thus by the definition of modular congruence, $p|(a^L - 1)$, and it follows that p is a common divisor of $a^L - 1$ and N. At this point it is clear that the $\gcd\left(a^L - 1, N\right)$ is either p or N. In the case that $\gcd\left(a^L - 1, N\right) = N$, we know $q|(a^L - 1)$. Recall our assumptions that $q-1 \nmid L$ and $q \nmid a$. By the division algorithm we see $L = (q-1)k + r$ for $0 < r < q-1$. Again, consider a^L. We see that by Theorem 1.2.38

$$a^L \equiv a^{(q-1)k+r} \equiv a^{(q-1)k}a^r \equiv a^r \pmod{q}.$$

Notice that $q|a^L - 1$ (or equivalently $a^L \equiv 1 \pmod{q}$) when $a^r \equiv 1 \pmod{q}$, which would imply that the order of a divides r (by Proposition 1.2.3). Thus $q|a^L - 1$ only

when L is congruent to a multiple of the order of a modulo $q-1$. For all integers a where this is not the case, $\gcd\left(a^L - 1, N\right) = p$. □

This is great! We have a method for factoring N with a little help from the factorization fairies. Unfortunately, the factorization fairies are a fickle lot. Can we find a value L that is divisible by $p-1$, but not $q-1$? The answer is yes when p is a *smooth number*.

Definition 4.2.14 (Smooth Numbers)**.** An integer is *B-smooth* if all its prime factors are less than or equal to B. In general, a number is called *smooth* if its prime factors are all small.

Pollard's $p-1$ algorithm, developed by John Pollard in 1974, makes use of the following observation about smooth numbers.

Remark 4.2.15. If $p-1$ is smooth, it will divide $n!$ for some relatively small value of n.

Definition 4.2.16 (Pollard's $p-1$ factorization algorithm)**.** For each value $n = 2, 3, 4, \ldots$ choose an integer a and compute $\gcd(a^{n!} - 1, N)$. In practice, we might simply take $a = 2$.

- If the gcd is equal to 1, then go on to the next value of n.
- If the gcd ever equals N, then we have been quite unlucky.

  If $\gcd(a^{n!} - 1, N) = N$, it follows that $N|(a^{n!} - 1)$. Thus, $p|(a^{n!} - 1)$, and we can write $pm = a^{n!} - 1$ for some $m \in \mathbb{Z}$ by the definition of divides. We see that $\gcd(pm, N) = N$ when $N|pm$, which would imply $Nk = pm$. Since $N = pq$, we have $pqk = pm$ by substitution. Thus, $m = qk$, and it follows that $q|m$. In this case, our choice of a resulted in a value of $a^{n!} - 1$ divisible by both p and q, and we are unable to factor N.

  We simply chose another value for a and try again.
- If the gcd is strictly between 1 and N, then we have a nontrivial factor of N and we are done.

Remark 4.2.17. Even for $a = 2$ and relatively small values of n, it is not feasible to calculate $a^{n!} - 1$. For example, the number $2^{100!}$ has more than 10^{157} digits, that's more than the number of elementary particles in the known universe! Fortunately, since we are only interested in the gcd of $a^{n!} - 1$ and N, it suffices to compute $a^{n!} - 1 \pmod N$ and then take the $d = \gcd(a^{n!} - 1 \pmod N, N)$, since if $d|a^{n!} - 1$ and $d|N$ then $d|a^{n!} - 1 \pmod N$. Thus we never need to work with numbers greater than N.

Remark 4.2.18. In implementing Pollard's $p-1$ algorithm, we do not need to compute the values of $n!$. Assuming we have already calculated $a^{(n-1)!}$, we can calculate $a^{n!}$ as $(a^{(n-1)!})^n$.

Example 4.2.19. We use Pollard's $p-1$ method to factor $N = 220459$. Let $a = 2$.

$$\begin{array}{ll} n=2 & \gcd(2^{2!}-1, N) = 1 \\ n=3 & \gcd(2^{3!}-1, N) = 1 \\ n=4 & \gcd(2^{4!}-1, N) = 1 \\ n=5 & \gcd(2^{5!}-1, N) = 1 \\ n=6 & \gcd(2^{6!}-1, N) = 1 \\ n=7 & \gcd(2^{7!}-1, N) = 1 \\ n=8 & \gcd(2^{8!}-1, N) = 449 \end{array}$$

Thus $220459 = (449)(491)$. Note that when $n = 4$, it becomes difficult to calculate $2^{n!}$. At this point, we employ Remarks 4.2.17 and 4.2.18 to calculate $2^{4!} = (2^{3!})^4 \pmod{220459}$. We use this technique for $n = 4, ..., 8$.

Remark 4.2.20. It is easy for Alice to avoid the dangers of Pollard's $p-1$ method when creating her RSA keys. She simply needs to check that her chosen secret primes p and q have the property that neither $p-1$ nor $q-1$ are smooth. From a cryptographic perspective, the importance of Pollard's method lies in the realization that even if we build a cryptosystem based on a seemingly hard problem such as integer factorization, we must be wary of special cases of the problem, that for subtle reasons, are easier to solve than the general case.

For example, consider the Discrete Log Problem $g^x = h$. Suppose g has smooth order N. The Pohlig-Hellman Algorithm tells us that the number of steps in solving a DLP depends on the prime factorization of $|g| = N$. Since N is smooth, it has small prime factors, which allows us to solve this DLP in a reasonable number of steps.

5

COMPLEXITY AND PRIMALITY (BY LEO GOLDSTEIN)

5.1 COMPLEXITY CLASSES

Recall that in section 3.3.1, we discussed the notion of "difficulty" for mathematical problems, and specifically how we can use Order Notation to quantify how hard a particular problem is to solve. For example, a problem that takes $\mathcal{O}(m)$ steps to solve on an input of size m would generally be considered easy, while a problem that takes $\mathcal{O}(2^m)$ steps is hard. But even these are imprecise statements. What does it mean for a problem to take a certain number of steps to solve? What counts as a step? To answer this question and more, we turn to the Turing Machine

Definition 5.1.1 (Turing Machine)**.** A Turing Machine is a theoretical representation of a machine consisting of a tape that can be read and written to, along with a set of states. The Turing Machine is mathematically a 7-tuple $T = \langle Q, q_0, F, \Gamma, b, \Sigma, \delta \rangle$ where

- Q is a set of states
- q_0 is the state the Turing Machine starts in
- $F \subseteq Q$ is the set of accepting states. If the machine stops here, it accepts the input it was given.
- Γ is a set of symbols which are allowed to appear on the machine tape.
- b is the blank symbol, representing no information
- $\Sigma \subseteq \Gamma$ is the set of symbols which are allowed in the machine input. Typically this is just {0,1}
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$ is the partial function which dictates how the machine moves. It reads the current state and the current symbol on the tape, and tells the machine which state to move to next, what to write to the tape in the current position, and whether to move left, right, or not at all on the tape.

The exact mechanics of a Turing Machine are beyond the purview of this text. For our purposes, it suffices to know the following theorem.

Theorem 5.1.2. *Every problem that can be solved by any currently understood model of computer can be solved by a Turing Machine.*

Indeed, it is believed (although unproven) that the result is even stronger than that.

Theorem 5.1.3 (Church-Turing Thesis)**.** *Every problem that can be solved using an algorithm can be solved by a Turing Machine*

Thankfully, the proven theorem is all we need. Since every real-world computer problem can be modeled by a Turing Machine, we can base our assessments on the complexity of a problem on the relatively mathematically simple model of a Turing Machine.

Definition 5.1.4 (Polynomial Time)**.** We define P to be the set of problems for which there exists a Turing Machine that, given an input of size m, solves the problem in $\mathcal{O}(poly(m))$ steps, where $poly(m)$ represents any polynomial function of m.

Remark 5.1.5. Notice that in the above definition we refer to the size of a given input. Generally speaking, for an input n, the size of n is $m = \log_2(n)$, being the number of binary bits required to represent n.

Complexity classes aren't just limited to time factors though. For example, we can define classes based on how much space the solving of a given problem will take up

Definition 5.1.6 (Log Space)**.** We define L to be the set of problems for which there exists a Turing Machine that, given an input of size m, solves the problem using $\mathcal{O}(log_2(m))$ spaces on the tape.

Now that we've created these two seemingly separate complexity classes, it seems only natural that we try and draw some sort of link between them. In this case, that link is both simple and useful

Theorem 5.1.7. $L \subseteq P$

Proof. Suppose we have some problem which is solved by a Turing Machine in $\log_2(m)$ spaces. Dealing with a binary alphabet, there are only $2^{\mathcal{O}(\log_2(m))} = m^{\mathcal{O}(1)}$ possible configurations of the tape. Since the Turing Machine can only cycle through up to $m^{\mathcal{O}(1)}$ configurations, it can only take at most $m^{\mathcal{O}(1)}$ steps, which places it in P □

So far, we've only been dealing with what are known as *deterministic* Turing Machines. These are Turing Machines which behave like, well, machines. Given a certain input, they follow a pre-programmed set of instructions in order to eventually achieve a goal.

But there are also what are known as *non-deterministic* Turing Machines. These machines have some element of choice in how they behave. Given an input, their programming says "You can choose option A, B, or C" and the machine gets to decide. For the purposes of complexity classes, we assume that the machine happens to be the world's luckiest guesser, always choosing the option that leads to the solution in the fastest possible way.

As another way of thinking about it, you can imagine a deterministic machine as moving through a series of branching paths, making its choices based entirely on its programming and creating a linear path through the branches. In contrast, the non-deterministic machine takes every choice at every branch, creating a tree of decisions rather than a path. If any limb of that tree ends in a positive result, the non-deterministic machine has solved the problem. Using this idea, we can define yet more complexity classes.

Definition 5.1.8 (Non-deterministic Polynomial Time)**.** We define NP to be the set of problems for which there exists a non-deterministic Turing Machine that, given an input of size m, solves the problem in $\mathcal{O}(poly(x))$ steps.

One of the ways we check whether or not a problem belongs to NP is by searching for a *witness*. A witness, in this case, is a demonstrable path that solves the problem in the appropriate amount of time.

Example 5.1.9. Consider the problem of factoring composite numbers. We construct a Turing machine that divides our input, a composite number n, by some number k less than n. It is easy to see that division takes place in polynomial time on the size of n, so it remains to find a witness. In this case, the witness is the factor of n. Since we know that there exists j such that $j|n$, when our machine non-deterministically branches out, the branch where it divides n by j will successfully factor n in polynomial time. Therefore the problem of factoring composite numbers is in NP

You might be asking at this point, how do we place problems in complexity classes? The most obvious answer is that if we can construct a Turing Machine that solves the problem in the appropriate time/space, that places it pretty clearly. But what if we don't actually have a good way to solve it? This is where the idea of reduction comes in.

Definition 5.1.10. Given two problems A and B we say that A is polynomially reducible to B, denoted $A \leq_p B$ if there exists a function $f(x)$ such that x is a valid solution to A if and only if $f(x)$ is a valid solution to B and $f(x)$ runs in polynomial time.

Generally speaking, $A \leq_p B$ captures the notion that B is at least as hard to solve as A.

Theorem 5.1.11. *If $A \leq_p B$ and $B \in P$, then $A \in P$*

Proof. Suppose that B is in P, i.e. that there exists a deterministic Turing Machine that solves it in polynomial time. Let x be an input on A. Since $A \leq_p B$, we take $f(x)$ and run that as an input on B. If B accepts $f(x)$, by Definition 5.1.10 A must accept x. Similarly, if B rejects $f(x)$ A must reject x, and so we have a process for solving A. Since B runs in $\mathcal{O}(poly(n))$ time, and $f(x)$ runs in $\mathcal{O}(poly(n))$, the two combined still run in polynomial time and therefore $A \in P$ □

Using this idea of reducibility, we can define exactly what it means for a problem to be hard.

Definition 5.1.12. For a given complexity class C, a problem Z is considered to be *C-hard* if for every problem Y in C, $Y \leq_p Z$. Informally, Z is at least as hard as every problem in C.

Definition 5.1.13. If Z is C-hard and $Z \in C$, we say that Z is *C-complete*

Hardness and completeness are most often used in terms of the complexity class NP, and indeed we will discuss some cryptosystems based on the difficulty of NP-complete problems in Chapters 10 and 11. The benefit of a system based on an NP-complete problem is that any polynomial solution would necessarily also solve every other problem in NP in polynomial time, and we're currently pretty sure that's not possible.

In fact, that leads to a very important question in the world of complexity theory. While it may seem like P and NP are incredibly different, it is still unknown whether or not they are actually distinct. No proof has yet been given for either direction.

5.2 AKS PRIMALITY TEST

Leaving aside the world of complexity theory for the moment, we return to primality. Our last visit to the prime numbers in Chapter 4 left us with a burning desire for a true identification algorithm, one that lets us distinguish between primes and non-primes without fail, but it also left us with the poultice of the probabilistic Miller-Rabin Test to ease our pain somewhat.

For many years mathematicians have struggled with the problem of an efficient solution to primality. The obvious choice, testing every number less than the square root of

our proposed prime to see if it is a factor takes exponential time with regards to the size of n. When working with sufficiently large numbers, it is a vastly inefficient process.

In 2002 however, Agrawal, Kayal, and Saxena came to the rescue of the greater math community, proclaiming to the rooftops "PRIMES is in P!" [11]

Their test, which would come to be known as the AKS Primality Test, is based on the following theorem, a corollary of Fermat's Little Theorem

Theorem 5.2.1. *Let $a \in \mathbb{Z}$, $n \in \mathbb{N}$ and let the $\gcd(a, n) = 1$. n is prime if and only if*

$$(x + a)^n \equiv x^n + a \pmod{n}$$

In order to prove this theorem, some new notation is necessary.

Definition 5.2.2 (Binomial Notation)**.** Throughout this text we will use $\binom{n}{k}$ to denote the number of possible combinations of size k out of a total of n elements.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Theorem 5.2.3 (Binomial Theorem)**.** *In the expanded form for any given polynomial of the form*

$$(x + a)^n = c_n x^n + c_{n-1} x^{n-1} + ... + c_1 x + c_0$$

The coefficient of the i-th term $c_i = \binom{n}{i} * a^{n-i}$

With that done, we can now prove Theorem 5.2.1

Proof. Suppose n is prime. Note that $\binom{n}{i} \equiv 0 \pmod{n}$ for all $i \neq 0, n$. Therefore, by the Binomial Theorem, $(x + a)^n \equiv x^n + a^n \pmod{n}$. Since n is prime, by Fermat's Little Theorem $a^n \equiv a \pmod{n}$ and so $(x + a)^n \equiv x^n + a \pmod{n}$.

Alternatively, suppose n is composite, and let q be a prime such that $q|n$ and k be the maximal power of q such that $q^k|n$. Note that by the Binomial theorem, the coefficient of the q-th term is $\binom{n}{q} * a^{n-q}$. Note further that q^k cannot divide $\binom{n}{q}$, and by our assumption that a and n are relatively prime, q^k is relatively prime to a^{n-q}. These two facts combine to tell us that q^k can't divide the q-th coefficient, which means n cannot divide the q-th coefficient. Therefore, the coefficient of the $q - th$ term cannot be equivalent to zero modulo n. Therefore the polynomials cannot be congruent.

□

This Lemma suggests an immediate test for primality, but the intuitive option of simply choosing an a and evaluating $(x + a)^n$ involves dealing with degree n polynomials,

and therefore up to n multiplications, placing this firmly out of polynomial time on the size of n.

The genius of the AKS primality test is that it reduces this problem modulo a specially chosen polynomial in order to vastly speed up the process.

Remark 5.2.4. For the remainder of this section, we will use $\log(n)$ to mean $\log_2(n)$. In addition, we will use $\pmod{f(x), m}$ to denote equivalance in $\mathbb{Z}_m[x]/\langle f(x)\rangle$

Definition 5.2.5 (AKS Primality Test)**.** For a given integer n

1. If $n = a^b$ for some $a, b \in \mathbb{N}$ and $b > 1$, output COMPOSITE
2. Find the smallest r such that $|n|^* \pmod{r}$ is greater than $\log^2(n)$
3. If $\exists c$ such that $c \leq r$ and $\gcd(n, c) \neq 1, n$, output COMPOSITE
4. If $n \leq r$ output PRIME
5. For all $1 \leq a \leq \lfloor \sqrt{\phi(r)} \log(n) \rfloor$, check if $(x + a)^n \equiv x^n + a \pmod{x^r - 1, n}$.If these polynomials are ever **not** equivalent, output COMPOSITE
6. Otherwise, output PRIME

Theorem 5.2.6 (Correctness of the AKS Primality Test)**.** *The algorithm defined above outputs PRIME if and only if n is prime.*

The proof of this theorem is complex, and we will split it into parts. To begin with, we want to ensure that it will never misidentify a prime as composite.

Lemma 5.2.7. *If n is prime, the AKS algorithm outputs PRIME*

Proof. Suppose n is a prime number. We need to demonstrate that the algorithm will never output COMPOSITE on n.

- Since n is prime, it can't be a power of some number a, so step 1 can never output COMPOSITE
- Since n is prime, there are no divisors of n not equal to 1 or n, meaning step 3 can never output COMPOSITE
- Since n is prime, by Theorem 5.2.1 $(x + a)^n \equiv x^n + a \pmod{n}$ for all a. Therefore by properties of factor ring, it follows that $(x + a)^n \equiv x^n + a \pmod{x^r - 1, n}$, and so step 5 can never output COMPOSITE

□

From here, it remains to show that if the algorithm outputs PRIME, n is prime. This can occur only at two points in the algorithm, in step 4 or in step 6.

Lemma 5.2.8. *If the algorithm defined above outputs PRIME in step 4, n is prime*

Proof. If we have reached step 4, that means in step 3 we checked every number c less than r for common divisors of n. If $n \leq r$, this means we have checked every number less than n for common divisors. Since there were none, n must be prime. □

Proving the correctness of step 6 is significantly more complex. The efficiency of the test relies on this choice of r such that we can reduce our primality test modulo $x^r - 1$. For that purpose, we begin with the following Lemma, presented without proof

Lemma 5.2.9. *If $k \geq 7$, then the least common multiple of the first k numbers is greater than or equal to 2^k*

From here, we can show that the r defined in step 2 exists.

Lemma 5.2.10. *There exists a natural number $r \leq \max(3, \log^5(n))$ such that the multiplicative order of n modulo r is greater than $\log^2(n)$*

Proof. If $n = 2$, then $r = 3$ satisfies all conditions. Therefore, we shall suppose that $n > 2$, which necessarily makes $\lceil \log^5(n) \rceil > 10$. For brevity's sake, we will let $B = \lceil \log^5(n) \rceil$.

Let

$$\pi = n^{\lfloor \log(B) \rfloor} * \prod_{i=1}^{\lfloor \log^2(n) \rfloor} (n^i - 1)$$

and let r be the smallest positive integer such that r does not divide π

Note that for all n greater than or equal to 2,

$$n^{\lfloor \log(B) \rfloor} * \prod_{i=1}^{\lfloor \log^2(n) \rfloor} (n^i - 1) < n^{\lfloor \log(B) \rfloor + \frac{1}{2}\log^2(n)(\log^2(n)-1)} \leq n^{\log^4(n)} \leq 2^{\log^5(n)} \leq 2^B$$

By way of contradiction, suppose $r > B$. Since we chose r to be the smallest number that does not divide the product π, it follows that for all $k \leq B$, k must divide π. Since every number less than or equal to B divides it, it follows that the lcm of the first B integers divides π. However, by Lemma 5.2.9, since $B > 10$ the lcm of the first B integers is greater than or equal to 2^B, but by the above inequality π is strictly less than 2^B, and so we have a contradiction. Therefore it follows that $r \leq B$

From here, it remains that show that the multiplicative order of n modulo r is greater than $\log^2(n)$. First, it must be demonstrated that n and r are relatively prime, in order for n to have any order at all modulo r.

By way of contradiction, suppose that every prime divisor of r divides $\gcd(r,n)$. This implies that $r = q_1^{k_1}q_2^{k_2}...q_m^{k_m}$ and $n = q_1^{j_1}q_2^{j_2}...q_m^{j_m} * l$ for some integer l. Since r does not divide π it cannot divide n, and so it follows that $k_i > j_i$ for at least one i. This further implies that there exists a power v such that $r|n^v$, $r \nmid n^{v-1}$. For our purposes, let q_h be the prime such that $k_h - j_h > k_i - j_i$ for all i, i.e. the prime which has the greatest power in r as compared to its power in n. Note that this means that $k_h > j_h * (v-1)$ and $k_h < j_h * v$

By way of further contradiction, suppose that $v > \lfloor \log(B) \rfloor$. This implies that since $k_h > j_h * (v-1)$, and $v > \lfloor \log(B) \rfloor$, $v - 1 \geq \lfloor \log(B) \rfloor$, and thus $k_h > j_h(\lfloor \log(B) \rfloor)$, by substitution. Since j_h is a positive integer, $k_h > \lfloor \log(B) \rfloor$. However, this implies that since $r = q_h^{k_h} * o$, where o represents the product of the other prime factors of r, $r \geq q_h^{k_h}$ for some prime q_h. Even if q_h is the smallest prime 2, $q_h^{k_h} > q_h^{\log(B)} \geq B$, and thus $r \geq q_h^{k_h} > B$. However, we have already proved that $r \leq B$, and so we have reached our first contradiction.

This means that the smallest power v where $r|n^v$ must be less than or equal to $\lfloor \log(B) \rfloor$. However, if this is the case, then $r|n^{\lfloor \log(B) \rfloor}$, which is yet another contradiction since we chose r so that it would not divide π. Thus we must conclude that not every prime divisor of r is a prime divisor of n.

From here, consider $\frac{r}{\gcd(r,n)}$. Since there exists at least one q such that $q|r$ and $q \nmid n$, clearly $q \nmid \gcd(r,n)$ and so $q|\frac{r}{\gcd(r,n)}$. Since $r = q * x$ for some x does not divide π, it follows that $\frac{r}{\gcd(r,n)} = q * y$ for some y also cannot divide π. However, we chose r to be the smallest number which does not divide π and so $r \leq \frac{r}{\gcd(r,n)}$. The only way this inequality is possible is if $\frac{r}{\gcd(r,n)} = r$, or $\gcd(r,n) = 1$. In other words, r and n must be relatively prime.

Now that we know r and n are relatively prime, we can start to consider the order of n modulo r. Thankfully, this section of the proof is quite simple. Since $r \nmid \pi$, $r \nmid n^i - 1$ for all $1 \leq i \leq \lfloor \log^2(n) \rfloor$. Thus, by definition of modular congruence $n^i \not\equiv 1 \pmod{r}$ for all $1 \leq i \leq \lfloor \log^2(n) \rfloor$. Therefore, the order of n modulo r must be greater than $\log^2(n)$ □

So at this point we've shown that there is always an r which fulfills the criteria set forward in step 3. With that done, we can finally begin to approach the final step of the process. From this point forward, we will be operating in the case that the algorithm has output PRIME in step 6 on our input of n. To begin with, we want to start talking about some prime factor of n.

Remark 5.2.11. Because $n \not\equiv 1 \pmod{r}$, which follows from the fact that the order of n modulo r is greater than 1, there exists a prime p such that $p|n$ and $p \not\equiv 1 \pmod{r}$ as well. Moreover,since n is relatively prime to r, p is also relatively prime to r. If we suppose that $p = n$, since the algorithm did not output PRIME in step 4, $p > r$. If we suppose that $p \neq n$, since the algorithm did not output COMPOSITE in step 3, p must

still be greater than r. In other words, we now have a number p with the properties that p is prime, $p|n$, the order of p modulo r is greater than 1, and $p > r$.

For the remainder of this proof, we will operate with n, r, and p fixed. In addition, let $k = \lfloor \sqrt{\phi(r)} \log(n) \rfloor$. With these numbers fixed, we can now introduce a new definition.

Definition 5.2.12 (Introspective Numbers)**.** For an arbitrary natural number m and polynomial $f(x)$, we say that m is *introspective* over $p, x^r - 1$ for $f(x)$ if

$$f(x)^m \equiv f(x^m) \pmod{x^r - 1, p}$$

Lemma 5.2.13. *$\frac{n}{p}$ and p are introspective for all polynomials $\{x + a | 0 \leq a \leq k\}$*

Proof. Firstly, we have that for all $0 \leq a \leq k$

$$(x + a)^p \equiv x^p + a \pmod{x^r - 1, p}$$

by Theorem 5.2.1, and so p is introspective for all polynomials of the form $\{x + a | 0 \leq a \leq k\}$. Given that the algorithm has output PRIME in step 6,

$$(x + a)^n \equiv x^n + a \pmod{x^r - 1, n}$$

by how we defined our algorithm. By properties of modular congruence therefore

$$(x + a)^n \equiv x^n + a \pmod{x^r - 1, p}$$

Therefore,

$$(x + a)^{\frac{n}{p}} \equiv ((x + a)^n)^{\frac{1}{p}} \equiv (x^n + a)^{\frac{1}{p}} \pmod{x^r - 1, p}$$

Let $y = x^n$ and $z = y^{\frac{1}{p}}$, and note that

$$(z + a)^p \equiv z^p + a \equiv (y^{\frac{1}{p}})^p + a \equiv y + a \equiv x^n + a \pmod{x^r - 1, p}$$

Therefore, the $p - th$ root of $x^n + a$ modulo $x^r - 1, p$ is $z + a \equiv x^{\frac{n}{p}} + a$, and so

$$(x + a)^{\frac{n}{p}} \equiv x^{\frac{n}{p}} + a \pmod{x^r - 1, p}$$

And therefore $\frac{n}{p}$ is also introspective for all polynomials of the form $\{x + a | 0 \leq a \leq k\}$

□

Using this fact and the following two lemmas, we can construct some very useful sets.

Lemma 5.2.14. *If m and v are introspective for $f(x)$, $m * v$ is also introspective for $f(x)$. In other words, introspection is closed over multiplication.*

Proof. Suppose m and v are introspective for $f(x)$, and consider $f(x)^{m*v} \equiv (f(x)^m)^v$. Since m is introspective for $f(x)$, $(f(x)^m)^v \equiv f(x^m)^v$. If we let $y = x^m$, since v is introspective for $f(x)$, it will also be introspective for $f(y)$ and so $f(x^m)^v \equiv f(y)^v \equiv f(y^v) \equiv f(x^{m*v})$. And thus we have shown that

$$f(x)^{m*v} \equiv f(x^{m*v}) \pmod{x^r - 1, p}$$

And so $m * v$ is introspective for $f(x)$ □

Lemma 5.2.15. *If m is introspective for $f(x)$ and $g(x)$, m is introspective for $f(x) * g(x)$. In other words, the set of polynomials a number is introspective for is closed under multiplication*

Proof. Let $f(x)$ and $g(x)$ be polynomials, and suppose m is introspective for both $f(x)$ and $g(x)$. Consider $(f(x) * g(x))^m \equiv f(x)^m * g(x)^m$

Since m is introspective for both polynomials, $f(x)^m * g(x)^m \equiv f(x^m) * g(x^m)$, and thus $(f(x) * g(x))^m \equiv f(x^m) * g(x^m) \pmod{x^r - 1, p}$. Therefore m is introspective for $f(x) * g(x)$

□

With these lemmas, we can construct two sets which will be vital for our proof.

Let $I = \{\left(\frac{n}{p}\right)^i * p^j | i, j \in \mathbb{N}\}$, or the set of powers of $\frac{n}{p}$ multiplied by powers of p. Let $J = \{(x)^{e_0} * (x+1)^{e_1} * ... * (x+k)^{e_k} | e_i \in \mathbb{N}\}$, or the set of products of polynomials of the form $(x + a)$ for $0 \leq a \leq k$. By the above three lemmas, it is clear that if $m \in I$, $f(x) \in J$, then m is introspective for $f(x)$.

Using these two sets, we define two new groups.

Firstly, let $G = \{m \pmod{r} | m \in I\}$. In other words, G is the set of equivalence classes of elements of I modulo r. Since both n, p are relatively prime to r, every element of I will also be relatively prime to r, and so G together with multiplication modulo r forms a subgroup of $\mathbb{Z}/r\mathbb{Z}^*$. We will denote $|G| = t$. Note that since n is a generator for G, and as previously established the order of n modulo r is greater than $\log^2(n)$, t must be greater than $\log^2(n)$ i.e. there are more than $\log^2(n)$ distinct elements of G.

In order to define the second group, we must first introduce some facts about cyclotomic polynomials.

Definition 5.2.16. We define α to be an *n-th root of unity* if $\alpha^n = 1$. If n is the smallest power such that $\alpha^n = 1$, we say that α is a *primitive* n-th root of unity. There are $\phi(n)$ primitive n-th roots of unity.

Definition 5.2.17. The *n-th cyclotomic polynomial* denoted $\Phi_n(x)$ is the polynomial whose roots are the primitive n-th roots of unity.

$$\Phi_n(x) = (x - \alpha_1)(x - \alpha_2)...(x - \alpha_{\phi(n)})$$

where α_i are primitive n-th roots of unity.

Theorem 5.2.18. *Over a finite field of prime order F_q, the n-th cyclotomic polynomial $\Phi_n(x)$ factors into irreducible polynomials of degree equal to the multiplicative order of q modulo n.*

Proof. By properties of field extensions, the irreducible factors of Φ_n over F_q will have degree equal to the degree of the primitive n-th roots of unity over F_q. In other words, α exists in some minimal field extension F_{q^k}. The degree of α over F_q is equal to k, and so Φ_n will factor into irreducible polynomials of degree k. Therefore, it is sufficient to show that the smallest k for which $\alpha \in F_{q^k}$ is the order of q modulo n.

Let F_{q^k} be the smallest field which contains α. By properties of finite fields, $\alpha^{q^k} = \alpha$, which means that $q^k \equiv 1 \pmod{|\alpha|}$ by Theorem 1.2.40 . Since α is a primitive n-th root of unity, $|\alpha| = n$ by definition. Therefore, $q^k \equiv 1 \pmod{n}$. Moreover, since F_{q^k} is the smallest field containing α, it follows that k is the smallest k such that $q^k \equiv 1 \pmod{n}$, and so the order of q modulo n is k.

Thus, the degree of α over F_q, and therefore the degree of the irreducible factors of $\Phi_n(x)$ is equal to the order of q modulo n □

With these, we can define our second group, H. To begin with, let $\Phi_r(x)$ be the r-th cyclotomic polynomial. By the above theorem, its irreducible factors over F_p have degree equal to the order of p modulo r. As previously established, this order is greater than 1, and so the irreducible factors must have degree greater than 1. Let $h(x)$ be one such irreducible factor, with degree > 1.

Since $h(x)$ is an irreducible polynomial in $F_p[x]$, $\mathbb{F} = F_p[x]/\langle h(x)\rangle$ is also a field. From here, we define $H = \{f(x) \pmod{h(x), p} | f(x) \in J\}$. In other words, H is the set of polynomials from J, (with the form $(x)^{e_o} * (x+1)^{e_1} * ... * (x+k)^{e_k}$ where e_i are some natural numbers) taken in the field $\mathbb{F}$ and therefore considered mod $h(x)$.

Lemma 5.2.19. *Let $f(x)$, $g(x)$ be arbitrary polynomials in J with degree less than t. If $f(x) \neq g(x)$, $f(x) \not\equiv g(x) \pmod{h(x), p}$*

Proof. By way of contradiction, suppose $f(x) \equiv g(x) \pmod{h(x), p}$. For any $m \in G$, $f(x)^m \equiv g(x)^m$, and since every element of I (and thus every element of G) is introspective over polynomials from J, $f(x^m) \equiv g(x^m) \pmod{h(x), p}$. If we let $Q(x) = f(x) - g(x)$, this implies that $Q(x^m) = f(x^m) - g(x^m) \equiv 0 \pmod{h(x), p}$, or in other words, that x^m is a root of the polynomial $Q(x)$ for every $m \in I$. Note that since we are operating modulo $h(x)$, and since $h(x)$ is a factor of the r-th cyclotomic polynomial, the value x is a primitive r-th root of unity. Since every m is relatively prime to r, every x^m will also be a primitive root of unity. Importantly, since $m < r$, each x^m will be distinct. Therefore, since every x^m is a root of $Q(x)$, and each is distinct, $Q(x)$ has at least t distinct roots (one for each $m \in G$), but this is a contradiction, since we chose $f(x)$ and

$g(x)$ to have degree less than t, which means $Q(x)$ must also have degree less than t. Therefore, $f(x) \not\equiv g(x) \pmod{h(x), p}$

□

Remark 5.2.20. The above lemma means that each of the linear polynomials $(x), (x+1)...(x+k)$ which generate P are equivalent if and only if their constant term is equivalent modulo r. However, $k = \lfloor \sqrt{\phi(r)} \log(n) \rfloor < \sqrt{r} \log(n)$ since $\phi(r) < r$ for all r. Since we have that order of n modulo r is greater than $\log^2(n)$, $\log^2(n) < \phi(r)$ since every element modulo r has order that divides $\phi(r)$. Since $\log^2(n) < \phi(r)$, certainly $\log(n) < \sqrt{\phi(r)} < \sqrt{r}$, and so by substitution $k < \sqrt{r} \log(n) < r$. Since $k < r$, every element less than or equal to k must be distinct modulo r, and therefore every polynomial $(x), (x+1), ..., (x+k)$ is distinct in H. Using this, we can create a lower bound on the order of H.

Lemma 5.2.21. $|H| \geq \binom{t+k}{t-1}$

Proof. Remark 5.2.20 tells us there are $k+1$ distinct polynomials of degree 1 in H. Recall that every member of H is also an element of J and so has the form

$$(x)^{e_0} * (x+1)^{e_1} * ... * (x+k)^{e_k} \pmod{h(x), p}.$$

These two facts, along with Lemma 5.2.19 tell us that any element we can construct using this form with degree less than t will be distinct in H. Thus, to provide a lower bound on the order of H, we need simply count the number of different constructions we can create. It is easiest to consider this as the sum of the different constructions with degree $t-1, t-2, ..., 2, 1$.

Before we proceed, we must reframe the problem. A product $(x)^{e_0} * (x+1)^{e_1} * ... * (x+k)^{e_k}$ will be distinct from another product so long as the exponent for at least one of the $k+1$ polynomials is different between them. When considering the problem of how many distinct polynomials we can construct with degree $t-1$, we can therefore reframe the issue as the distinct combinations of $e_0 + e_1 + ... + e_k = t-1$, which is the same as the number of distinct $(k+1)$-tuples that sum to $t-1$.

Using the combinatorial "stars and bars" strategy, we envision a line of $t-1$ stars, and consider the number of possible ways to disperse k + signs among them. Counting the stars between plus signs gives us the value of that place in the $k+1$-tuple. For example,

$$+***+*********+**+*****$$

is a unique permutation of these symbols that gives us the tuple $< 0, 2, 0, 8, 2, 5 >$, which sums to 17, the number of stars. By reframing it this way, in terms of a permutation of symbols, we find that the total number of distinct $(k+1)$-tuples that sums to $t-1$ is equal to $\binom{t+k-1}{t-1}$. Similarly, the number of $(k+1)$-tuples that sums to $t-2$ is equal to $\binom{t+k-2}{t-2}$, and so on.

Using this method, we can represent the total number of distinct polynomials of degree less than t, denoted $< (t)$, as the sum of the total numbers of distinct polynomials with degree u for each $u < t$, denoted $\#(u)$. In this way, we have

$$< (t) = \#(t-1) + \#(t-2) + ...\#(2) + \#(1) + \#(0)$$

Which is equal to

$$\binom{t+k-1}{t-1} + \binom{t+k-2}{t-2} + ... + \binom{k+2}{2} + \binom{k+1}{1} + 1$$

Note that there is a single polynomial of degree zero, that being the polynomial $f(x) = 1$ which serves as the identity on H.

This is a complex seeming sum, but fortunately we can use Pascal's Rule to simplify it

Theorem 5.2.22 (Pascal's Rule)**.** $\binom{n}{k} + \binom{n}{k-1} = \binom{n+1}{k}$

Using this, we can collapse the sum starting from the right side. $\binom{k+1}{1} = k+1$, so by adding the 1 combination representing the only polynomial of degree 0, we get $k+2 = \binom{k+2}{1}$. From here, we apply Pascal's Rule to add $\binom{k+2}{2} + \binom{k+2}{1} = \binom{k+3}{2}$, which we then add $\binom{k+3}{3} + \binom{k+3}{2} = \binom{k+4}{3}$, and so on, repeating the process until finally we terminate with

$$\binom{t+k-1}{t-1} + \binom{t+k-1}{t-2} = \binom{t+k}{t-1}$$

Therefore, the number of distinct polynomials in J with degree less than t is equal to $\binom{t+k}{t-1}$, and thus by Lemma 5.2.19, there are at least $\binom{t+k}{t-1}$ distinct elements in H, and so

$$|H| \geq \binom{t+k}{t-1}$$

□

In addition to a solid lower bound we can create an upper bound on the order of H in the special case where n is not the power of a prime.

Lemma 5.2.23. *If n is not a power of a prime, $|H| \leq n^{\sqrt{t}}$*

Proof. Consider the subset of I

$$I' = \{\left(\frac{n}{p}\right)^i * p^j | 0 \leq i, j \leq \lfloor\sqrt{t}\rfloor\}$$

Assuming $\frac{n}{p} \neq p^b$ for some power b,i.e. assuming that n is not a power of p, two elements of I' are equal $(\frac{n}{p})^{i_1} * p^{j_1} = (\frac{n}{p})^{i_2} * p^{j_2}$ if and only if $i_1 = i_2$ and $j_1 = j_2$. This implies that there are $(\lfloor\sqrt{t}\rfloor + 1)^2$ distinct elements in I'. Since $(\lfloor\sqrt{t}\rfloor + 1)^2 > t$, there are more elements in I' than there are equivalence classes of elements of I modulo r in G, which implies that at least two elements of I' must be equivalent modulo r. Let us call these elements m_1, m_2, and without loss of generality let $m_1 > m_2$. Since $m_1 \equiv m_2 \pmod{r}$, by the definition of modular equivalence $m_1 = q_1 * r + w$ and $m_2 = q_2 * r + w$. Consider

$$x^{m_1} = x^{q_1 * r + w} = x^{q_1 * r} * x^w = (x^r)^{q_1} * x^w$$

Operating modulo $x^r - 1$, $x^r \equiv 1$, and so

$$x^{m_1} \equiv (x^r)^{q_1} * x^w \equiv 1^{q_1} * x^w \equiv x^w \pmod{x^r - 1}$$

Since this all applies for m_2 as well,

$$x^{m_1} \equiv x^w \equiv x^{m_2} \pmod{x^r - 1}$$

Let $f(x)$ be an arbitrary element of J. Since $f(x)$ is in J and m_1 and m_2 are in I, they are both introspective over $f(x)$ and so $f(x)^{m_1} \equiv f(x^{m_1}) \pmod{x^r - 1, p}$. By the above equivalence, $f(x^{m_1}) \equiv f(x^{m_2})$ which is of course in turn equivalent to $f(x)^{m_2}$, by the introspective property of m_2. Therefore we have shown that $f(x)^{m_1} \equiv f(x)^{m_2} \pmod{x^r - 1, p}$.

If we let $D(x) = x^{m_1} - x^{m_2}$, by what we have just shown, $f(x)$ is a zero of $D(x)$ in $\mathbb{F}$, because

$$D(f(x)) = f(x)^{m_1} - f(x)^{m_2} \equiv 0 \pmod{x^r - 1}$$

. Since $f(x)$ was an arbitrary element of J, it follows that every polynomial in J is a root of $D(x)$, so there must be at least $|J|$ roots in $\mathbb{F}$. Since $H = \{f(x) \pmod{x^r - 1, p} | f(x) \in J\}$, and $H \subseteq \mathbb{F}$ at least $|H|$ of those roots must be distinct in $\mathbb{F}$. Therefore, $|H|$ must be less than or equal to the degree of $D(x)$. Since we chose $m_1 > m_2$, $deg(D(x)) = m_1$, and since m_1 was in I', by the construction of I' $m_1 \leq (\frac{n}{p} * p)^{\lfloor\sqrt{t}\rfloor} \leq n^{\sqrt{t}}$

Thus $|H| \leq m_1 \leq n^{\sqrt{t}}$ □

Using the combination of these two bounds, we can finally prove the accuracy of our algorithm.

Theorem 5.2.24. *If the algorithm outputs PRIME in step 6, n is prime*

Proof. Suppose the algorithm outputs PRIME in step 6. Using the previous work and definitions, we have from Lemma 5.2.21 that

$$|H| \geq \binom{t+k}{t-1}$$

By definition, $t > \log^2(n)$, which implies that $\sqrt{t} > \log(n)$, and thus $t > \sqrt{t} * \log(n)$. This in turn implies that $t \geq \sqrt{t} * \log(n) + 1$, and so by substitution

$$|H| \geq \binom{k + \lfloor \sqrt{t} \log(n) \rfloor + 1}{\lfloor \sqrt{t} \log(n) \rfloor}$$

Note that since $G \subseteq \mathbb{Z}/r\mathbb{Z}^*$, $|G| \leq |\mathbb{Z}/r\mathbb{Z}^*|$, and so $t \leq \phi(r)$. Therefore $\sqrt{t} \leq \sqrt{\phi(r)}$. Since $k = \lfloor \sqrt{\phi(r)} \log(n) \rfloor$, $k \geq \lfloor \sqrt{t} \log(n) \rfloor$ and so again by substitution

$$|H| \geq \binom{2\lfloor \sqrt{t} \log(n) \rfloor + 1}{\lfloor \sqrt{t} \log(n) \rfloor}$$

For the sake of brevity, let $s = \lfloor \sqrt{t} \log(n) \rfloor$. By definition of binomial notation

$$\binom{2s+1}{s} = \frac{(2s+1)!}{(s)! * (s+1)!} = \frac{(2s+1) * (2s) * ... * (s+2)}{(s) * (s-1) * ... * (2) * (1)}$$

Note that the i-th term in the numerator is strictly greater than twice the i-th term in the denominator, and so by applying this inequality across the board, and seperating out the $(s+2)$, we get

$$\frac{(2s+1) * (2s) * ... * (s+3) * (s+2)}{(s) * (s-1) * ... * (2) * (1)} > \frac{2(s) * 2(s-1) * ... * 2(2)}{(s) * (s-1) * ...(2)} * \frac{(s+2)}{1} = 2^{s-1} * \frac{s+2}{1}$$

Substituting back in the $\lfloor \sqrt{t} \log(n) \rfloor$ we have

$$\binom{2(\lfloor \sqrt{t} \log(n) \rfloor) + 1}{\lfloor \sqrt{t} \log(n) \rfloor} > 2^{\lfloor \sqrt{t} \log(n) \rfloor - 1} * (\lfloor \sqrt{t} \log(n) \rfloor + 2)$$

As established above, $\sqrt{t} > \log(n)$, so $\lfloor \sqrt{t} \log(n) \rfloor > \log^2(n) \geq 1$ and therefore $\lfloor \sqrt{t} \log(n) \rfloor \geq 4$, so

$$2^{\lfloor \sqrt{t} \log(n) \rfloor - 1} * (\lfloor \sqrt{t} \log(n) \rfloor + 2) \geq 2^{\lfloor \sqrt{t} \log(n) \rfloor - 1} * 2^2 = 2^{\lfloor \sqrt{t} \log(n) \rfloor + 1}$$

Putting all these inequalities together, we have that

$$|H| \geq \binom{2(\lfloor \sqrt{t}\log(n) \rfloor) + 1}{\lfloor \sqrt{t}\log(n) \rfloor} > 2^{\lfloor \sqrt{t}\log(n) \rfloor + 1}$$

Note that

$$2^{\lfloor \sqrt{t}\log(n) \rfloor + 1} \geq 2^{\log(n) * \sqrt{t}} = (2^{\log(n)})^{\sqrt{t}} = n^{\sqrt{t}}$$

And so $|H| > n^{\sqrt{t}}$

However, taking the contrapositive of Lemma 5.2.23, $|H| > n^{\sqrt{t}}$ implies that n is a prime power, i.e. $n = p^u$ for some u. If $u > 1$, the algorithm would have identified n as composite in step 1. Therefore, $u = 1$, and so $n = p$, which means n is a prime. □

Now that we have verified the usefulness of our algorithm, it comes time to see what complexity class it lies in. We will do this by breaking the algorithm into its component steps.

In analyzing the time complexity of the AKS Primality test, it will prove useful to define a tool related to $\mathcal{O}$ notation

Definition 5.2.25. We define $\mathcal{O}'(f(x)) = \mathcal{O}(f(x) * poly(\log f(x)))$

The above definition seems complex, but the important thing to note for our purposes is that if a process takes $\mathcal{O}'(poly(\log(n))$ time, it will also take $\mathcal{O}(poly(\log(n)))$ time. The polynomials themselves will be different, but the polynomial time relation is preserved.

The purpose of this notation is that it allows us to combine related time complexities in an intuitive way, which will come in handy shortly

The proof of the time complexity of step 1 is unfortunately beyond the scope of this text, and will be taken as a lemma.

Lemma 5.2.26. *Verifying for a given n that $n \neq a^b$ for some $b > 1$ takes $\mathcal{O}'(\log^3(n))$ time*

Step 2 involves finding an r which fulfills the condition that the order of n modulo r is greater than $\log^2(n)$

Lemma 5.2.27. *Finding an appropriate r in step 2 takes $\mathcal{O}'(log^7(n))$ time*

Proof. For a given r, the process of testing the order of n modulo r involves testing that $n^k \not\equiv 1 \pmod{r}$ for all $k \leq \log^2(n)$. In effect, this is performing $\mathcal{O}(log^2(n))$ operations modulo r, and so the process will take $\mathcal{O}'(\log^2(n)\log(r))$ time. Since $r \leq \log^5(n)$, we need to test at most $\log^5(n)$ r's, and so taking that into account the entire step will take $\mathcal{O}'(\log^7(n))$ time □

Step 3 involves taking the gcd of r numbers. The process of taking a gcd with n requires $\mathcal{O}(\log(n))$ steps, and since we have to do r of them the total time is $\mathcal{O}(r\log(n))$. Since $r \leq \log^5(n)$, we can rewrite that as $\mathcal{O}(\log^6(n))$.

Step 4 is simply checking whether $n \leq r$, which takes only $\mathcal{O}(\log(n))$ time.

It is in step 5 is where things start getting a little more complex.

Lemma 5.2.28. *Step 5 of the algorithm takes $\mathcal{O}'(\log^{21/2}(n))$ time.*

Proof. The meat of step 5 is checking $\lfloor\sqrt{\phi(r)}\log(n)\rfloor$ equations to make sure they are equal. The polynomials involved have degree r and coefficients with size $\mathcal{O}(\log(n))$. Each check requires $\mathcal{O}(\log(n))$ multiplications, and so we simply combine all these factors to get that each equation takes $\mathcal{O}'(r\log^2(n))$ steps. Beacuse we are verifying $\lfloor\sqrt{\phi(r)}\log(n)\rfloor$ equations, the whole process takes time

$$\mathcal{O}'(r\log^2(n) * \sqrt{\phi(r)}\log(n)) \leq \mathcal{O}'(r\sqrt{\phi(r)}\log^3(n)) \leq \mathcal{O}'(r^{\frac{3}{2}}\log^3(n))$$

Again, since $r \leq \log^5(n)$, we substitute that in and get a total time complexity for step 5 of $\mathcal{O}'((\log^5(n)^{\frac{3}{2}} * \log^3(n)) = \mathcal{O}'(\log^{\frac{21}{2}}(n))$

□

In comparison to this lengthy final step, the rest of the steps time is insignificant and gets absorbed into the larger polynomial, making the total time required for this algorithm $\mathcal{O}'(\log^{\frac{21}{2}}(n))$. This, at last, gives us the result we have been looking for.

Theorem 5.2.29. *The problem of determining whether a given number n is prime is in P*

Finally we have a deterministic test for primality that operates in polynomial time on the size of n. The Miller-Rabin test is noticeably faster than the AKS Primality test, but the former has the major disadvantage of only deciding that a given number is likely to be prime. It is possible for the Miller-Rabin test to yield a false positive, although in that case we have been *quite* unlucky.

The naive primality test, checking whether every number less than the square root of n fails to divide it, takes $\mathcal{O}(\sqrt{n})$ steps, and therefore operates in exponential time on the size of n.

There have of course been other tests for primality over the years, but until AKS each of them suffered from one of these two flaws. Either they operated probablistically and had a chance of failure, or they took far too much time for large values of n.

In the field of cryptology choosing large primes is often vitally important for the security of the cryptosystem. For example, both the RSA and Elliptic Curve cryptosystems

rely on the use of large primes. To a cryptologist, possessing an algorithm like the AKS test without any of these flaws can be essential to maintaining the security and efficiency of the overall system.

6

PROBABILISTIC CRYPTOLOGY

In this section, we will think about the transmission of messages from the perspective of Probability Theory and use those concepts to define the uncertainty associated with plaintext messages. Additionally, we will discuss the encryption and decryption methods of the only cryptosystem proven to be completely secure.

6.1 INFORMATION THEORY (BY JOURNEY PENNEY)

Information Theory is a field of mathematics that explores the quantification and communication of information. The field was introduced in 1948 by Claude Shannon in his published paper called "A Mathematical Theory of Communication". In this paper, Shannon proposed new concepts and mathematical models which showed that information of all forms (text, sound, image, video, etc.) can be quantified– specifically it can be quantified into binary digits, known as bits. Shannon's ideas were very influential, as they demonstrated how we can unify different modes of communication. He also incorporated Probability Theory into his concepts by associating the information quantities with probability values. Although Shannon never personally used the phrase "information theory" in his paper, the name most likely stemmed from his emphasized usage of the word "information". Additionally, it is important to note that the field of Information Theory focuses on how information is transmitted, not the information itself. The following is a quote from an interview with Richard Fano, who worked together with Claude Shannon: "I didn't like the term Information Theory. Claude didn't like it either. You see, the term 'information theory' suggests that it is a theory about information – but it's not. It's the transmission of information, not information" [12].

The transmission of information is very relevant in cryptography, where we study the communication between Alice and Bob through the encryption and decryption of messages. In fact, one year after publishing "A Mathematical Theory of Communication", Claude Shannon then applied his ideas to the field of cryptography in a paper called "Communication Theory of Secrecy Systems" (1949). This paper became a foundation for modern cryptography [13].

In particular, Shannon introduced the concepts of Entropy and Perfect Secrecy in his two previously mentioned papers from 1948 and 1949. We will see that these concepts are important in cryptography with regards to our message, key, and ciphertext spaces.

6.1.1 *Probability Theory*

Before we can begin to discuss Entropy and Perfect Secrecy, we must first cover the following basic terms and concepts from Probability Theory [14].

Definition 6.1.1 (Probability)**.** For the purposes of this book, we can informally define *probability* as the likelihood that some event will occur.

Definition 6.1.2 (Random Phenomenon)**.** A *random phenomenon* is any process that results in an outcome that cannot be predicted with certainty.

Note: A random phenomenon can refer to practically anything, such as the outcome of a coin flip, the number of days of rain per year in a particular city, or the height of a randomly selected person in the world.

Definition 6.1.3 (Random Variable)**.** A *random variable* X is a variable whose value is a possible outcome of a random phenomenon. Each outcome has an associated probability value, denoted P(X=x).

Example 6.1.4. Let the random variable X denote the outcome of rolling a 6-sided fair die. Then, our possible values of X are $x = 1, x = 2, x = 3, x = 4, x = 5$, and $x = 6$. Since we are dealing with a fair die, the probability of any outcome is $\frac{1}{6}$. As an example, the notation for the probability of rolling a 2 is: $P(X = 2) = \frac{1}{6}$.

Definition 6.1.5 (Conditional Probability)**.** Let A and B be any two random variables. Then, the *conditional probability*, $P(A = a|B = b)$, is the probability that A takes on some particular value, given that we already know the value of B.

Definition 6.1.6 (Independent Random Variables)**.** For any two random variables A and B, we say that A and B are *independent* if the value of one variable has no effect on the probability distribution of the other variable, i.e. $P(A = a|B = b) = P(A = a)$ and $P(B = b|A = a) = P(B = b)$ for all possible values of a and b.

Definition 6.1.7 (Properties in Probability Theory)**.** For any random variables A and B,

- $0 \leq P(A) \leq 1$
- $\sum_{a \in A} P(a) = 1$

- $P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$
- A and B are independent if and only if $P(A \cap B) = P(A)P(B)$

Theorem 6.1.8 (Bayes' Theorem)**.** *Let A and B be random variables. If $P(B) > 0$, then*

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}.$$

Proof. The proof of Bayes' Theorem follows directly from the third property in Definition 6.1.7. Suppose that A and B are random variables such that $P(B) > 0$. We know that $P(A \cap B) = P(A|B)P(B)$ and $P(A \cap B) = P(B|A)P(A)$. Therefore, by equating the two equations, we have

$$P(A|B)P(B) = P(B|A)P(A),$$

which implies that

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}.$$

□

6.1.2 *Entropy*

Now that we have a better grasp of Probability Theory, we can begin to study entropy. For a random variable X, the entropy of X can be thought of as the average amount of uncertainty that exists about the outcome of an experiment (that has not occurred yet) involving X. In Probability Theory, an experiment simply refers to a procedure that can be repeated and has a set of possible outcomes. We more formally define entropy as follows, for the case that our random variable is discrete. This simply means that the number of possible outcomes for the random variable is countable.

Definition 6.1.9 (Entropy)**.** Let X be a discrete random variable. The *entropy* of X, denoted $H(X)$ is

$$H(X) = \sum_{x \in X} P(X = x) \log_2 \left(\frac{1}{P(X = x)} \right)$$

Remark 6.1.10. Although the base of the logarithm is arbitrary [15], we typically use $\log_2$ so that entropy is measured in bits, which we mentioned earlier is a binary digit (0 or 1). This relates to Claude Shannon's idea that all information can be quantified; specifically, all information can be encoded into 0's and 1's using a given coding mechanism. Thus, entropy can also be used to describe the average number of bits required to represent an outcome of a given random variable.

Remark 6.1.11. We note that the lower bound on entropy is 0, and the upper bound on entropy is $\log_2(n)$, when the random variable has n possible outcomes [16]. We will discuss when it is the case that a random variable would have an entropy of 0 and when a random variable would have an entropy of $\log_2(n)$.

First, consider the lower bound of entropy. We will have an entropy of 0 when we have a random variable X, in which one of its outcomes x_1 has a probability of 1, while the other outcomes $x_2, x_3, ..., x_n$ have a probability of 0. In this case, we see that $P(X = x_i) = 0$ for $2 \leq i \leq n$ in the first part of the product in the summation of Definition 6.1.9. Thus, when computing entropy, the only outcome left to consider in the summation is $x_1 \in X$ as follows:

$$\begin{aligned} H(X) &= \sum_{x \in X} P(X = x) \log_2 \left(\frac{1}{P(X = x)} \right) \\ &= P(X = x_1) \log_2 \left(\frac{1}{P(X = x_1)} \right) \\ &= (1) \log_2(1) \\ &= 0 \end{aligned}$$

This makes sense, because if an outcome of a random variable has a 100% probability of occurring, then entropy should be 0 since there is no uncertainty about what the outcome might be.

Next, we will consider the upper bound of entropy. We will have entropy of $\log_2(n)$ when we have a random variable Y, in which all of its outcomes $y_1, y_2, ..., y_n$ are equally probable. This would imply that $P(Y = y_i) = \frac{1}{n}$ for $1 \leq i \leq n$. We calculate the entropy of Y as follows:

$$\begin{aligned} H(Y) &= \sum_{y \in Y} P(Y = y) \log_2 \left(\frac{1}{P(Y = y)} \right) \\ &= \sum_{y \in Y} \frac{1}{n} \log_2 \left(\frac{1}{\frac{1}{n}} \right) \\ &= (n)\left(\frac{1}{n}\right) \log_2(n) \\ &= \log_2(n) \end{aligned}$$

Since this is the upper bound, we see that uncertainty will be highest when each possible outcome has an equal probability of occurring. This also makes sense, because when all outcomes are equally probable, one can do no better than guessing at random which of the outcomes actually occurs.

Now, in order to apply the concept of entropy to cryptosystems, we simply consider our message space (M), key space (K), and ciphertext space (C) to be random variables.

Note that we assume K and M are independent random variables since the key is selected ahead of time and has no effect on the message that will later be encrypted.

Given our knowledge of Conditional Probabilities, we can now understand Claude Shannon's following definition.

Definition 6.1.12 (Message Equivocation)**.** Consider the message space M and key space K to be random variables. The *message equivocation* is defined to be $H(M|C)$, i.e. the entropy of the message given the ciphertext.

Note that the value of the message equivocation is of interest in the field of cryptography, because it refers to the uncertainty remaining about the plaintext message, given that the ciphertext has been intercepted (and thus is known) by the adversary [17].

We realize that a highly secure cryptosystem will maximize its message entropy, because there should be a lot of uncertainty in what the particular plaintext message might be. In fact, it would be ideal if a cryptosystem has $H(M|C) = H(M)$. This would imply that the uncertainty about the plaintext message is not lessened by knowing the ciphertext; the uncertainty about the message is the same before and after the ciphertext has been intercepted.

6.1.3 *Perfect Secrecy*

The previous statement, which mentioned the ideal situation of having $H(M|C) = H(M)$, is actually true for a cryptosystem that is perfectly secure. This type of cryptosystem is known for being "unbreakable" and is more formally defined below.

Definition 6.1.13. A cryptosystem (M, K, C, e, d) achieves *perfect secrecy* if

$$P(M = m|C = c) = P(M = m)$$

for all $m \in M$ and $c \in C$.

Thus, even if a ciphertext is intercepted by an adversary, the ciphertext reveals no information about the original message. The probability of any particular message m is the same probability whether the ciphertext has been seen or not.

Remark 6.1.14. Perfect secrecy is also frequently described in context of *posterior* and *prior* probabilities. Perfect secrecy is achieved if the posterior probability of the plaintext message given knowledge of the ciphertext is equal to the prior probability of the plaintext message. Posterior and prior probabilities are common terms used in Probability Theory. Basically, the prior probability is a probability value of a random variable before any additional evidence/information is known, and the posterior probability is a probability

value of a random variable given that there is information known about another random variable.

Notice that a cryptosystem that achieves perfect secrecy is very interesting and different from the other cryptosystems that we have studied thus far. For example, the security of RSA relies on the assumption that the adversary cannot factor a large composite number N within a reasonable number of steps, while the security of the multiplicative DLP assumes that the problem is only solvable in exponential-time (using the trial-and-error algorithm). On the other hand, with a perfectly secure cryptosystem, the adversary can have unbounded computational power and still will not have the resources to break the cryptosystem. Essentially, when dealing with perfect secrecy, the adversary can do no better than guessing at random what the plaintext message might be.

The following terms [18] are used to describe the type of security of a cryptosystem.

- Computational Security
  We say that a cryptosystem is *computationally secure* if the best algorithm for breaking such cryptosystem requires N operations, where N is a large number. Essentially, this is saying that the amount of computing time required is not practical. For example, a cryptosystem that relies on the Discrete Logarithm Problem is computationally secure.
- Provable Security
  We say that a cryptosystem is *provably secure* if it can only be broken by somehow reducing the original problem. An example of this situation would be a cryptosystem that is only secure under the assumption that an integer n cannot be factored, but could be broken otherwise. Thus, a cryptosystem that relies on RSA is an example of this type of security.
- Unconditional Security
  We say that a cryptosystem is *unconditionally secure* if it cannot be broken even if the adversary has infinite computational resources. This is the case for cryptosystems that achieve perfect secrecy.

Remark 6.1.15. When studying perfect secrecy, we make the reasonable assumption that M and C have probability distributions where all messages and ciphertexts occur with nonzero probability. Since probabilities range from 0 to 1 by Definition 6.1.7, this means we only need to consider

$$m \in M \text{ such that } P(M = m) > 0 \quad \text{and} \quad c \in C \text{ such that } P(C = c) > 0.$$

This is because, if we have some message $m_o \in M$ with $P(M = m_o) = 0$, then we trivially have $P(M = m_o|C = c) = 0 = P(M = m_o)$ for all $c \in C$. Also, if there exists some ciphertext $c_o \in C$ with $P(C = c_o) = 0$, then no message will ever be encrypted as c_o and, therefore, c_o can be omitted from C. Thus, we only consider messages and ciphertexts with nonzero probabilities.

Theorem 6.1.16 (Necessary Condition for Perfect Secrecy)**.** *If a cryptosystem* (M, K, C, e, d) *achieves perfect secrecy, then the dimension of the key space must be at least as large as the dimension of the message space, i.e.* $dim(K) \geq dim(M)$.

Proof. Assume that (M, K, C, e, d) is a cryptosystem that achieves perfect secrecy. Suppose that both M and C have probability distributions such that all $m \in M$ and all $c \in C$ occur with nonzero probability.

By contradiction, suppose that $\dim(K) < \dim(M)$. Let $c = e_k(m)$ for some $m \in M$ and $k \in K$. Next, we can decrypt c using all possible keys $k \in K$ and define $M' = \{m' \in M : m' = d_k(c) \text{ for some } k \in K\}$. Note that there is a unique key used for every message in M', and that there may be some keys in K that encrypt messages that do not yield our particular ciphertext c. Thus, we see that

$$\dim(M') \leq \dim(K).$$

Combining this with our earlier assumption, we have

$$\dim(M') \leq \dim(K) < \dim(M).$$

This implies that there exists some message $m_o \in M$ that does not belong to M'. Thus, $P(M = m_o|C = c) = 0$. Also, recall from our assumption that all messages occur with nonzero probability. Since all probability values range from 0 to 1 by Definition 6.1.7, we have $P(M = m_o) > 0$. It follows that

$$P(M = m_o|C = c) = 0 < P(M = m_o),$$

which implies that

$$P(M = m_o|C = c) \neq P(M = m_o).$$

Notice that this is a contradiction to the definition of perfect secrecy. Therefore, we have shown that

$$\dim(K) \geq \dim(M).$$

□

Remark 6.1.17. Theorem 6.1.16 implies that for any message $m \in M$, the key k used to encrypt m must be at least the same length as m. Using Shannon's idea that all information can be quantified by encoding it into 0's and 1's, it is typical to take our key and message spaces to be $\mathbb{Z}/2Z$ to some power. More specifically, if $K = (\mathbb{Z}/2\mathbb{Z})^a = \{0,1\}^a$ and $M = (Z/2\mathbb{Z})^b = \{0,1\}^b$, then Theorem 6.1.16 states that perfect secrecy requires $a \geq b$.

Theorem 6.1.18 (Modified Shift Cipher Achieves Perfect Secrecy)**.** *Suppose that the Shift Cipher has been modified such that*

- $M = K = C = \mathbb{Z}_{26}$

- *The key changes after the encryption of each letter in the message*
- *The keys are selected at random, where each of the 26 keys is used with equal probability,* $\frac{1}{26}$

Then, for any probability distribution of M, the modified Shift Cipher achieves perfect secrecy.

Proof. We will prove that the described cryptosystem achieves perfect secrecy using Bayes' Theorem.

We can begin by finding the probability distribution on C. In other words, we want to compute $P(C = c)$, which we note is one component of Bayes' Theorem. Let $c \in \mathbb{Z}_{26}$. We see that

$$\begin{aligned} P(C = c) &= \sum_{k \in \mathbb{Z}_{26}} P(K = k) P(M = d_k(c)) && (1) \\ &= \sum_{k \in \mathbb{Z}_{26}} \frac{1}{26} P(M = d_k(c)) && (2) \\ &= \frac{1}{26} \sum_{k \in \mathbb{Z}_{26}} P(M = d_k(c)) && (3) \\ &= \left(\frac{1}{26}\right)(1) && (4) \\ &= \frac{1}{26} && (5) \end{aligned}$$

For (1), notice that in order to get a particular c, we need both a key k and a message m such that $m = d_k(c)$, which we can use to write $P(C = c)$ in a different form. Since K and M are independent, we know by Definition 6.1.7 that we can multiply the probabilities of our key being k and our message being $d_k(c)$. But, we also need to sum this product over all keys in K, because there could be multiple keys that can encrypt a message into c.

In (2), we replace $P(K = k)$ with $\frac{1}{26}$. This is because every key has an equal probability of $\frac{1}{26}$ of being selected from K.

Then, (3) simply takes $\frac{1}{26}$ outside of the summation.

For (4), we note that $\sum_{k \in \mathbb{Z}_{26}} P(M = d_k(c))$ is the probability that our message is the decryption of our particular c using key k, but also summed over all keys in our entire key space. This covers all possibilities, so by Definition 6.1.7 is equal to 1.

Thus, in (5), we have $P(C = c) = \frac{1}{26}$.

Our next computation is to find $P(C = c | M = m)$, which is another component of Bayes' Theorem.

We see that we can compute the probability as follows:

$$\begin{aligned} P(C = c|M = m) &= P(K = (c - m) \mod 26) \\ &= \frac{1}{26} \end{aligned}$$

Recall that there is a unique key for every pair of message m and ciphertext c. Thus, given a particular message m, the probability of a particular ciphertext c will be the same as the probability that k is, in fact, a key that encrypts m into c. Note that $c = m + k \pmod{26}$ in this shift cipher. Therefore, the key must satisfy $k = c - m \pmod{26}$. In other words, we are looking for $P(K = (c - m) \pmod{26})$. Next, we notice that $P(K = (c - m)) \mod 26)$ is the probability that the key is the specific key that encrypts m into c. Since the key is randomly selected from K of uniform distribution, the probability of any such key is equal to one divided by the total number of possible keys, or $\frac{1}{26}$.

Now, we can use Bayes' Theorem. Substituting in the two probabilities that we obtained above, it follows that

$$\begin{aligned} P(M = m|C = c) &= \frac{P(M = m)P(C = c|M = m)}{P(C = c)} \\ &= \frac{P(M = m)\frac{1}{26}}{\frac{1}{26}} \\ &= P(M = m) \end{aligned}$$

Thus, we have shown, by definition, that the modified shift cipher achieves perfect secrecy. □

6.1.4 *One-Time Pad*

The One-Time Pad is the cryptosystem known for achieving perfect secrecy. It was invented by Gilbert Vernam in 1917. To begin describing the process of the One-Time Pad, we let $n \geq 1$ be an integer and $M = K = C = (\mathbb{Z}/2\mathbb{Z})^n$. Suppose that K is uniformly distributed. Thus, each key has an equal probability of being selected from K, i.e. $P(K = k) = \frac{1}{2^n}$ for any $k \in K$.

We can illustrate the processes of the Key Creation, Encryption, and Decryption by again considering Alice and Bob.

- **Key Creation:** Alice and Bob randomly choose a secret key $k = (k_1, k_2, ..., k_n)$, which is to be used only once.

- **Encryption:** In order to encrypt a message $m = (m_1, m_2, ..., m_n)$ for Alice, Bob uses the encryption function

$$e_k(m) = (m_1 + k_1, m_2 + k_2, ..., m_n + k_n) \pmod 2$$

- **Decryption:** Upon receiving Bob's ciphertext, Alice can decrypt it using the decryption function

$$d_k(c) = (c_1 + k_1, c_2 + k_2, ..., c_n + k_n) \pmod 2$$

Note: This process of performing addition modulo 2, bit by bit, is also commonly referred to as the "exclusive or", XOR, logical operator.

Example 6.1.19. Suppose that the following key k has been randomly chosen as a one-time pad: $k =$ 1100010 1010111 1100010 1010001 0011110 1101001. Show how the key k can be used to encrypt the message "ATTACK" (which can be converted into 1000001 1010100 1010100 1000001 1000011 1001011 using ASCII binary code) and can be used to decrypt the associated ciphertext.

- **Encryption:**
  1000001 1010100 1010100 1000001 1000011 1001011 (plaintext)

  1100010 1010111 1100010 1010001 0011110 1101001 (key)

  0100011 0000011 0110110 0010000 1011101 0100010 (ciphertext)
- **Decryption:**
  0100011 0000011 0110110 0010000 1011101 0100010 (ciphertext)

  1100010 1010111 1100010 1010001 0011110 1101001 (key)

  1000001 1010100 1010100 1000001 1000011 1001011 (plaintext)

Theorem 6.1.20 (Perfect Secrecy of the One-Time Pad)**.** *The One–Time Pad achieves perfect secrecy.*

Proof. Assume that $M = K = C = (\mathbb{Z}/2\mathbb{Z})^n$ for some integer $n \geq 1$. Let $m \in M$ and $c \in C$. For any $k \in K$, define $C(k) = \{e_k(m) : m \in M\}$.

We proceed with a similar process as the proof of Theorem 6.1.18, by calculating probabilities and applying Bayes' Theorem.

First, we wish to find $P(C = c)$. Notice that we have

$$\begin{aligned} P(C = c) &= \sum_{k:c \in C(k)} P(K = k) P(M = d_k(c)) \\ &= \sum_{k:c \in C(k)} \frac{1}{2^n} P(M = d_k(c)) \end{aligned}$$

This is because, using similar logic as the proof of Theorem 6.1.18, we can replace $P(C = c)$ with $\sum_{k:c\in C(k)} P(K = k)P(M = d_k(c))$. We note that in order to get a particular c, we need both a key k and a message m such that $m = d_k(c)$. Since K and M are independent, we know by Definition 6.1.7 that we can multiply the probabilities of our key being k and our message being $d_k(c)$. But, to cover all such ciphertexts, we must sum over all keys where $c \in C(k)$, because there could be multiple keys that produce this ciphertext c. Then, $P(K = k)$ can be replaced with $\frac{1}{2^n}$ since the K is uniformly distributed, meaning that any key has an equal probability of being selected from K.

Next, we will determine $P(C = c|M = m)$. We see that

$$\begin{aligned} P(C = c|M = m) &= P(K = (c - m)) \mod 2) \\ &= \frac{1}{2^n} \end{aligned}$$

First, we replace $P(C = c|M = m)$ with $P(K = (c - m)) \mod 2)$ above. Recall that there is a unique key for every pair of message m and ciphertext c. Thus, given a particular message m, the probability of a particular ciphertext c will be the same as the probability that k is a key that encrypts m into c. Because $c = m + k \pmod 2$ by the encryption process of the One-Time Pad, we know that the key must satisfy $k = c - m \pmod 2$. In other words, we are looking for $P(K = (c - m) \mod 2)$.

Then, $P(K = (c - m) \mod 2)$, the probability that our key was specifically the key that encrypts m into c, is simply $\frac{1}{2^n}$. This is because the key is randomly selected from K with uniform distribution, where every key has a probability equal to one divided by the total 2^n number of possible keys. We know that there are 2^n possible keys, because each digit of the key has two options (0 or 1) and the sequence of digits is of length n.

Now, we can apply Bayes' Theorem. We see that

$$\begin{aligned} P(M = m|C = c) &= \frac{P(C = c|M = m)P(M = m)}{P(C = c)} && (1) \\ &= \frac{\frac{1}{2^n}P(M = m)}{\sum_{k:c\in C(k)} \frac{1}{2^n}P(M = d_k(c))} && (2) \\ &= \frac{P(M = m)}{\sum_{k:c\in C(k)} P(M = d_k(c))} && (3) \\ &= \frac{P(M = m)}{1} && (4) \\ &= P(M = m) && (5) \end{aligned}$$

To begin, (1) is a direct application of Bayes' Theorem.

In (2), we make the substitutions for $P(C = c|M = m)$ and $P(C = c)$ that were discussed above.

In (3), we are able to take the $\frac{1}{2^n}$ from the denominator outside of the summation and then cancel it with the $\frac{1}{2^n}$ in the numerator.

Then in (4), we have $\sum_{k:c\in C(k)} P(M = d_k(c))$ in the denominator. This is the probability that our message is the decryption of our particular c using key k, and summed over all keys that could have yielded c. This covers all possibilities is equal to 1 by Definition 6.1.7.

After simplifying, (5) shows that we are left with $P(M = m)$. Thus, we have shown, by definition, that the One-Time Pad is perfectly secure. □

Although the One-Time Pad is theoretically great, there are some limitations to this cryptosystem (and perfect secrecy in general) that are important to discuss.

- The length of the key cannot be any shorter than the length of the message

  The key, which must be selected ahead of time, must be long enough to accommodate whatever the future message might be. This is not practical, because a very long key is often needed for a decently sized message.

- After a key is used to encrypt one message and decrypt the associated ciphertext, the same key should never be reused

  It is not convenient, but to maintain perfect secrecy, every new message will require a new random key for the encryption. If the same key is used more than once, then information is revealed about the plaintext messages. For example, suppose that messages m and m' were both encrypted using the same key k. We would have

  $$c = e_k(m) = (m_1 + k_1, m_2 + k_2, ..., m_n + k_n) \pmod 2$$
  $$c' = e_k(m') = (m'_1 + k_1, m'_2 + k_2, ..., m'_n + k_n) \pmod 2$$

  Note that an adversary could compute

  $$c + c' \equiv (m_1 + k_1 + m'_1 + k_1, ..., m_n + k_n + m'_n + k_n) \equiv m + m' \pmod 2$$

  and then could perform language analysis to determine the messages.

  In fact, several of the Soviet Union's intelligence agencies encrypted their messages using a one-time pad system; however, they made this mistake of reusing the same one-time pads. Because of this error, the NSA was able to decrypt the Soviets' messages during a counterintelligence program called the Venona project [19].

- The key must be truly random

  To acquire a random sequence of numbers is actually a difficult task, especially for a long key. Computer programs with random number generators use algorithms;

thus, the numbers are not truly random and are actually referred to as pseudo-random. Obtaining a truly random sequence of numbers often involves the task of measuring some physical phenomenon expected to be random (such as atmospheric noise) [20].

Given the above limitations, one may wonder when we would want to go through the trouble of guaranteeing perfect secrecy. Well, it is important when dealing with very sensitive information. For example [21]:

- The CIA has used one-time pads to send messages between their headquarters and agents in the field.
- The NSA has used one-time pads to send nuclear launch messages.
- During World War II, the British Special Operations Executive used one-time pads when communicating between offices.

Overall, with regards to both the limitations and the applications of the One-Time Pad, we realize that this type of cryptosystem is only useful in a very specific type of situation. To establish a shared key that must be communicated privately and that can only be used for one encryption, it requires both parties to have easy access to frequent and completely secure communication. Notice that if the message was already determined, then Bob could simply tell Alice the message in private when they had planned on meeting to select a key. However, the reason that they go to great lengths to secretly communicate just the key is because the message does not exist yet; this cryptosystem is utilized in situations where the message is going to emerge unexpectedly and urgently. Therefore, it is well worth going through all of the requirements of the One-Time Pad (length, one-time use, and randomness of the key), and also the precautions to privately establish the key ahead of time, in situations where Alice and Bob must be prepared to securely communicate an important message that might transpire at any given time.

6.2 PROBABILISTIC ENCRYPTION (BY MARGARET ALTENHOF-LONG)

As discussed in previous sections, the goal of public key cryptosystems is to communicate information over insecure channels without revealing that information to potential eavesdroppers. Further, we have seen that it is optimal for an encryption scheme to conceal even partial information about a plaintext message.

Definition 6.2.1. (Semantic Security). A cryptosystem is *semantically secure* if whatever information is efficiently computable about the plaintext given the ciphertext is also efficiently computable without the ciphertext.

Remark 6.2.2. While perfect secrecy means a given ciphertext reveals no information about its corresponding plaintext message, semantic security refers to the complexity

of extracting information about the plaintext given the ciphertext. Semantic security implies that it is computationally difficult to calculate information about the plaintext from the ciphertext.

6.2.1 *Probabilistic Cryptosystems*

In 1982, Shafi Goldwasser and Silvio Micali developed the concept of semantic security while trying to formulate a protocol for a mental poker game in which all partial information about a card, including suit and color, could be hidden [22]. This led to a new encryption model that sought to obscure all partial information about sent messages. As described in previous chapters, the security of public key cryptosystems rests in one-way trapdoor functions. Inspired mainly by the RSA encryption system, Goldwasser and Micali raised two main objections to the basic trapdoor model. They claimed that, assuming that a function f is trapdoor:

(1) It may still be possible to calculate x given $f(x)$ if x has a special form.

(2) It may be possible to calculate partial information about x given $f(x)$.

To address these objections, Goldwasser and Micali introduced the idea of the *unapproximable trapdoor predicate*.

Definition 6.2.3. (Unapproximable Trapdoor Predicate) A predicate B is *trapdoor* and *unapproximable* if anyone can select an x such that $B(x) = 0$ or a y such that $B(y) = 1$, but only someone with knowledge of the trapdoor information can calculate $B(z)$ for some given z.

Example 6.2.4. Give an example of an unapproximable trapdoor predicate.

Consider the predicate $B(x)$ defined by the statement "x is congruent to a perfect square modulo N". We see that if x is congruent to a perfect square modulo N, then $B(x) = 1$; otherwise, if x is not congruent to a perfect square modulo N, then $B(x) = 0$. As mentioned previously, it is difficult to compute n^{th} roots in a finite field, and thus it is difficult to decide whether a particular x has a square root, and it follows that B is unapproximable. However, we will soon see that knowledge of the factorization of N will provide trapdoor information to decide whether an integer x satisfies B.

Since a predicate B returns only 1's and 0's, there are multiple x's such that $B(x) = 0$ and multiple y's such that $B(y) = 1$. This leads to the idea of probabilistic encryption.

Definition 6.2.5. (Probabilistic Encryption). An encryption scheme is *probabilistic* if it uses randomness in its encryption algorithm so that the same plaintext message, when encrypted different times, may yield different ciphertexts.

For a message m represented by the binary bits $m_1, ..., m_k$, we can encode m by randomly selecting a series of x_i's such that $B(x_i) = m_i$. In this way, we see that the are many possible encodings of m, and thus an adversary will not be able to tell if the same message has been sent twice. Comparatively, a "deterministic" encryption scheme will send a plaintext to the same ciphertext each time, and thus, is vulnerable to chosen ciphertext attacks.

Goldwasser and Micali presented an implementation of the probabilistic model based on the quadratic residuosity problem. In the following sections, we explore the concept of quadratic residuosity and describe the protocol for the Goldwasser-Micali public key cryptosystem.

Quadratic Residuosity Problem

First described by Gauss in 1801, the quadratic residuosity problem involves determining whether numbers are congruent to perfect squares in composite moduli of unknown factorization.

Definition 6.2.6. (Quadratic Residue). Let a and N be integers. If there exists an x such that $a \equiv x^2 \pmod{N}$, then a is a *quadratic residue* modulo N. If no such x exists, then a is a *quadratic non-residue* modulo N.

The general idea of the quadratic residuosity problem is to determine for integers a and N whether a is a quadratic residue modulo N. In order to formalize the quadratic residuosity Problem in the setting of cryptography, we must first introduce some definitions and notation from number theory.

Definition 6.2.7. (Legendre Symbol). Let a be an integer and let p be an odd prime. The *Legendre symbol* $\left(\frac{a}{p}\right)$ is defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is a quadratic residue modulo } p \text{ and } a \not\equiv 0 \pmod{p}, \\ -1 & \text{if } a \text{ is a quadratic non-residue modulo } p, \\ 0 & a \equiv 0 \pmod{p} \end{cases}$$

The Legendre symbol provides an easy way to notate residues and non-residues in a prime modulus, but it is necessary to be able to calculate the value of a Legendre symbol. Fortunately, the following result allows us to determine $\left(\frac{a}{p}\right)$.

Theorem 6.2.8. *(Euler's Criterion). Let p be an odd prime. Then an integer $a \not\equiv 0$ modulo p is a quadratic residue modulo p if and only if $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$.*

Proof. Let p be an odd prime and let $a \not\equiv 0 \pmod{p}$.

First, suppose a is a quadratic residue modulo p. By definition of quadratic residue, there exists x such that $a \equiv x^2 \pmod{p}$. Thus, $a^{\frac{p-1}{2}} \equiv (x^2)^{\frac{p-1}{2}} \equiv x^{p-1} \equiv 1 \pmod{p}$ by Fermat's Little Theorem.

Now, suppose $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$. Let g be a generator of $\mathbb{Z}/p\mathbb{Z}$. Thus, $a \equiv g^k \pmod{p}$ for some integer k, and it follows that

$$1 \equiv a^{\frac{p-1}{2}} \equiv (g^k)^{\frac{p-1}{2}} \equiv g^{k(\frac{p-1}{2})} \pmod{p}.$$

Thus, by Proposition 1.2.3, $|g||k^{\frac{p-1}{2}}$, and since g is a generator, it follows that

$$(p-1)|k(\frac{p-1}{2}).$$

By definition of divides, $(p-1)l = k(\frac{p-1}{2})$ or equivalently $k = 2l$ for some integer l. Now, construct the element $j \equiv g^l \pmod{p}$. It follows that $j^2 \equiv g^{2l} \equiv g^k \equiv a \pmod{p}$. Thus, a is congruent to a square modulo p and is therefore a quadratic residue modulo p. □

Remark 6.2.9. In the previous theorem, since $a \not\equiv 0 \pmod{p}$, it follows by Fermat's Little Theorem that $a^{p-1} \equiv 1 \pmod{p}$. By Lemma 4.2.7, 1 and -1 are the only square roots of 1 modulo p, and thus $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$ or $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$. Thus, if a is not a quadratic residue, then $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$. Therefore, we can write

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}.$$

The Legendre symbol is only defined for a prime modulus. In order to explore quadratic residuosity in composite moduli, we turn to a generalization of the Legendre symbol, the Jacobi symbol.

Definition 6.2.10. (Jacobi Symbol). Let a and $N = p_1^{n_1} \cdots p_k^{n_k}$ be integers. The *Jacobi symbol* $\left(\frac{a}{N}\right)$ is the product of Legendre symbols for each prime factor of N:

$$\left(\frac{a}{N}\right) = \left(\frac{a}{p_1}\right)^{n_1} \cdots \left(\frac{a}{p_k}\right)^{n_k}.$$

Example 6.2.11. Let $N = 3599 = 59 \cdot 61$. Calculate the Jacobi symbols $\left(\frac{25}{N}\right)$ and $\left(\frac{30}{N}\right)$.

Using the definition of Jacobi symbol, we can split the Jacobi symbol into a product of Legendre symbols.

$$\left(\frac{25}{N}\right) = \left(\frac{25}{59 \cdot 61}\right) = \left(\frac{25}{59}\right)\left(\frac{25}{61}\right).$$

Now, applying Euler's Criterion, we have

$$\left(\frac{25}{59}\right)\left(\frac{25}{61}\right) = (25^{\frac{59-1}{2}} \pmod{59})(25^{\frac{61-1}{2}} \pmod{61}) \equiv (1)(1) \equiv 1.$$

For the second Jacobi symbol, we follow the same process:

$$\left(\frac{30}{59 \cdot 61}\right) = \left(\frac{30}{59}\right)\left(\frac{30}{61}\right) = (30^{\frac{59-1}{2}} \pmod{59})(30^{\frac{61-1}{2}} \pmod{61}) \equiv (-1)(-1) \equiv 1.$$

In the previous example, both Jacobi symbols have value 1. However, while 25 is a perfect square modulo N, 30 is not, which can be seen from the fact that 30 is a quadratic non-residue modulo both 59 and 61. A quadratic non-residue modulo N with Jacobi symbol equal to 1 is called a *pseudosquare*. Without the knowledge of the factorization of N, it is unclear whether a number with Jacobi symbol equal to 1 is a quadratic residue.

Definition 6.2.12. The *Quadratic Residuosity Problem* is to decide, given a composite integer $N = pq$ and an integer a such that $\left(\frac{a}{N}\right) = 1$, whether or not a is a quadratic residue modulo N.

When the factorization of $N = pq$ is unknown, the quadratic residuosity problem is assumed to be computationally hard. The difficulty of this problem serves as the basis for the Goldwasser-Micali system, whose security is further strengthened by the use of randomness in the encryption algorithm. We can think of the quadratic residuosity problem as the unapproximable trapdoor predicate "is quadratic residue modulo N". While there are ways of finding x's that satisfy the predicate and $y's$ that do not, without knowledge of the trapdoor information p and q, an adversary cannot easily determine whether a given z satisfies the predicate.

Goldwasser-Micali Cryptosystem

Like RSA, the Goldwasser-Micali system is based on three main components.

- **Key Creation**: Alice sets up her public/private key pair. Her private key consists of two large primes p and q. For her public key, she chooses a non-residue y modulo pq such that the Legendre symbols $\left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = -1$ and thus the Jacobi symbol $\left(\frac{y}{N}\right) = 1$. Alice then publishes y and the product N of p and q.
- **Encryption**: To send a message m to Alice, Bob encodes m as a string of bits $(m_1, ..., m_k)$. For each m_i, Bob randomly generates $x_i \in (\mathbb{Z}/N\mathbb{Z})^*$ and computes

  $$c_i = (x_i)^2 y^{m_i} \pmod{N}.$$

  Bob then sends the ciphertext

  $$c = (c_1, ..., c_k).$$

- **Decryption**: To decrypt, Alice uses p and q to determine whether each c_i is a quadratic residue modulo N. Since the chosen value y is not a perfect square modulo N, c_i is a square only if $m_i = 0$ and thus

$$c_i = (x_i)^2 y^0 = (x_i)^2.$$

  Therefore, if c_i is a quadratic residue, then $m_i = 0$; otherwise, $m_i = 1$.

Example 6.2.13. Suppose Alice published her public key $y = 30$ and $N = 3599$ where y is a non-residue modulo 3599. Alice receives the ciphertext

$$c = (2317, 2056, 8, 1425, 334, 1423, 2854, 2271, 838, 589, 1471, 379, 1989, 1849, 910, 2185).$$

Given that $N = pq$ where $p = 59$ and $q = 61$, decrypt the transmitted ciphertext.

Since we know the factorization of N, we can decide the residuosity of each c_i in the ciphertext. As in Example 6.2.11, we calculate each Jacobi symbol as a product of Legendre symbols.

$$\begin{aligned}
\left(\frac{2317}{3599}\right) &\equiv \left(\frac{2317}{59}\right)\left(\frac{2317}{61}\right) \equiv 1 \cdot 1 \\
\left(\frac{2056}{3599}\right) &\equiv \left(\frac{2056}{59}\right)\left(\frac{2056}{61}\right) \equiv -1 \cdot -1 \\
\left(\frac{8}{3599}\right) &\equiv \left(\frac{8}{59}\right)\left(\frac{8}{61}\right) \equiv -1 \cdot -1 \\
\left(\frac{1425}{3599}\right) &\equiv \left(\frac{1425}{59}\right)\left(\frac{1425}{61}\right) \equiv 1 \cdot 1 \\
\left(\frac{334}{3599}\right) &\equiv \left(\frac{334}{59}\right)\left(\frac{334}{61}\right) \equiv -1 \cdot -1 \\
\left(\frac{1423}{3599}\right) &\equiv \left(\frac{1423}{59}\right)\left(\frac{1423}{61}\right) \equiv 1 \cdot 1 \\
\left(\frac{2854}{3599}\right) &\equiv \left(\frac{2854}{59}\right)\left(\frac{2854}{61}\right) \equiv 1 \cdot 1 \\
\left(\frac{2271}{3599}\right) &\equiv \left(\frac{2271}{59}\right)\left(\frac{2271}{61}\right) \equiv 1 \cdot 1 \\
\left(\frac{838}{3599}\right) &\equiv \left(\frac{838}{59}\right)\left(\frac{838}{61}\right) \equiv 1 \cdot 1
\end{aligned}$$

$$\left(\frac{589}{3599}\right) \equiv \left(\frac{589}{59}\right)\left(\frac{589}{61}\right) \equiv -1 \cdot -1$$
$$\left(\frac{1471}{3599}\right) \equiv \left(\frac{1471}{59}\right)\left(\frac{1471}{61}\right) \equiv -1 \cdot -1$$
$$\left(\frac{379}{3599}\right) \equiv \left(\frac{379}{59}\right)\left(\frac{379}{61}\right) \equiv 1 \cdot 1$$
$$\left(\frac{1989}{3599}\right) \equiv \left(\frac{1989}{59}\right)\left(\frac{1989}{61}\right) \equiv -1 \cdot -1$$
$$\left(\frac{1849}{3599}\right) \equiv \left(\frac{1849}{59}\right)\left(\frac{1849}{61}\right) \equiv 1 \cdot 1$$
$$\left(\frac{910}{3599}\right) \equiv \left(\frac{910}{59}\right)\left(\frac{910}{61}\right) \equiv 1 \cdot 1$$
$$\left(\frac{2185}{3599}\right) \equiv \left(\frac{2185}{59}\right)\left(\frac{2185}{61}\right) \equiv -1 \cdot -1$$

When both Legendre symbols are 1, c_i is a quadratic residue modulo 3599, which implies $m_i = 0$. Otherwise, $m_i = 1$. Thus, we decipher the message as

$$m = (0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1).$$

Assuming Bob is representing letters as binary ASCII values, we can split the message into 8-bit segments. Looking up the values 01101000 and 01101001 in a standard ASCII chart, we find that Bob's message can be transliterated as "hi".

As seen in the previous example, even a short message can yield quite a lengthy ciphertext due to the bit-by-bit encryption. Practically speaking, in a modulus N with relatively large factors p and q, the Goldwasser-Micali system is inefficient.

We can analyze the system in terms of the four desirable properties of cryptosystems presented in Definition 2.2.1. For any key, in this case the non-residue y, and any plaintext, it is easy to calculate $e(m)$, though as we have seen, longer messages will take significantly more computation time. Similarly, it is is computationally easy to decrypt ciphertexts given knowledge of the factorization of N.

For the third property, having any number of ciphertexts, it remains computationally difficult to calculate $d(c)$ since the system is semantically secure, as will be discussed in the next section. As for the fourth property, it is interesting to note that a pair (m, c) of a plaintext and corresponding ciphertext is not unique since the algorithm is probabilistic. Thus, the system is safe against chosen plaintext attacks. In the next section we further explore the security of probabilistic encryption schemes.

6.2.2 *Security of Probabilistic Encryption Schemes*

The Goldwasser-Micali system is as hard for an adversary Eve to attack as the quadratic residuosity problem. The following proposition shows that the encryption is at least as hard to break as its underlying problem.

Proposition 6.2.14. *Fix $N = pq$ for distinct odd primes p and q. Suppose that Eve has access to an oracle that decrypts arbitrary Goldwasser-Micali ciphertexts encrypted using arbitrary non-residues y modulo N. Eve can use this oracle to solve the quadratic residuosity problem. Similarly, if Eve has access to an oracle which solves the quadratic residue problem, she can use it to decrypt messages that have been encrypted using the Goldwasser-Micali cryptosystem.*

Proof. Fix $N = pq$ for distinct odd primes p and q.

Suppose that Eve has access to an oracle that decrypts arbitrary Goldwasser-Micali ciphertexts encrypted using arbitrary non-residues y modulo N. In other words, the oracle returns

$$O(y, (c_1, ..., c_k)) = (m_1, ..., m_k)$$

for ciphertext $c = (c_1, ..., c_k)$ and plaintext $m = (m_1, ..., m_k)$. Thus, Eve knows c_i, m_i, and y. Since $c_i = (x_i)^2 y^{m_i}$, Eve knows that when $m_i = 1$, c_i is a quadratic non-residue modulo N since y is a quadratic non-residue modulo N. Similarly, when $m_i = 0$, $c_i = (x_i)^2$ and is thus a quadratic residue modulo N. Therefore, Eve can decide whether an arbitrary integer is a quadratic residue modulo N.

Now, suppose that Eve has access to an oracle which solves the quadratic residue problem. In other words, the oracle returns

$$O(a, N) = \left(\frac{a}{N}\right)$$

for an integer a. Thus, for an arbitrary c_i, Eve can decide whether c_i is a quadratic residue modulo N. Thus, if c_i is a quadratic residue, Eve knows that $m_i = 0$ and if c_i is a quadratic non-residue, Eve knows that $m_i = 1$. It follows that Eve can decrypt a message encrypted using the Goldwasser-Micali cryptosystem.

□

Further, Goldwasser and Micali proved that any probabilistic encryption scheme is semantically secure. Thus, it is in fact possible to communicate over insecure channels while concealing all partial information. The following theorems are stated here without proof[1].

1 For proofs, see Goldwasser and Micali. "Probabilistic Encryption", *Journal of Computer and System Sciences*, Vol. 28, No. 2.

Theorem 6.2.15. *Each probabilistic public key cryptosystem is polynomially secure. In other words, it is impossible, in polynomial time, to decide which plaintext m corresponds to a ciphertext c with a probability significantly greater than 0.5.*

Theorem 6.2.16. *Each polynomially secure public key cryptosystem is semantically secure.*

In a perfect world, we would want to conceal all partial information in every sensitive communication exchange. Thus, due to their many desirable properties, probabilistic cryptosystems remain of great interest in cryptography.

As discussed earlier, Goldwasser and Micali's original implementation is highly inefficient since it involves bit-by-bit encryption and decryption. However, other subsequent implementations have improved efficiency. Notably, in 1999 Pascal Paillier proposed a system based on the higher residuosity problem, an extension of the quadratic residuosity problem to higher powers [23]. More recently, in 2010, a generalization of the Paillier system, the Damgård-Jurik cryptosystem further saves time and computing power in the encryption and decryption processes. Both of the systems have applications in electronic voting [24].

Probabilistic cryptography offers a secure alternative to the classic deterministic models and remains a growing field of study.

7

ELLIPTIC CURVE CRYPTOSYSTEMS

The advent of the RSA cryptosystem spurred a tremendous amount of research on the factorization of large numbers. Several improved factorization methods were developed at the end of the 20th century. Included in these was an algorithm developed by Hendrik Lenstra in 1984. Lenstra's algorithm is an elliptic analogue of Pollards' $p-1$ factorization algorithm described in Section 4.2.3. Although less efficient than other methods developed around the same time, Lenstra's algorithm brought elliptic curves to the attention of the cryptographic community.

In 1985, Neal Koblitz and Victor Miller, working independently, came up with the idea of using elliptic curves to create public-key cryptosystems. Elliptic Curve Cryptography, or ECC, was born. ECC is based on the algebraic structure of elliptic curves over finite fields. In particular, elliptic curve cryptosystems are based on the presumed difficulty of the discrete log problem over elliptic curves. The size of the elliptic curve group determines the difficulty of the elliptic curve discrete log problem (ECDLP). It is believed that the same level of security afforded by an RSA-based system with large modulus can be achieved using a much smaller elliptic curve group. This reduces storage and transmission requirements, making implementation of the public-key cryptosystem more efficient and less costly.

As is true of all public-key cryptosystems, there is no proof of the security of ECC. The security rests in the *presumed* difficulty of some mathematical problem. That said, ECC has gained prominence in the last decade and was endorsed by the NSA, making it a recommended algorithm for protecting government information classified up to top secret.

7.1 ELLIPTIC CURVES

Equations of the form $Y^2 = X^3 + AX + B$ are called *Weierstrass Equations* after the German mathematician Karl Weirstrass[1] who studied them extensively in the last half of

1 Karl Weierstrass is the mathematical great^6-grandfather of Prof. McNicholas, and thus the mathematical great^7-grandfather of the authors of this text

the 19^{th} century. For a given Weierstrass equation E: $Y^2 = X^3 + AX + B$, the quantity $\Delta_E = 4A^3 + 27B^2$ is called the discriminant of E. The *curve* associated with a given Weierstrass equation is the set of points in the plane satisfying the given equation. Figures 7 and 8 show curves corresponding to Weierstrass equations with non-zero discriminants. Figure 9 shows a curve with $\Delta_E = 0$.

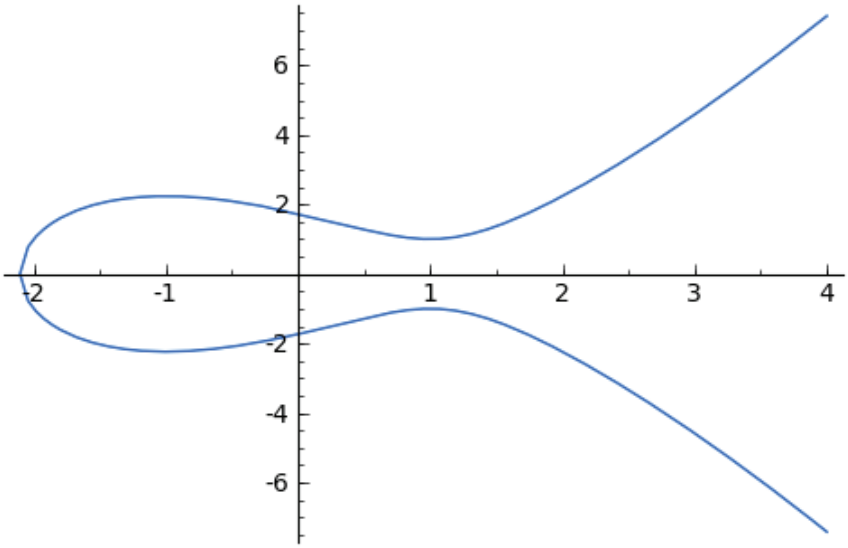


Figure 7.: E_1 : $Y^2 = X^3 - 3X + 3$

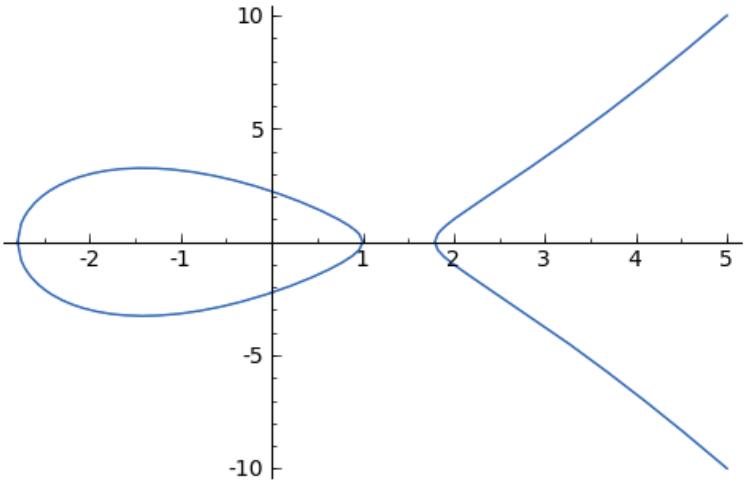


Figure 8.: E_2 : $Y^2 = X^3 - 6X + 5$

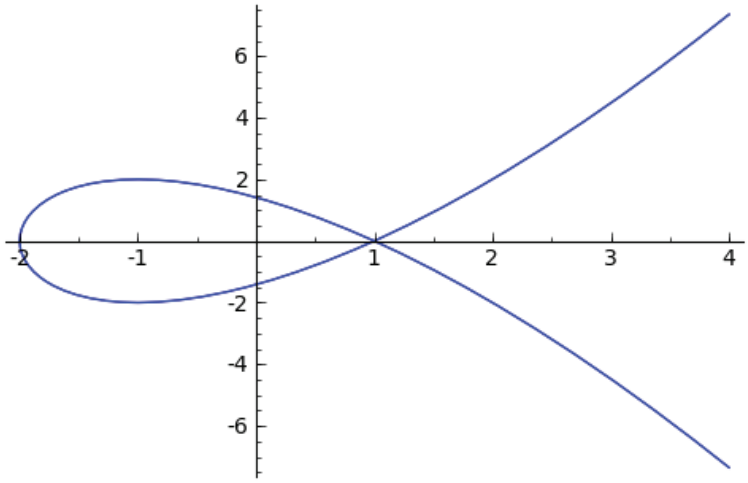


Figure 9.: E_3 : $Y^2 = X^3 - 3X + 2$

The set of points satisfying a Weierstrass equation with non-zero discriminant is called an *Elliptic Curve*. Amazingly, for elliptic curves there is a natural, geometric definition of addition. Let P and Q be two distinct points on an elliptic curve E. Draw the line L passing through P and Q. This line will intersect the curve at one other point, call it $R = (x, y)$. The sum $P \oplus Q$ is defined to be the point $R' = (x, -y)$, i.e. the reflection of R about the x-axis. This process is illustrated in figure 10.

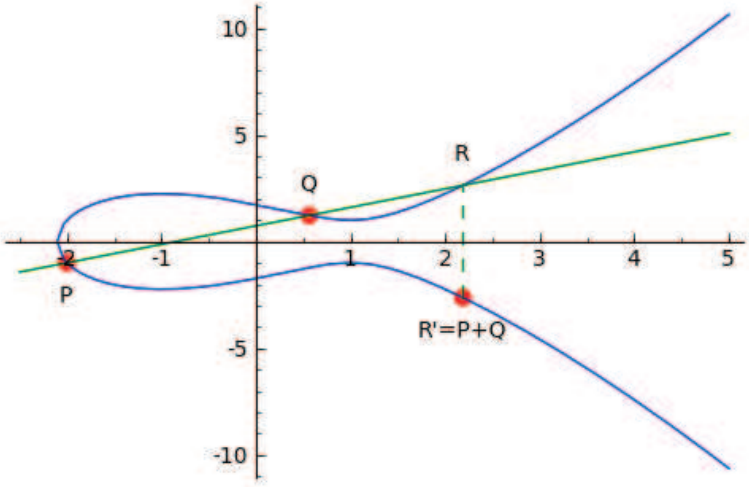


Figure 10.: The addition law on an elliptic curve

Example 7.1.1. Let E be the elliptic curve

$$Y^2 = X^3 - 15X + 18.$$

We find the sum of the points $P = (7, 16)$ and $Q = (1, 2)$ on the elliptic curve as follows:

1.) *Find the equation of the line that intersects both points*

To the find the equation, $y = mx + b$, of the line that intersects both $P = (7, 16)$ and $Q = (1, 2)$, first use point-slope form as follows:

$$y - 16 = (\frac{16 - 2}{7 - 1})(x - 7)$$

$$y = \frac{7}{3}x - \frac{1}{3}$$

2.) *Find the three x-coordinates of the points of intersection with the elliptic curve*

We know from the definition of addition on elliptical curves that this line intersects the curve at three different points. To find these points we substitute $y = \frac{7}{3}x - \frac{1}{3}$ into the equation for the elliptic curve $y^2 = x^3 - 15x + 18$. Then find the roots of the resulting cubic equation $\left(\frac{7}{3}x - \frac{1}{3}\right)^2 = x^3 - 15x + 18$. Simplifying and moving all terms to the right side of the equation gives us the the following equation:

$$0 = 9x^3 - 49x^2 - 121x + 161$$

We know two of the roots are $P_x = 7$ and $Q_x = 1$. Using this knowledge we can factor $(x - 7)(x - 1)(x + \frac{23}{9})$. Therefore the third root of this equation, and the x-coordinate of the unknown point of intersection is $-\frac{23}{9}$.

3.) *Find* $P \oplus Q$

Plugging the x-coordinate into the equation of the line we find the associated y-coordinate to be:

$$y = \frac{7}{3}\left(-\frac{23}{9}\right) - \frac{1}{3} = -\frac{170}{27}$$

Thus $y = \frac{7}{3}x - \frac{1}{3}$ intersects the elliptic curve at $P = (7, 16), Q = (1, 2)$ and $R = (-\frac{23}{9}, -\frac{170}{27})$. Therefore by definition of addition of points on an elliptic curve, $R' = P \oplus Q = \left(-\frac{23}{9}, \frac{170}{27}\right)$.

There are a few subtleties in the previous definition of addition that we need to address. First, how do we add a point P to itself? In other words, how do we double a point? How do we add a point $P = (x, y)$ to the point $-P = (x, -y)$?

- To double a point, we take the line L to be the tangent line at P and proceed as before.

- If we draw the line L going though the point P and its reflection $-P$ across the x-axis, we find L is a vertical line which does not intersect the curve at any other point... at least no other point we see. We define the point at infinity O to be the point where all vertical lines intersect, and set $P \oplus -P = O$.

Example 7.1.2. Using the same point P and elliptic curve E as in Example 7.1.1, we find $P \oplus P$ as follows:

Let $P = (7, 16)$ and let $E : Y^2 = X^3 - 15X + 18$.

To find the slope of the tangent line, we find the derivative of the elliptic curve $Y^2 = X^3 - 15X + 18$. Using implicit differentiation, we have

$$2YY' = 3X^2 - 15,$$

which implies that the slope is

$$Y' = \frac{3X^2 - 15}{2Y}.$$

Next, we use substitution to find the slope of the tangent line at $P = (7, 16)$:

$$Y' = \frac{3(7^2) - 15}{2(16)} = \frac{33}{8}.$$

Now, using our point P and the slope of the tangent line at P, we can find the equation of the tangent line as follows:

$$y - 16 = \frac{33}{8}(x - 7)$$

$$y = \frac{33}{8}x - \frac{33}{8}(7) + 16$$

$$y = \frac{33}{8}x - \frac{103}{8}$$

Then we solve our system of equations: $Y^2 = X^3 - 15X + 18$ and $y = \frac{33}{8}x - \frac{103}{8}$

$$\left(\frac{33}{8}x - \frac{103}{8}\right)^2 = X^3 - 15X + 18$$

$$0 = x^3 - \frac{1089}{64}x^2 + \frac{5838}{32}x - \frac{9457}{64}$$

$$0 = 64x^3 - 1089x^2 + 5838x - 9457$$

$$0 = (x - 7)^2(64x - 193)$$

Thus, the roots are $x = 7$ and $x = \frac{193}{64}$. Using $\frac{193}{64}$ as the x-coordinate of R and the equation of the tangent line $y = \frac{33}{8}x - \frac{103}{8}$, we find the y-coordinate of R to be

$$y = \frac{33}{8}\left(\frac{193}{64}\right) - \frac{103}{8}$$

$$y = -\frac{233}{512}.$$

So we have that $R = \left(\frac{193}{64}, -\frac{233}{512}\right)$ and $P \oplus P = R' = \left(\frac{193}{64}, \frac{233}{512}\right)$

Furthermore, we note that for an arbitrary point $Q \in E$, $Q \oplus O = Q$.

While the geometric definition of addition on elliptic curves is conceptually clear, it is difficult to implement. Fortunately, we can use elementary geometry and differential calculus to find explicit formulas for the addition of elliptic curve points.

Theorem 7.1.3 (Elliptic Curve Addition Algorithm). *Let $E: \quad Y^2 = X^3 + AX + B$ be an elliptic curve, and let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two non-O points on E.*

- *If $P_1 = -P_2$ then $P_1 \oplus P_2 = O$*
- *Otherwise, define λ by*

$$\lambda = \left\{ \begin{array}{ll} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq P_2 \\ \\ \frac{3x_1^2 + A}{2y_1} & \text{if } P_1 = P_2 \end{array} \right\}$$

and let

$$x_3 = \lambda^2 - x_1 - x_2 \qquad \text{and} \qquad y_3 = \lambda(x_1 - x_3) - y_1.$$

Then $P_1 \oplus P_2 = (x_3, y_3)$.

Proof. Let $E: Y^2 = X^3 + AX + B$ be an elliptic curve, and let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two non-O points on E. Consider the trivial case, $P_1 = -P_2$. By definition of the point at infinity, $P_1 \oplus P_2 = O$.

Consider the other case, when $P_1 \neq -P_2$. Notice that λ is the slope of the line between P_1 and P_2 and $y = \lambda x + c$ is the line that intersects P_1 and P_2. Consider $y^2 = (\lambda x + c)^2 = x^3 + Ax + B$. Expanding and rearranging this equation we have

$$0 = x^3 - \lambda^2 x^2 + (A - 2c\lambda)y + B - c^2.$$

Let $m = (A - 2c\lambda)$ and $n = B - c^2$. Thus the cubic equation can be written

$$0 = x^3 - \lambda^2 x^2 + mx + n.$$

We know x_1, x_2, x_3 are three solutions of the cubic, $0 = x^3 - \lambda^2 x^2 + mx + n$. So $(x - x_1)(x - x_2)(x - x_3) = x^3 - \lambda^2 x^2 + mx + n$. After expanding this we see that $(x_1 + x_2 + x_3) = \lambda^2$ since they are equal coefficients. Thus, solving for x_3, we have $x_3 = \lambda^2 - x_1 - x_2$.

Now using point slope form to rewrite the equation of a line as $y - y_1 = \lambda(x - x_1)$. Plugging in x_3, we have $y = \lambda(x_3 - x_1) + y_1$. We then reflect y over the x-axis to obtain $y_3 = -y = \lambda(x_1 - x_3) - y_1$.

To understand the geometric meaning of λ for the $P_1 = P_2$ case we need to differentiate the elliptic curve equation $Y^2 = X^3 + AX + B$. Doing so yeilds the equation $(2Y)dY = (3X^2 + A)dX$ which implies that $\frac{dY}{dX} = \frac{3X^2+A}{2Y}$. Notice now that when $P_1 = P_2$, λ is defined as the slope of the line tanget to the curve at the point $P_1 = P_2 = (x_1, y_2)$. This means that geometrically, the same method of factoring used in the previous applies to this case and we come to the same conclusion.

Thus, $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda(x_1 - x_3) - y_1$ in all cases, so $P_1 \oplus P_2 = (x_3, y_3)$. □

Theorem 7.1.4. *The set of points on an elliptic curve, together with the point at infinity, forms an Abelian Group under addition.*

7.1.1 *Elliptic Curves over Finite Fields*

Underlying our previous discussion of elliptic curves was the assumption that we were working in $\mathbb{R}$. To apply the theory of elliptic curves to cryptography, we need to work in the realm of finite fields. For more information on finite fields, see Section 1.2.4.

Definition 7.1.5 (Elliptic Curve E over $\mathbb{F}_p$)**.** The elliptic curve $E(\mathbb{F}_p)$ is the set of points in $\mathbb{F}_p \times \mathbb{F}_p$ satisfying the equation

$$E: \quad Y^2 = X^3 + AX + B,$$

where A and B are elements of $\mathbb{F}_p$ satisfying $4A^3 + 27B^2 \neq 0$; along with the point at infinity, O. In other words,

$$E(\mathbb{F}_p) = \big\{(x, y) : x, y \in \mathbb{F}_p \text{ satisfying Wierstrass equation } E \text{ with } \Delta_E \neq 0\big\} \cup \{O\}.$$

For ease, we will limit our discussion to finite fields $\mathbb{F}_p$ with $p > 3$ (for example see figure 11). While fields of characteristic 2 and 3 are important in cryptography, they are more complicated to study.

Example 7.1.6. Consider the elliptic curve $E : Y^2 = X^3 + 3X + 8$ over $\mathbb{F}_{13}$. We can find the points of $E(\mathbb{F}_{13})$.

Let $E(\mathbb{F}_{13}) = \big\{(x, y) : x, y \in \mathbb{F}_{13}, \text{ satisfying } Y^2 = X^3 + 3X + 8 \pmod{13}\big\} \cup \{O\}$.

In the following two tables, we find the points of $E(\mathbb{F}_{13})$. On the left, we calculate all of the squares in $\mathbb{F}_{13}$. On the right we plug in all of the possible x coordinates into the elliptic curve equation. If the resulting y^2 value is a square in $\mathbb{F}_{13}$, then the (x, y) pair is a point on the curve.

y	y^2
0	0
1	1
2	4
3	9
4	3
5	12
6	10
7	10
8	12
9	3
10	9
11	4
12	1

x	y^2
0	8
1	12
2	9
3	5
4	6
5	5
6	8
7	8
8	11
9	10
10	11
11	7
12	4

Comparing the two y^2 columns, we see that the points 5, 6, 7, 8, and 11 are not squares in $\mathbb{F}_{13}$, which means there are no points on the curve with x coordinates of 0, 3, 4, 5, 6, 7, 8, 10, or 11. The remaining points are $(1, \sqrt{12})$, $(2, \sqrt{9})$, $(0, \sqrt{10})$, and $(12, \sqrt{4})$. Using the table on the left, we find that the full set of points is

$$E(\mathbb{F}_{13}) = \{(1,5), (1,8), (2,3), (2,10), (9,6), (9,7), (12,2), (12,11)\} \cup \{O\}$$

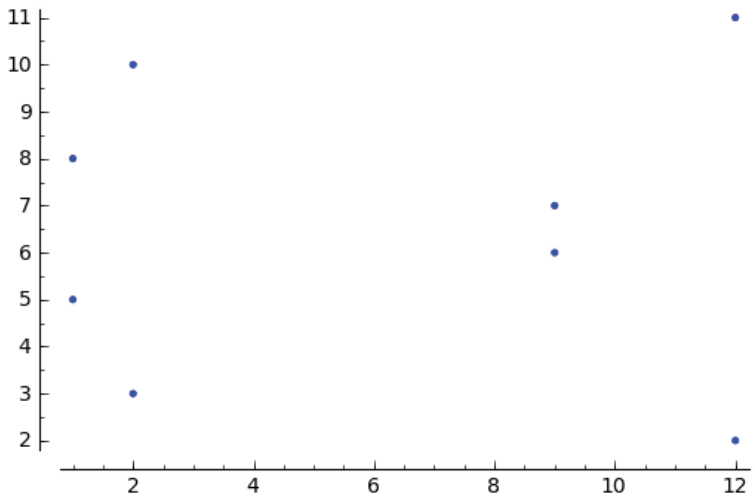


Figure 11.: $E: \quad Y^2 = X^3 + 3X + 8 \qquad$ over $\mathbb{F}_{13}$

Theorem 7.1.7. *Let E be an elliptic curve over $\mathbb{F}_p$, and let P and Q be points in $E(\mathbb{F}_p)$. The addition algorithm defined in theorem 7.1.3 applied P and Q (with all arithmetic done modulo p) returns a point in $E(\mathbb{F}_p)$.*

Proof. Let E be an elliptic curve over $\mathbb{F}_p$, and let P and Q be points in $E(\mathbb{F}_p)$ where $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ and $P \oplus Q = (x_3, y_3)$. By construction of the algorithm defined in Theorem 7.1.3, $P \oplus Q$ will output a point on E. Therefore it is sufficient to show that, if P and Q are points in $\mathbb{F}_p$, using the algoritm with all arithmetic performed modulo p will result in an element of $\mathbb{F}_p$. In order to calculate λ modulo p, we must use the inverses of $(x_2 - x_1)$ and $(2y_1)$ in their appropriate conditions instead of dividing by those terms. Since multiplication and addition are closed in F_p, so long as $(x_2 - x_1)^{-1}$ and $(2y_1)^{-1}$ exist, $P \oplus Q$ will return a point in $\mathbb{F}_p$.

Consider P, Q are not inverses, and consider $P = Q$. Then $P \oplus Q = P \oplus P = (x_3, y_3)$ where, $x_3 = \left((3x_1^2 + A)(2y_1)^{-1}\right)^2 - 2x_1$ and $y_3 = \left((3x_1^2 + A)(2y_1)^{-1}\right)(x_1 - x_3) - y_1$ with all calculations being done modulo p. By way of contradiction assume that $2y_1$ does not have a multiplicative inverse in F_p. This would mean that $2y_1 \equiv 0 \pmod{p}$. Since there are no zero-divisors in F_p and 2 is relatively prime to p, this implies that $y_1 \equiv 0 \pmod{p}$. Since 0 is its own additive inverse, if $y_1 \equiv 0 \pmod{p}$ then P is its own inverse $P = -P$ which contradicts the assumption we made for this case. Therefore, (x_3, y_3) is an element of $E(\mathbb{F}_p)$ when P and Q are equal and not inverses.

Now consider $P \neq -Q$ where P and Q are still not inverses, then $P \oplus Q = (x_3, y_3)$ where $x_3 = \left((y_1 - y_2)(x_1 - x_2)^{-1}\right)^2 - x_1 - x_2$ and $y_3 = \left((y_1 - y_2)(x_1 - x_2)^{-1}\right)^2 (x_1 - x_3) - y_1$ with all calculations being done modulo p. By definition of a field, every element of $\mathbb{F}_p$ has an inverse except for the additive identity, zero. If $x_1 - x_2 \equiv 0$ then $x_1 \equiv x_2 \pmod{p}$. Since we already specified that P and Q are distinct points that are not inverses of each other, we know that $x_1 \neq x_2$. Therefore $(x_1 - x_2)^{-1}$ exists and (x_3, y_3) is an element of $E(\mathbb{F}_p)$ when P and Q are not equal and not inverses.

When P and Q are inverses, by definition, $P \oplus Q = O$ which is an element of $E(\mathbb{F}_p)$. Therefore in all cases, $E(\mathbb{F}_p)$ is closed under the $\oplus$ operation. □

Theorem 7.1.8. *The set $E(\mathbb{F}_p)$ with addition defined by Theorem 7.1.3, and computed modulo p, forms a finite Abelian group.*

Proof. 1.) *Existence of an Identity*

By definition, the element O acts as the identity for $E(\mathbb{F}_p)$ since, for any point P, $P \oplus O = P$

2.) *Existence of Inverses*

If P is the point (x, y) we define its inverse, $-P$ to be the point $(x, -y)$. By the definition of the point at infinity, $P \oplus -P = O$ for any point in $E(\mathbb{F}_p)$.

3.) *Closed under* $\oplus$

See Theorem 7.1.7

4.) *Associative and Communative*

As discussed in the previous two theorems, since the $\oplus$ operation is actually just a combination of several multiplicative and additive operations in the field F_p it inherits the associative and communicative properties that these operations had in F_p.

Therefore $E(\mathbb{F}_p)$ is an abelian group under the $\oplus$ operation. □

Example 7.1.9. Consider once again

$$E: \quad Y^2 = X^3 + 3X + 8 \qquad \text{over} \quad \mathbb{F}_{13},$$

and points $P = (9,7)$ and $Q = (1,8)$ in $E(\mathbb{F}_{13})$. We find $P \oplus Q$.

Since $P \neq Q$, as stated in Theorem 7.1.3

$$\lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} \pmod{13}$$

So

$$\lambda \equiv \frac{8-7}{1-9} \equiv 1(-8)^{-1} \equiv 1(5)^{-1} \equiv 5^{-1} \equiv 8 \pmod{13}$$

Now substitute λ, x_1 and x_2 in the equation

$$x_3 \equiv \lambda^2 - x_1 - x_2$$

$$(8)^2 - 9 - 1 \equiv 54 \equiv 2 \pmod{13}$$

and when we substitute λ, x_1 and x_2 in the equation

$$y_3 \equiv \lambda(x_1 - x_3) - y_1$$

$$8(9-2) - 7 \equiv 49 \equiv 10 \pmod{13}.$$

Thus $x_3 \equiv 2 \pmod{13}$ and $y_3 \equiv 10 \pmod{13}$ so $P \oplus Q \equiv (2,10)$.

We also find $2P = P \oplus P$.

Since $P = P$

$$\lambda \equiv \frac{3x_1^2 + A}{2y_1} \pmod{13}$$

So

$$\lambda \equiv \frac{3(9)^2 + 3}{2(7)} \equiv 246(14)^{-1} \equiv 12(1)^{-1} \equiv 12 \pmod{13}$$

$$x_3 \equiv \lambda - x_1 - x_1 \equiv (12)^2 - 9 - 9 \equiv 126 \equiv 9 \pmod{13}$$

and

$$y_3 \equiv \lambda(x_1 - x_3) - y_1 \equiv 12(9-9) - 7 \equiv -7 \equiv 6 \pmod{13}$$

Therefore $2P = P \oplus P = (9, 6)$.

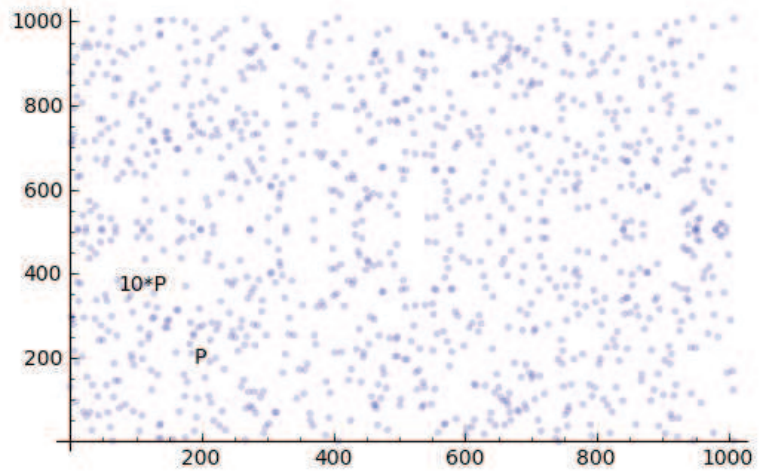


Figure 12.: $E : Y^2 = X^3 + 3X + 8$ over $\mathbb{F}_{1009}$, and two point P and $10P$ in E

Figures 11 and 12 show the elliptic curve

$$E: \quad Y^2 = X^3 - 6X + 5$$

over two finite fields. As these figures demonstrate, any geometric intuition on the relationship between P and nP for some point $P \in E$ is obfuscated by the finite field structure. This illustrates how difficult the discrete logarithm problem is for elliptic curves over finite fields.

7.2 ELLIPTIC CURVE CRYPTOSYSTEMS

Before examining the elliptic curve analogues to Diffie-Hellman key exchange and El-Gamal public key encryption (described in Sections 3.2.1 and 3.2.2), we must define the underlying “hard" mathematical problem upon which these elliptic curve cryptosystems are based.

7.2.1 *Elliptic Curve Discrete Logarithm Problem*

Definition 7.2.1 (Elliptic Curve Discrete Logarithm Problem)**.** Let E be an elliptic curve over the finite field $\mathbb{F}_p$ and let P and Q be points in $E(\mathbb{F}_p)$. The *Elliptic Curve Discrete Logarithm Problem* (ECDLP) is the problem of finding an integer n such that $Q = nP$.

Remark 7.2.2. The elliptic curve discrete log problem is not well-posed for all values of Q and P in $E(\mathbb{F}_p)$ if P is not a generator of $E(\mathbb{F}_p)$. However, for cryptographic purposes, we can assume that it is. In practice, Alice will select a public value P and a private integer n. She will then compute and publish the value $Q = nP$. Thus, the discrete log problem of finding n from Q will have at least one possible solution.

Even for values of P and Q which make the ECDLP well-posed, there are infinitely many possible answers since if $Q = nP$, notice that $Q = nP = (n + m|P|)P$ for some integer m.

Definition 7.2.3 (Elliptic Discrete Logarithm Function)**.** Given an elliptic curve E and an element $P \in E(\mathbb{F}_p)$ of order s, let G be the subgroup of $E(\mathbb{F}_p)$ generated by P, i.e. $G = \{Q \in E(\mathbb{F}_p) | Q = nP \text{ for some } n \in \mathbb{Z}\}$. The *Elliptic Discrete Logarithm Function* is the function

$$\log_P : G \to \mathbb{Z}/s\mathbb{Z}.$$

Theorem 7.2.4. *The elliptic discrete logarithm satisfies*

$$\log_P(Q_1 + Q_2) = \log_P(Q_1) + \log_P(Q_2) \qquad \forall Q_1, Q_2 \in E(\mathbb{F}_P).$$

Remark 7.2.5. In order to implement elliptic curve cryptosystems, we need an efficient way of calculating nP from the known values of $n \in \mathbb{Z}$ and $P \in E(\mathbb{F}_p)$. Analogous to the Fast Powering Algorithm described in Appendix 1.3, the Double-and-Add Algorithm provides one efficient method of calculating nP for large values of n. The main idea is to write n in *ternary form*, that is as a sum or difference of powers of 2.

$$n = n_0 + n_1 \cdot 2 + n_2 \cdot 2^2 + \ldots + n_r \cdot 2^r \qquad \text{where } n_0, n_1, \ldots, n_r \in \{-1, 0, 1\}$$

To find nP, you compute the quantities

$$Q_0 = P, \qquad Q_1 = 2Q_0, \qquad Q_2 = 2Q_1, \qquad \ldots \qquad , Q_r = 2Q_{r-1}.$$

The desired value nP is then

$$nP = n_0Q_0 + n_1Q_1 + \ldots + n_rQ_r.$$

Example 7.2.6. Given a point $P \in E(\mathbb{F}_p)$, let $n = 349$ Thus

$$n = 349 = n_0 + n_1 \cdot 2 + n_2 \cdot 2^2 + \ldots + n_r \cdot 2^r$$

for some integer r. Since 2^8 is the largest integer that is smaller than 349 let $r = 8$. So

$$349 = n_0 + n_1(2^1) + n_2(2^2) + n_3(2^3) + n_4(2^4) + n_5(2^5) + n_6(2^6) + n_7(2^7) + n_8(2^8)$$

and we have that

$$349 = -1 + (-1)(2^2) + 0(2^3) + 0(2^4) + 1(2^5) + 1(2^6) + 0(2^7) + 1(2^8)$$

is one of the options when choosing $n_0, n_1, \ldots, n_r \in \{-1, 0, 1\}$. Therefore when we consider $349P$ we find

$$349P = -1(Q_0) - 1(Q_1) + 1(Q_5) + 1(Q_6) + 1(Q_8)$$

Ternary form enables us to complete this example in a variety of ways. This is just one option.

7.2.2 *Elliptic Curve Diffie-Hellman Key Exchange*

The following protocol allows Alice and Bob to establish a shared, secure, symmetric key while conducting all communication over insecure lines.

Step 1. Alice and Bob agree upon an elliptic curve E and finite field $\mathbb{F}_p$. They also choose a particular value $P \in E(\mathbb{F}_p)$. Thus P and $E(\mathbb{F}_p)$ are public information.

Step 2. Alice picks a secret integer a and computes $Q_A = aP$. Similarly, Bob picks a secret integer b and computes $Q_B = bP$. The values a and b are known as the private keys and are not public knowledge. Alice and Bob then transmit Q_A and Q_B.

Step 3. Upon receiving Q_B, Alice uses her private key to calculate $K = aQ_B$. Similarly, upon receiving Q_A, Bob calculates $K' = bQ_A$. By the commutative and associative properties of elliptic curve addition,

$$K = aQ_B = abP = aPb = Q_Ab = bQ_A = K'.$$

Thus, Alice and Bob share the secret key K.

One way for an adversary Eve to discover Alice and Bob's secret key is to solve the ECDLP. Since Q_A, Q_B, P, and $E(\mathbb{F}_p)$ are all public knowledge, if Eve can solve the ECDLP, she can find a from Q_A and use it to calculate the shared secret key $K = aQ_B$. Of course there might be some other way for Eve to discover K from Q_A, Q_B, P, and $E(\mathbb{F}_p)$. The precise problem that Eve needs to solve is the elliptic curve Diffie-Hellman problem.

Definition 7.2.7 (Elliptic Curve Diffie-Hellman Problem)**.** Let $E(\mathbb{F}_p)$ be an elliptic curve over a finite field and let $P \in E(\mathbb{F}_p)$. The *Elliptic Curve Diffie-Hellman Problem* (ECDHP) is the problem of computing the value of abP from the known values of aP and bP.

Remark 7.2.8. The elliptic Diffie-Hellman key exchange protocol described above requires Alice and Bob to transmit points on an elliptic curve. Such points consist of two pieces of information, the x and y coordinate. In practice, Alice and Bob need only transmit the x coordinate. Suppose Alice chooses the private key a and computes $Q_A = aP = (x_A, y_A)$. There are at most two points in $E(\mathbb{F}_p)$ having x-coordinate x_A, namely Q_A and $-Q_A$. Once Alice transmits x_A, Bob can use the definition of the elliptic curve E to find $|y_A|$. Thus, Bob will know $\pm Q_A$. From this, he can calculate $K' = \pm bQ_A$. Similarly, Alice can find $|y_B|$ from x_B, and calculate $K = \pm aQ_B$. The values $K = (x_K, y_K)$ and $K' = (x_{K'}, y_{K'})$ are either equal or $K = -K'$. Either way, the x-coordinates x_K and $x_{K'}$ are equal. Alice and Bob can then use this shared secret value x_K as their symmetric key.

7.2.3 *Elliptic ElGamal Public Key Cryptosystem*

The ElGamal public key encryption scheme described in Section 3.2.2 can be easily adapted to the elliptic curve setting.

In the *Elliptic ElGamal Public Key Cryptosystem*, Alice establishes her public/private key pair as follows. First she selects a large prime p, an elliptic curve E, and a point $P \in E(\mathbb{F}_p)$. She then selects an integer a as her private key and calculates

$$Q_A = aP$$

This is her public key which she publishes along with P and $E(\mathbb{F}_p)$.

- Encryption: Suppose Bob wants to encrypt a message $m \in E(\mathbb{F}_p)$. Bob chooses an integer k to be his ephemeral key and calculates

  $$c_1 = kP \qquad \text{and} \qquad c_2 = m \oplus kQ_A$$

  Bob's ciphertext is the tuple (c_1, c_2).

- Decryption: Upon receiving (c_1, c_2), Alice computes $c_2 \oplus -ac_1$.

$$\begin{aligned} c_2 \oplus -ac_1 &= m \oplus kQ_A \oplus -ac_1 \\ &= m \oplus kQ_A \oplus -akP \\ &= m \oplus kQ_A \oplus -kaP \\ &= m \oplus kaP \oplus -kaP \\ &= m \oplus 0 \\ &= m \end{aligned}$$

  to recover the plaintext m.

Remark 7.2.9. There are a few practical difficulties to overcome when implementing the elliptic ELGamal cryptosystem. First, there is no obvious way to attach a plaintext message to points in $E(\mathbb{F}_p)$. Secondly, because the ciphertext consists of two elliptic curve points, which themselves are tuples in $\mathbb{F}_p \times \mathbb{F}_p$, for large values of p, storing and transmitting the ciphertext can be costly. This cannot be mitigated by only transmitting x-coordinates since the y-coordinates are required to decrypt the sent message. However, if we transmitted the x-coordinate with one additional bit of information (a one or a zero) that would indicate if the y-coordinate was negative or positive, it could work. This could be done without much extra storage space or hassle.

7.3 LENSTRA'S METHOD

Recall Pollards $p-1$ factorization algorithm described in Section 4.2.3. Factors of $N = pq$ are found by searching for powers of a such that

$$a^L \equiv 1 \pmod{p} \qquad \text{and} \qquad a^L \not\equiv 1 \pmod{q},$$

for some random value of a. By Fermat's Little Theorem, this is likely to work when $p-1 | L$ but $q-1 \nmid L$. The determining factor in our ability to find such an L with a reasonable amount of effort is whether or not $p-1$ and $q-1$ are smooth. The quantity $p-1$ is important in the algorithm because it is the order of the multiplicative group $\mathbb{F}_p^*$. Thus, for all $a \not\equiv 0 \pmod{p}$, $a^{p-1} \equiv 1 \pmod{p}$.

As we saw in Section 7.1, the points on an elliptic curve $E(\mathbb{F}_p)$, under elliptic curve point addition, form an Abelian group. Hendrik Lenstra realized that this elliptic curve group is analogous to the multiplicative group of units $\mathbb{F}_p^*$. Using this idea, he developed a factorization method analogous to Pollard's $p-1$ algorithm, but set in the realm of elliptic curves.

7.3.1 *Lenstra's Algorithm*

Consider the elliptic curve $E(\mathbb{Z}/N\mathbb{Z})$ for some composite number N, i.e.

$$E: \quad Y^2 = X^3 + AX + B$$

where A and B are in $\mathbb{Z}/N\mathbb{Z}$, and $\triangle_E \not\equiv 0 \pmod{N}$. Let $P = (a, b)$ be a point on E, by which we mean

$$b^2 = a^3 + Aa + B \pmod{N}.$$

Using the elliptic curve addition algorithm defined in Theorem 7.1.3, we can attempt to calculate multiples of P. Since $E(\mathbb{Z}/N\mathbb{Z})$ does not form an additive group, nP is not necessarily an element of $E(\mathbb{Z}/N\mathbb{Z})$ for all values of $n \in \mathbb{Z}$.

Remark 7.3.1. The algorithm of Theorem 7.1.3 can go wrong if either $(x_2 - x_1)^{-1}$ or $(2y_1)^{-1}$ do not exist in $\mathbb{Z}/N\mathbb{Z}$. This is because Theorem 7.1.3 requires us to calculate $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ or $\lambda = \frac{3x_1^2 + A}{2y_1}$. Since N is composite, there are some elements in $\mathbb{Z}/N\mathbb{Z}$ that are not relatively prime to N and, therefore, do not have inverses. In other words, we cannot calculate $P_1 \oplus P_2 = (x_3, y_3)$.

Definition 7.3.2 (Lenstra's Factorization Algorithm)**.** Let N be a composite number.

- Take an elliptic curve
$$E(\mathbb{Z}/N\mathbb{Z}): \quad Y^2 = X^3 + AX + B,$$
and a point $P = (a, b) \in E(\mathbb{Z}/N\mathbb{Z})$
- Calculate multiples of P,
$$2!P, \quad 3!P, \quad 4!P, \quad \ldots$$
using the algorithm defined in Theorem 7.1.3 with all calculations done modulo N
- If at any point, the elliptic curve point addition algorithm fails, then we have found a value d such that $\gcd(d, N) \neq 1$.
  - If $\gcd(d, N) < N$, then we have found a factor of N.
  - If $\gcd(d, N) = N$, we have been unlucky and should start over with a new point P'.

Remark 7.3.3. We have glossed over one detail in our definition of Lenstra's algorithm. How do we find an elliptic curve $E(\mathbb{Z}/N\mathbb{Z})$ and an initial point P on the curve? The obvious method is to fix an equation for the curve E, plug in values for X and see if the quantity $X^3 + AX + B$ is a square modulo N. Unfortunately, as we have already seen, this is difficult to do unless we know the factorization of N. To get around this problem, we start by selecting a random point $P = (a, b)$ and a random value $A \in \mathbb{Z}/N\mathbb{Z}$, and setting
$$B \equiv b^2 - a^3 - A \cdot a \pmod{N}.$$
Thus, we are guaranteed the point P lies on the curve $E: \quad Y^2 = X^3 + AX + B$.

Lenstra's method relies on the fact that in calculating multiples of P, we will find a value d such that $\gcd(d, N) \neq 1$. How is this better than randomly guessing values v_i less than N and checking the $\gcd(v_i, N)$? The answer lies in the following two theorems.

Theorem 7.3.4. *Let N be the product of distinct primes, and P a point on $E(\mathbb{Z}/N\mathbb{Z})$. When using the elliptic curve addition algorithm defined in Theorem 7.1.3 to calculate nP for some $n \in \mathbb{Z}$, if the algorithm does not fail, then it returns a point $nP \in E(\mathbb{Z}/N\mathbb{Z})$. If the algorithm does fail, then $nP = O \in E(\mathbb{F}_q)$ or $nP = 2P' \in E(\mathbb{F}_q)$ where q is a prime divisor of N and P' is some point in $E(\mathbb{F}_q)$*

Proof. Suppose the algorithm fails. Consider nP. Note that $nP = P \oplus (n-1)P = P_1 \oplus P_2$ and suppose that P_1 and P_2 are distinct. Since the algorithm fails, $(x_2 - x_1)^{-1}$ doesn't exist in $\mathbb{Z}/N\mathbb{Z}$ and so $\gcd(x_2 - x_1, N) \neq 1$. Therefore there exists a prime q such that $q|(x_2 - x_1)$ and $q|N$. This implies that $x_2 \equiv x_1 \pmod{q}$ by definition of congruence. When we consider $P_1' = (x_1, \pm y)$ and $P_2' = (x_2, \pm y) \pmod{q}$ we find we have 2 cases.

Case 1:

If $P_1' \equiv P_2'$ then

$$P_1' \equiv P_2' \equiv P' \to P_1' \oplus P_2' = P' \oplus P' = 2P' \pmod{q}$$

Case 2:

If $P_1' \not\equiv P_2' \pmod{q}$

$$P_1' \equiv -P_2' \pmod{q} \to P_1' \oplus P_2' = P_1' \oplus -P_1' = O \pmod{q}$$

Now, assume that P_1 and P_2 are not distinct, or $P_1 = P_2$. This would imply that $x_1 \equiv x_2 \pmod{N}$. If q were a prime that divided N, then that would imply that $x_1 \equiv x_2 \pmod{q}$ as well since $q|N$. Now, the same chain of logic as the previous case can be applied to conclude that $nP = O \in E(\mathbb{F}_q)$ or $nP = 2P' \in E(\mathbb{F}_q)$.

Therefore if the algorithm fails, in all cases $nP = O \in E(\mathbb{F}_q)$ or $nP = 2P' \in E(\mathbb{F}_q)$.

If the algorithm does not fail, by Theorem 7.1.3 the sum will be on the curve as a point of the form nP.

□

Remark 7.3.5. What has happened if our algorithm fails and our found value d has the property $\gcd(d, N) = N$? If our algorithm fails, we have found a value $d = x_2 - x_1$ or $d = 2y_1$ where d does not have an inverse in $\mathbb{Z}/N\mathbb{Z}$ and implies $\gcd(d, N) = N$. This implies that d must have been a multiple of N which is to say that $d \equiv 0 \pmod{N}$. If the points were distinct this would imply that $x_2 \equiv x_1 \pmod{N}$ or that we happened to add together two points that were inverses of each other. If the points were the same this would imply that $2_y \equiv 0 \pmod{N}$ or that the tangent line had an undefined slope at that point. This is the same as saying that we happened to add a point to itself that was its own inverse. Either way, we have been quite unlucky.

Theorem 7.3.6. *[Hasse] Let E be an elliptic curve over $\mathbb{F}_p$. The number of points in the elliptic curve group $E(\mathbb{F}_p)$ is given by*

$$\#E(\mathbb{F}_p) = p + 1 - t_p \qquad \text{where } |t_p| \leq 2\sqrt{p}.$$

Since the order of the point P in $E(\mathbb{F}_q)$ must divide $\#E(\mathbb{F}_q)$, and $\#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}$, if N has a relatively small prime factor q, then we are likely to stumble across a value

$n!$ which is multiple of the order of P in $E(\mathbb{F}_q)$ (and thus a value for which the elliptic curve addition algorithm fails) more quickly than we would randomly chose a value v_i such that $\gcd(v_i, N) \neq 1$. Thus, the efficiency of Lenstra's method depends on the size of the smallest prime factor of N, not on the size of N itself. If q is the smallest prime factor, then the average running time of the algorithm is approximately

$$\mathcal{O}\left(e^{\sqrt{2(\log q)(\log\log q)}}\right) \text{ steps.}$$

Example 7.3.7. We use Lenstra's method to factor 4453.

Consider the elliptic curve

$$E(\mathbb{Z}/4453\mathbb{Z}): \quad Y^2 = X^3 + 10X - 2,$$

and the point $P = (1, 3)$.

To use Lenstra's method, the first step is to calculate multiples of $P(2!P, 3!P, ...)$ until the addition algorithm fails. So, first, we calculate $2P$:

$$P \oplus P = (1,3) \oplus (1,3)$$

$$\lambda = \frac{3x_1^2 + A}{2y_1} = \frac{3(1^2) + 10}{2(3)} = 13(6^{-1}) \equiv 3713 \pmod{4453}$$

$$x_3 = \lambda^2 - x_1 - x_2 = 3713^2 - 1 - 1 = 4332 \mod 4453$$
$$y_3 = \lambda(x_1 - x_3) - y_1 = 3713 * (1 - 4332) - 3 \equiv 3230 \pmod{4453}$$

Therefore $2P = (4332, 3230)$ and the algorithm succeeded. Next, we need to calculate $3!P$ or $6P$. We begin by adding $2P$ to itself to get $4P$. In this case:

$$\lambda = \frac{3x_1^2 + A}{2y_1} = \frac{3(4332^2) + 10}{2(3230)} = 3856(6460^{-1}) \equiv 3856 \pmod{4453}$$

$$x_3 = 2143^2 - 4322 - 4322 \equiv 1648 \pmod{4453}$$

$$y_3 = 2143 * (4322 - 1648) - 3230 \equiv 4212 \pmod{4453}$$

And therefore, $4P = (1648, 4212)$. To get $6P$ we now need to add $2P$ and $4P$. When we try to do this, we find that

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3230 - 4212}{4332 - 1648} \equiv 3471(2684^{-1}) \pmod{4453}.$$

However, the algorithms fails because 2684 does not have a multiplicative inverse (mod 4453). Now, to factor 4453, we can find $\gcd(2684, 4453)$ and see if it is a prime. Since $\gcd(2684, 4453) = 61$, we factor 4453 as $4453 = 61 * 73$.

8

CHOOSING A SECURE ELLIPTIC CURVE (BY DUSAN PEKICH)

The previous section explained how we can create crypotsystems with the elliptic curves descried by Weierstrass Equations. However, in practice, how can we actually choose a finite field and elliptic curve to create our cryptosystem with? Are all finite fields and elliptic curves created equally, or are there some elliptic curves that create cryptosystems with back doors and vulnerabilities that could potentially be taken advantage of by an adversary? This chapter starts by going over some common attacks that can be performed on elliptic curve crypotsystems and what elliptic curves, specifically, are vulnerable to these attacks. Then, we expand our elliptic curve knowledge to be able to form elliptic curve cryptosystems with elliptic curves over finite fields of characteristic of two, and finite fields with a prime power order. Finally, we use this extended knowledge of elliptic curves to develop a method to be able to reliably choose a secure field and elliptic curve that can stand up to the attacks explained at the beginning of the chapter. All of this requires a deeper understanding of how, exactly, Weierstrass Equations are related to Elliptic Curves.

8.1 ATTACKS ON ELLIPTIC CURVE CRYPTOSYSTEMS

8.1.1 *Pollard-ρ Attack and Polhlig-Hellman Attack*

As mentioned in Section 3.4.3, the Pollard-ρ algorithm is able to solve a DLP relatively easily if the size of a group is smooth. Additionally, the Pohlig-Hellman Algorithm described in Chapter 3 also gives us an easy way of solving a DLP if the order of a group is smooth. Recall if a number is B-smooth that means that for all primes p_i in its prime factorization, $p_i < B$. These algorithms can also be used to solve the additive DLP in $E(F_q)$ [25], [26]. Even though adding points on an elliptic curve takes more computer power and is slower than simply multiplying integers, choosing an elliptic curve where $\#E(F_q)$ is smooth or only the product of small primes would not produce a secure cryptosystem. This means, to avoid building an elliptic curve for which the DLP is easy, we want to make sure that the size of our additive group on the elliptic curve,

$\#E(F_q)$ is divisible by a sufficiently large prime p. In practice, it suffice to show that $\#E(F_q)$ is divisible by some prime $p > 2^{160}$ [25], [27]

8.1.2 *Semaev-Smart-Satoh-Araki Attack*

Also called an Anomalous Attack, the Semaev-Smart-Atoh-Araki Attack utilizes a mathematical concept known as p-adic numbers.

Definition 8.1.1 (P-adic numbers)**.** If $a \in \mathbb{Q}$ is a rational number and p is a prime integer we can write a as $a = p^r \frac{m}{n}$ where $r \in \mathbb{N}$ and $m, n \in \mathbb{Z}$ and are not multiples of p. We define then define the p-adic order of a as $\text{ord}_p(a) = r$ and the *norm* of a non-zero a as $|a|_p = p^{-r}$. The set of p-adic numbers $\mathbb{Q}_p$ is the completion of the rational numbers with respect to the metric d_p defined as $d_p(a, b) = |a - b|_p$. The set of p-adic numbers a such that $\text{ord}(a) \geq 0$ form the set of p-adic integers known as $\mathbb{Z}_p$.

Definition 8.1.2 (P-adic expansion)**.** Every element of $\mathbb{Q}_p$ can be expressed uniquely in the form of an infinite series $c_{-n}p^{-n} + ... + c_0 + c_1 p + ... + c_m p^m + ...$ where the c_i's are integers between zero and $p - 1$. This comes directly from the completion of the rational numbers using the metric.

Now, we can start to look at elliptic curves over the p-adic numbers. If we have an elliptic curve $E(Q_p)$ over the p-adic numbers for some p, where each of the coefficients of its Weierstrass Equation is an element of Q_p. How can we relate these curves to the elliptic curves over the integers that we are used to?

Definition 8.1.3 (Lifting an Elliptic Curve Modulo-p)**.** Consider an elliptic curve over a field $\bar{E}(\mathbb{F}_p)$. We find the *lift* of a point $\bar{P} \in \bar{E}$ by first considering the curve $E(\mathbb{Q}_p)$ where the Weierstrass equation describing E is the same as the as the one describing $\bar{E}$. This can be done since elements of $\mathbb{F}_p$ form a subgroup of $\mathbb{Q}_p$ (for all $a \in \mathbb{F}_p$, $a = p^0 \frac{a}{1}$). Next, we consider the point $P \in E$ where the x-coordinate of P is the same as the x-coordinate of $\bar{P}$. The y-coordinate of P is the p-adic expansion that satisfies the weierstrass equation. Note that this is not necessarily the y-coordinate of $\bar{P}$ since all addition and multiplication is no longer done modulo p. We call the point $P \in E(\mathbb{Q}_p)$ a *lift* of the point $\bar{P} \in \bar{E}(\mathbb{F}_p)$ and $E(Q_p)$ is the *lift* of the elliptic curve $\bar{E}(\mathbb{F}_p)$.

In the same way that multiple points on an elliptic curve over the integers can have the same x-coordinate, many points on an elliptic curve over the p-adic numbers can have the same x-coordinate, meaning that every point on an elliptic curve over a finite field can have many possible lifts in a curve over the p-adic numbers. This leads us to the following definition.

Definition 8.1.4 ($E_1(\mathbb{Q}_p)$ and $E_2(\mathbb{Q}_p)$)**.** If $E(Q_p)$ is the lift of the elliptic curve $\bar{E}(\mathbb{F}_p)$ We define $E_1(\mathbb{Q}_p)$ as the set of points on E that the point at infinity on $\bar{E}$ can lift to. And we define $E_2(\mathbb{Q}_p)$ as the set of points on E such that the p-adic order of their x-coordinates is less than or equal to negative four. Or, in other words, $\text{ord}_p(x_P) \leq -4$.

I will now present the three theorems that we need to perform an Anomalous Attack. Included here are brief explanations as to why these theorems are true but the proofs are fully explained by the references [28].

Theorem 8.1.5. *Lifting an elliptic curve preserves elliptic curve addition on the points of those curves. Or, for all $\bar{P}, \bar{Q}, \bar{R} \in \bar{E}$, if $\bar{P} \oplus \bar{Q} = \bar{R} \in \bar{E}$ and P and Q are lifts of $\bar{P}$ and $\bar{Q}$ respectively then the sum of P and Q in E will be a lift of $\bar{R}$, their sum in $\bar{E}$.*

Proof. The preservation of the elliptic curve operation makes sense here because if we started with a point in the elliptic curve over the p-adic numbers, we could find the point on the elliptic curve over $\mathbb{F}_p$ that could lift to it by reducing the x-coordinate modulo p and solving for the y coordinate and these actions would preserve the multiplication and addition that is involved in elliptic curve addition.

□

Theorem 8.1.6. *If $\#\bar{E}(\mathbb{F}_p) = n$, then adding any point on its lift $E(\mathbb{Q}_p)$ to itself n times returns a point in $E_1(\mathbb{Q}_p)$. Furthermore, adding any point on E_1 to itself n times returns a point in $E_2(\mathbb{Q}_p)$.*

Proof. Since the additive order of every point on $\bar{E}$ would have to divide n, every point added to itself n times in $\bar{E}$ would return the point at infinity, which, since lifting points on an elliptic curve preserves elliptic curve addition, would mean that every point on E added to itself n times would be a possible lift of the point at infinity on $\bar{E}$ or, in other words, be a point in the subgroup $E_1(\mathbb{Q}_p)$. The second part of this theorem is also true but much more complicated to prove [28].

□

Theorem 8.1.7. *There exists and isomorphism between the factor group of E_1 mod E_2 and the additive group on F_p. Or:*

$$E_1(\mathbb{Q}_p)/E_2(\mathbb{Q}_p) \cong F_p^+$$

Proof. The isomorphism mentioned here is a composition between the formal logarithm denoted log_F and the inverse of a projection mapping on the Weierstrass equation denoted ϑ^{-1}. We will denote the isomorphism as ψ_p. The entire proof can be found in the references [28].

□

Now, consider the elliptic curve discrete logarithm problem, or the ECDLP. For a curve $\bar{E}(\mathbb{F}_p)$ and two points $\bar{P}$ and $\bar{Q}$ we want to find m such that $m\bar{P} = Q$. It turns out this is easy to do if $\#\bar{E}(\mathbb{F}_p) = p$

Definition 8.1.8 (Smart Anomalous Attack)**.** Assume $\bar{P}$ and $\bar{Q}$ are points on an elliptic curve $\bar{E}(\mathbb{F}_p)$ where $\#\bar{E}(\mathbb{F}_p) = p$. To carry out this attack, we first calculate the lifts of the points $\bar{Q}$ and $\bar{P}$, P and Q which are in $E(\mathbb{Q}_p)$. Since $m\bar{P} - \bar{Q} = O \in \bar{E}$ we know that $mP - Q = R \in E$ where R is a lift of the point at infinity or, in other words, an element of E_1, by Theorem 8.1.5. Then, we multiply both sides of this equation by p to get $mpP - pQ = pR$ where we know that mpP and pQ are elements of E_1 and that pR is an element of E_2 by Theorem 8.1.6. Since mpP and pQ are both elements of E_1 we can apply the ψ_p defined in Theorem 8.1.7 and it turns out that the fact that all points we are running through the isomorphism are multiples of p makes the formal logarithm of these points easier to calculate and the DLP very easy to solve in the co-domain [28].

For this reason, we want to make sure that, for the elliptic curve that we choose, $\#E(F_p) \neq p$ to avoid this type of attack.

8.1.3 *MOV Reduction Attack*

Weil pairings have been used to show that you can reduce the time needed to solve the DLP for Elliptic curves to sub-exponential time if the order of the point P that Alice chooses divides $q^k - 1$ for any $1 \leq k \leq 20$. See Chapter 9 for more info.

8.2 TRACE OF FROBONEIUS

When choosing an elliptic curve to build our cryptosystem with, we want to make sure that the size of the additive group on the elliptic curve we choose, $\#E(F_p)$ does not meet any of the previously mentioned criteria, otherwise, our cryptosystems might have back doors that allow for easy decryption. In the previous chapter, we discussed that the size of the additive group on an elliptic curve $\#E(\mathbb{F}_p)$ is equal to $p + 1 - t_p$ where $|t_p| \leq 2\sqrt{p}$ (Theorem 7.3.6). Although this gives us some idea of the size of the additive group, it does not tell us if $\#E(\mathbb{F}_p)$ is divisible by a large prime or if it is equal to p. This means that the only way we are able to calculate the exact size of an additive group on an elliptic curve is to guess and check each combination of points to see which ones satisfy the elliptic curve's Weierstrass equation. This can be problematic if we are wanting to work with elliptic curves with sizes on the order of 2^{160} since it could take a very long time to calculate $\#E(\mathbb{F}_p)$. On top of that, once the size is calculated, it might not even be a secure curve meaning that the process would have to start over (in which case we have

been quite unlucky). Luckily, we can create a shortcut by considering elliptic curves over fields of prime power order, $E(F_{p^n})$.

Remark 8.2.1. How do elliptic curves over finite fields of prime power order differ from elliptic curves over finite fields of prime order. Well, since addition and multiplication are still well-defined operations that distribute and commute just like they do in fields of prime order, and because all elements in fields of prime power order are still guaranteed to have a multiplicative inverse, elliptic curve addition can be carried out in the same way in these fields. The main difference is that the elliptic curves over these fields can no longer easily be represented as being a subset of a Cartesian product of the integers $(\mathbb{Z}_n \times \mathbb{Z}_n)$ since prime powered fields are not isomorphic to any subset of the integers and the operations on them are *not* addition and multiplication modulo p^n. As a result, these curves cannot be easily visualized or graphed. Each point on the elliptic curve is a tuple of elements from the Cartesian product $\mathbb{F}_{p^n} \times \mathbb{F}_{p^n}$ of the form (x, y) where x and y satisfy the corresponding Weierstrass equation. All operations in the Wierstrass equation and in the elliptic curve addition algorithm are now the group operations on F_{p^n} rather than modular addition and multiplication [25]. From here on out we will denote our finite fields as F_q where q is implied to be a prime power for simplicity.

Definition 8.2.2 (Trace of Froboneius)**.** The Trace of Froboneius, t, of an elliptic curve, $E(\mathbb{F}_q)$, is defined as:

$$t = q + 1 - \#E(\mathbb{F}_q)$$

As we will see, this quantity has several useful applications in elliptic curve cryptography.

If we calculate, by exhaustively guessing every combination, the size of an elliptic curve over a finite field, $\#E(F_q)$, it turns out that there is an improved version of Hasses' Theorem that allows us to compute the size of the same elliptic curve over $\mathbb{F}_{q^k}$ that uses the Trace of Froboneius [27].

Theorem 8.2.3. *If $E(F_q)$ is an elliptic curve with Trace of Froboneius t, then $\#E(F_{q^k}) = q^k + 1 - \alpha^k - \beta^k$ where α and β are the complex numbers determined from the factorization:*

$$1 - tx + qx^2 = (1 - \alpha x)(1 - \beta x)$$

So what exactly is the Trace of Froboneius? It is actually derived from the Froboneius mapping on the field that the elliptic curve is over, denoted ϕ. Recall from Abstract Algebra that the characteristic of a field is the additive order of the multiplicative identity and that a field $\mathbb{F}_{p^n}$ has a characteristic of p.

Definition 8.2.4 (Froboneius Mapping)**.** Let $\mathbb{F}$ be a field with characteristic k, then the Froboneius mapping ϕ is an automorphism on $\mathbb{F}$ defined as:

$$\phi(x) = x^k \qquad \forall x \in \mathbb{F}$$

This mapping allows for short cuts for point doubling on fields of characteristic two, as we shall see later in the chapter [26]. We are also interested in fields of characteristic two because when we work with elliptic curves over these fields, of the form $\mathbb{F}_2^n$, it is easier to calculate their size. In fact, in many commercial settings where elliptic curve cryptography is employed, fields of characteristic two are used [26], [27]. However, recall from the previous chapter that, so far, all of our elliptic curve definitions and algorithms have excluded elliptic curves over fields of characteristic two and three. To understand why this is, we need to back up and understand the general form of the Weierstrass Equation.

8.3 GENERAL WEIERSTRASS

So far, in this chapter and the last, we have been calling the equations describing elliptic curves “Weierstrass Equations". Elliptic curves are special because we can define our addition algorithm on them, that is to say that a line defined by two points on the curve will pass through exactly one more point on the curve and a line tangent to the curve at any point will pass through exactly one more point. An elliptic curve achieves this characteristic by being non-singular, over the real numbers this is equivalent to saying that the curve does not have any cusps or intersections [29].

Another way to think of elliptic curves (and the way that yields the general form $Y^2 = X^3 + AX + B$ that we are used to) is as a projection of a three-dimensional *Weierstrass Elliptic Function* onto its two-dimensional affine plane. To simplify things for the purpose of this explanation of elliptic curve cryptography, and start to understand what this means, we will start with the definition of the General Weierstrass Equation [30].

Definition 8.3.1 (General Weierstrass Equation)**.** A Weierstrass Equation has the form:

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

This equation describes a Weierstrass Elliptic Function in three-dimensional space. It is called an "elliptic" function in this case because it is periodic, or repeating, in two directions. One special trait of the general Weierstrass Equation is that any equation can be projected onto the xy plane at $z = 1$ without losing any information.

Theorem 8.3.2. *Every General Weierstrass Equation can be written in the two-dimensional form*

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

without losing any information.

Proof. We can re-write the general Weierstrass equation as:

$$Z^3(\frac{Y^2}{Z^2} + a_1\frac{X}{Z}\frac{Y}{Z} + a_3\frac{Y}{Z}) = Z^3(\frac{X^3}{Z^3} + a_2\frac{X^2}{Z^2} + a_4\frac{X}{Z} + a_6)$$

$$\frac{Y^2}{Z^2} + a_1\frac{X}{Z}\frac{Y}{Z} + a_3\frac{Y}{Z} = \frac{X^3}{Z^3} + a_2\frac{X^2}{Z^2} + a_4\frac{X}{Z} + a_6$$

Therefore, if we consider the variables $x = X/Z$ and $y = Y/Z$ we can make this a two dimensional equation:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Which represents x-y cross section of the curve at $Z = 1$. The idea is that for every point on the three dimensional curve, (X, Y, Z) we can assign a point $(X/Z, Y/Z, Z/Z)$ which we can write as a two dimensional point without losing any information since the third coordinate will always be one. This two dimensional point is (x, y) and it lies on the two dimensional curve defined above. Therefore, to check and see if a point (x_1, y_1, z_1) satisfies the Weierstrass equation it is equivalent to check and see if the point $(\frac{x_1}{z_1}, \frac{y_1}{z_1})$ is on the two dimensional curve. Therefore we can write the three dimensional curve in two dimensions without losing any information. The one exception is that we cant perform this check when $Z = 0$. However, if we look at the three dimensional Weierstrass equation when $Z = 0$, it reduces to $X^3 = 0$ or $X = 0$. The line defined when X and Z equal zero in three dimensions is just the y-axis. the y axis underneath our cross section at $z = 1$ intersects the cross section at $(0, \infty, 0)$ when all lines intersect. So we add the point $(0, \infty)$ to our two dimensional curve so that we know that if we check to see if a point with a zero z coordinate satisfies the three dimensional Weierstrass equation, it will only do so if the point lies at the y axis. We call this point O and it is where all lines in the y direction intersect. This process of projecting a three dimensional equation onto two dimensions is called using non-homogeneous coordinates [30] □

Now the statement that an elliptic curve is the projection of a Weierstrass elliptic function is starting to make more sense. In fact the form derived in the previous proof *does* describe an elliptic curve. Why then, are we able to confidently say that every elliptic curve has the form $Y^2 = X^3 + AX + B$? To further simplify this equation, we need to use *Elliptic Curve Isomorphisms*.

Definition 8.3.3 (Elliptic Curve Isomophism)**.** Two elliptic curves E and E' are considered isomorphic if a function exists that maps each point of E to a point of E' that is bijective and preserves elliptic curve addition. Furthermore, the infinity point of E must be mapped to the infinity point of E' [30].

Two elliptic curves that are isomorphic will not look the same on a graph and they do not describe sets of the same points, however, for the purposes of creating elliptic curve cryptosystems and solving the DLP, two elliptic curves that are isomorphic are interchangeable because all of the relationships between the points are preserved. We just need a function to consider that would be able to reliably map points in this way to begin to simplify the Weierstrass equation further.

Theorem 8.3.4. *Two Elliptic curves E and E' over a finite field $\mathbb{F}_q$ are isomorphic if and only if they are related by a substitution of variables of the form $x = u^2x + r$ and $y = u^3y + su^2x + t$ for some $u, r, s, t \in \mathbb{F}_q$ where $u \neq 0$. That is to say that a function ϕ that maps the points on E to the points on E' is an isomorphism if and only if it maps every point (x, y) on E to the point $(u^2x + r, u^3y + su^2x + t)$ on E' for some $u, r, s, t \in \mathbb{F}_q$ where $u \neq 0$ [30].*

Proof. Let E be an elliptic curve of the form $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$. First of all note that substituting x and y with $x = u^2x + r$ and $y = u^3y + su^2x + t$ in this equation will still allow it to remain an elliptic curve of this general form with enough foiling and rearranging since no x^4 or y^3 terms will be introduced. If we think about how elliptic curve addition works geometrically, if $P \oplus Q = R$ for some points P, Q, and R in E, it means that P, Q, and R are co-linear, or on the same line. A function ϕ mapping E to E' preserves elliptic curve addition if and only if $\phi(P) \oplus \phi(Q) = \phi(R)$ on E'. Since elliptic curves have the property that only up to three points on them can be co-linear, the function ϕ will preserve elliptic curve addition if and only if the points $\phi(P), \phi(Q)$, and $\phi(R)$ are co-linear on E'. This means that ϕ preserves elliptic curve addition if and only if it changes the x and y coordinates of each point with a linear transformation of the form $\phi((x, y)) = (m_1x + b_1, m_2y + b_2)$. Since only these transformations will map straight lines on E to straight lines on E'. Without loss of generality, lets assume that this transformation changes the x coordinate like so: $x \to u^2x + r$. Now, we want to transform the y coordinate linearly as well but this transformation is also constrained by the relation $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$. Plugging in the transformed x-coordinate and solving for y tells us that the most general linear transformation we can do on the y-coordinate is $y \to u^3y + su^2x + t$ which still takes the form of a linear transformation on y. Therefore a function ϕ that maps the points on E to the points on E' is an isomorphism if and only if it maps every point (x, y) on E to the point $(u^2x + r, u^3y + su^2x + t)$ on E' for some $u, r, s, t \in \mathbb{F}_q$ where $u \neq 0$. □

Now, it turns out that with substitutions of variables of this form, we can show that every elliptic curve of the most general form $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ is isomorphic to an elliptic curve of the form $Y^2 = X^3 + AX + B$. The process of showing this will also show why we can only use this form when the characteristic of the field the elliptic curve is defined over is *not* two or three. Consider the following lemma and theorem. The language of these proofs was kept general enough so that it could be applied to fields of prime power order where the elements are not integers.

Lemma 8.3.5. *A non-identity element of a field can be split in half if and only if the field does not have a characteristic of two. A non-identity element of a field can be split into thirds if and only if the field does not have a characteristic of three.*

Proof. Let $\mathbb{F}$ be a field whose characteristic is not two. Let a be an arbitrary element of $\mathbb{F}$. Consider the element 2 in $\mathbb{F}$ which we define as the unity added to itself 2 times or $(1+1)$. Since the characteristic of $\mathbb{F}$ is not two, the additive order of the unity is not two and $1+1 \neq 0$ which, by definition of a field means that 2 has a multiplicative inverse 2^{-1}. Consider the element $a * 2^{-1}1$. Note now that $a * 2^{-1} + a * 2^{-1} = a * 2 * 2^{-1} = a(1) = a$ since multiplication is defined as repeated addition and is communicative on the elements of $\mathbb{F}$. Therefore, any non-identity element can be split in half in $\mathbb{F}$. A similar proof shows that any non-identity element can be split into thirds in a field that does not have a characteristic of three.

Now, Let $\mathbb{F}$ be a field of characteristic two let a be an arbitrary non-identity element that can be split in half. This means that $a = b + b$ where b is another element of $\mathbb{F}$. This can also be written as $a = b + b = (1+1)b$ where 1 is the unity of $\mathbb{F}$ since multiplication is defined as repeated addition and must distribute over addition in a field. However, since the characteristic of $\mathbb{F}$ is two the additive order of the unity is two and therefore $1+1=0$. This implies that $a = (1+1)b = 0b = 0$. This is a contradiction because we assumed that a was a non-identity element. Therefore, no non-identity element in $\mathbb{F}$ can be split in half. A similar proof shows that splitting a non-identity element into thirds in a field of characteristic three is also impossible.

□

Theorem 8.3.6. *Every elliptic curve over a finite field with a characteristic that is not two or three is isomorphic to a curve of the form*

$$Y^2 = X^3 + AX + B$$

where A and B are elements of the field.

Proof. Consider an arbitrary elliptic curve, E, of the form $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ over a field $\mathbb{F}$ which has a characteristic that is not two or three. Since the characteristic of the field is not two, we can choose the following substitution of coordinates that is of the form described in Theorem 8.3.4, $x \to x$ and $y \to y - \frac{a_1}{2}x - a_32$. Substituting these variables into E gives us the new, isomorphic, elliptic curve E' which, after much foiling and rearranging can be shown to have the form $y^2 = x^3 + a_2'x^2 + a_4'x + a_6'$ where $a_2' = a_2 + \left(\frac{a_1}{2}\right)^2$ and $a_4' = a_4 + \left(\frac{a_1}{2}\right)\left(\frac{a_3}{2}\right)$ and $a_6' = a_6 + \left(\frac{a_3}{2}\right)^2$.

We can now do a substitution of variables on E' to arrive at another elliptic curve that is isomorphic to E. This time, using the fact that the characteristic of the field is not three, we substitute $x \to x - \frac{a_2'}{3}$ and $y \to y$. Substituting this into E' and simplifying

gives us the new elliptic curve E'' of the form $y^2 = x^3 + a_4''x + a_6''$ where $a_4'' = a_4' - \frac{a_2'^2}{3}$ and $a_6'' = a_6' + 2\left(\frac{a_2'}{3}\right)^3 - \frac{1}{3}a_2'a_4'$. Note that E'' has the form that we know and love, $Y^2 = X^3 + AX + B$ and is isomorphic to our arbitrary starting curve E [30].

□

As you can see, we can only make these substitutions of variables and simplify the Weierstrass equation in this way when we can split elements into halves and thirds, which, by the above lemma, is only possible when the characteristic of the field we are working in is not two or three. This means that when the field we are working in *does* have charactersitc two, like prime power fields of the form $\mathbb{F}_{2^n}$ do, the elliptic curve equations, and elliptic curve addition algorithms have to take entirely different forms.

8.4 ELLIPTIC CURVES OVER FIELDS OF CHARACTERISTIC TWO

In fields of characteristic two, we cannot simplify our elliptic curve equation nearly as much as we can in fields of characteristic greater than three. Since the addition algorithm for the most general form of an elliptic curve, $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ would be horribly long and tedious, cryptographers usually split elliptic curves over fields of characteristic two into two groups, those with a zero j-invariant and those with a non-zero j-invariant. Each of these types of elliptic curve is able to be simplified to a reasonable length in fields of characteristic two. This begs the question: What is the j-invariant of a curve? To answer this, we also need a more general definition for the discriminant of a curve, ΔE which we defined for curves over fields with characteristic greater than three in the last chapter. Keep in mind that for these definitions to also apply to fields of prime power order, 0 must represent the additive identity of the field, rather than the integer [31]

Definition 8.4.1 (General Discriminant)**.** If we express our elliptic curve, E, as a function $f(x)$ the discriminant can be defined using the roots of the equation $f(x) = 0$ or e_1, e_2 and e_3 in the equation $f(x) = (x - e_1)(x - e_2)(x - e_3)$. The discriminant of E is given by:

$$\Delta E = 16(e_1 - e_2)^2(e_1 - e_3)^2(e_2 - e_3)^2$$

Definition 8.4.2 (j-Invariant)**.** For an elliptic curve $E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ the j-invariant, $j(E)$ is given by:

$$j(E) = \frac{((a_1^2 + 4a_2)^2 - 24(2a_4 + a_1a_3))^3}{\Delta E}$$

Definition 8.4.3 (Elliptic Curves with a Zero j-Invariant Over Fields of Characteristic Two)**.** Any elliptic curve, E, over a field of characteristic two can be simplified to the form:

$$y^3 + cy = x^3 + ax + b$$

Where a, b, and c are elements of the field and $c \neq 0$ if the curve's j-invariant is equal to zero [27].

Definition 8.4.4 (Elliptic Curves with a Non-Zero j-Invariant Over Fields of Characteristic Two)**.** Any elliptic curve, E, over a field of characteristic two can be simplified to the form:

$$y^2 + xy = x^3 + ax^2 + b$$

Where a and b are elements of the field and $b \neq 0$ if the curve's j-invariant is *not* equal to zero [27].

Now, we need to define two new addition algorithms for adding points of these two types of elliptic curves. These definitions can be found in more detail in the references but they act very similarly to the addition algorithms we defined in the previous chapter. However, it is more difficult to visualize these geometrically since they are usually applied to elliptic curves over finite fields of prime power order [27]

Theorem 8.4.5. *Let $E: \quad y^2 + cy = x^3 + ax + b$ be an elliptic curve, and let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two non-O points on E.*

- *If $P_1 = -P_2$ then $P_1 \oplus P_2 = O$*
- *Otherwise, define Then $P_1 \oplus P_2 = (x_3, y_3)$. by*

$$x_3 = \left\{ \begin{array}{ll} \left(\frac{y_1+y_2}{x_1+x_2}\right) + x_1 + x_2 & \text{if } P_1 \neq P_2 \\ \\ \frac{x_1^4+a^2}{c^2} & \text{if } P_1 = P_2 \end{array} \right\}$$

and

$$y_3 = \left\{ \begin{array}{ll} \frac{y_1+y_2}{x_1+x_2}(x_1 + x_3) + y_1 + c & \text{if } P_1 \neq P_2 \\ \\ \frac{x_1^2+a}{c}(x_1 + x_3) + y_1 + c & \text{if } P_1 = P_2 \end{array} \right\}$$

Theorem 8.4.6. *Let $E: \quad y^2 + xy = x^3 + ax^2 + b$ be an elliptic curve, and let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two non-O points on E.*

- *If $P_1 = -P_2$ then $P_1 \oplus P_2 = O$*

- *Otherwise, define Then* $P_1 \oplus P_2 = (x_3, y_3)$. *by*

$$x_3 = \left\{ \begin{array}{cc} \left(\frac{y_1+y_2}{x_1+x_2}\right) + \frac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a & \text{if } P_1 \neq P_2 \\ \\ x_1^2 + \frac{b}{x_1^2} & \text{if } P_1 = P_2 \end{array} \right\}$$

and

$$y_3 = \left\{ \begin{array}{cc} \frac{y_1+y_2}{x_1+x_2}(x_1 + x_3) + y_1 + x_3 & \text{if } P_1 \neq P_2 \\ \\ x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right)x_3 + x_3 & \text{if } P_1 = P_2 \end{array} \right\}$$

8.5 CHOOSING A SECURE ELLIPTIC CURVE

Now we have all the tools we need to be able to choose a good elliptic curve to build a cryptosystem with and also to be able to perform elliptic curve addition on the points of that curve, even if it is over a field with characteristic two.

Algorithm 8.5.1 (Method for Choosing a Secure Elliptic Curve).

- Choose an elliptic curve, E, over the field F_{2^m} for some large m.
- Calculate $\#E(F_{2^l})$ where E is the same elliptic curve and l is a small number that divides m.
- Calculate the Trace of Frobenius for $E(F_{2^l})$ and then use Hasse's Improved Theorem to calculate $\#E(F_{2^m})$.
- Check and see if $\#E(F_{2^m})$ is divisible by a large prime and that it is not equal to 2^m. If it does not meet either of these criteria, go back to the beginning and choose a different curve or finite field.

9

DEMYSTIFYING THE WEIL PAIRING (BY ALICIA NICHOLS)

Following the previous chapter which depicted the different attacks on elliptic curve cryptosystems due to the reduction of the ECDLP and how (through our choices of elliptic curves and finite fields) these risks can be eliminated, this chapter will focus on one way in which the points on these elliptic curves relate with each other to form both cryptosystems and the attacks that risk their security: the Weil Pairing. A specific type of bilinear pairing (a method of forming one output from 2 inputs) used in Pairing-Based Cryptography, the Weil Pairing was defined in 1940 by Andre Weil based in Picard Groups, but was then expanded throughout the following years to equivalent definitions on elliptic curves, which will be our focus. Although it was used as a tool for the cryptological attacker before the 90's-creating algorithms such as the MOV Attack-it has become a useful tool for cryptographers as it allows for the expansion of the much aforementioned Diffie-Hellman Key Exchange Cryptosystem from bipartite to tripartite. In order to portray the power of the Weil Pairing this chapter will delve into its strenuous definition, explore its the different properties and finally showcase its prowess within the dangerous MOV Attack and the incredibly useful Tripartite DH Key Exchange Crytosystem.

9.1 FANTASTIC FUNCTIONS AND THEIR DAUNTLESS DIVISORS

The modern definition of the Weil Pairing of points on an elliptic curve lies in the relation between functions on said elliptic curve and the divisors of these functions (which will be defined below) so to understand the Weil Pairing, we must start with the expanded form of a rational function.

A rational function $f(X)$ factored in the complex plane has the form

$$f(X) = \frac{a(X-\alpha_1)^{e_1}(X-\alpha_2)^{e_2}\cdots(X-\alpha_r)^{e_r}}{b(X-\beta_1)^{d_1}(X-\beta_2)^{d_2}\cdots(X-\beta_s)^{d_s}}$$

We denote the values $\alpha_1, \cdots \alpha_r$ as the *zeros of* $f(X)$ and $\beta_1, \cdots \beta_r$ as the *poles of* $f(X)$ which are the values where $f(X)$ goes to zero and $f(X)$ is undefined, respectively. Their

exponents $e_1 \cdots e_r$ and $d_1 \cdots d_r$ are their associated *multiplicities*. We use these zeros and poles and their multiplicities to define the *divisor of a function*.

Definition 9.1.1 (Divisor of a Function)**.** For a rational function $f(X)$, the divisor of $f(X)$ is the formal sum

$$div(f(X)) = e_1[\alpha_1] + e_2[\alpha_1] \cdots e_r[\alpha_r] - d_1[\beta_1] - d_2[\beta_2] \cdots d_r[\beta_r]$$

such that $\alpha, \beta \in \mathbb{C}$ and $d, e \in \mathbb{Z}$

How does a divisor of an arbitrary rational function relate to a function on a specific elliptic curve? When we examine the functions over an elliptic curve $E : Y^2 = X^3 + AX + B$ of the form $f(X, Y)$ we see that there exists points on E such that f has a zero and points such that f has a pole. When we assign multiplicities to these points we can extend our definition of a divisor of a function to a divisor of a function over an elliptic curve $E(\mathbb{F}_p)$

Definition 9.1.2 (Divisor of a Function on E)**.** Let E be an elliptic curve over a field $\mathbb{F}_p$ and a point P on $E(\mathbb{F}_p)$, a divisor of a function on E is the formal sum of the multi-set of points on E denoted

$$div(f) = \sum_P a_P[P]$$

where $a_P \in \mathbb{Z}$ are the coefficients of the divisor and only a finite number of $a_P = 0$.

Since an elliptic curve is also a function we can find its divisor and thus can apply its form to any specific elliptic curve we examine.

Example 9.1.3. Let $E : Y^2 = X^3 + AX + B$. To find $div(Y)$ first consider its factorization. Since E is a cubic we know that it has three zeros, $\alpha_1, \alpha_2, \alpha_3$, in its factorization which we can represent as

$$E : Y^2 = X^3 + AX + B = (X - \alpha_1)^1(X - \alpha_2)^1(X - \alpha_3)^1$$

Thus the points $P_1 = (\alpha_1, 0)$, $P_2 = (\alpha_2, 0)$, and $P_3 = (\alpha_3, 0)$ represent these zeros. Since E has a point of inflection at the point $Z = 0$, homogenizing our elliptic curve equation we see that the only pole of the function is at infinity and furthermore since E is a cubic this pole must have multiplicity three. we see that

$$div(Y) = 1[P_1] + 1[P_2] + 1[P_3] - 3[O]$$

With this generic form of an elliptic curve divisor we can define any general divisor of an elliptic curve function, which will provide us with an extremely useful theorem that is intrinsic to the modern definition and calculation of the Weil Pairing.

Definition 9.1.4 (Divisor on E)**.** Let $E(\mathbb{F}_p)$ be an elliptic curve. We formally define a divisor on E as any formal sum

$$D = \sum_{P \in E(\mathbb{F}_p)} a_P[P]$$

for $a_P \in \mathbb{Z}$ and only a finite number of $a_P = 0$

Furthermore we can define the *Degree*, *Sum*, and *Support* of a divisor as

Definition 9.1.5 (Degree)**.** Given a divisor D of an elliptic curve $E(\mathbb{F}_p)$ and a point P on $E(\mathbb{F}_p)$, the degree of D is defined as the sum of the coefficients of D

$$deg(D) = \sum_{P} a_P$$

where $a_P \in \mathbb{Z}$.

Definition 9.1.6 (Sum)**.** Given a divisor D of an an elliptic curve $E(\mathbb{F}_p)$ and a point P on $E(\mathbb{F}_p)$, the sum of D is the sum of the product of the coefficients of D and P

$$sum(D) = \sum_{P \in E(\mathbb{F}_p)} a_P P$$

where $a_P P$ denotes $\underbrace{P + P + \cdots + P}_{a_P - times}$

Definition 9.1.7 (Support)**.** Given a divisor D of an an elliptic curve $E(\mathbb{F}_p)$ and a point P on $E(\mathbb{F}_p)$, the support of D is the set

$$supp(D) = \{P \in E(\mathbb{F}_p) | a_P \neq 0\}$$

Remark 9.1.8. For the purposes of this chapter, for points P and Q, $P + Q$ will denote $P \oplus Q$ and $P - Q$ will denote $P \oplus (-Q)$ in accordance to the addition algorithm 7.1.3 since the definition of the Weil Pairing is quite confusing.

So we've defined divisors on an elliptic curve E and the divisors of a function over E but how do they relate to each other? If there exists a divisor for a function over E does

that divisor also divide the entire curve? Conversely, does a divisor on E necessarily have to divide any function over E. The following theorem answers these question and is the key to how we define functions and their divisors in the definition of the Weil Pairing.

Theorem 9.1.9. *A divisor D of a curve $E(\mathbb{F}_p)$ is a divisor of a function f if and only if $deg(D) = 0$ and $sum(D) = O$*

Proof. Using Weierstrass Equations, this proof can be completed by considering a group isomorphism defined as $\gamma : \mathbb{C}/ < w_1, w_2 > \to E(\mathbb{C})$. See proof in [32] □

Remark 9.1.10. By this theorem, it follows that for any divisor D on $E(\mathbb{F}_p)$, if $deg(D) = 0$ and $sum(D) = O$ then there must exist a function f on E such that $div(f) = D$. Consider the divisor $D = n[P] - n[O]$ for a point P on $E(\mathbb{F}_p)$. Notice that $deg(D) = n - n = 0$ and $sum(D) = nP - nP = \underbrace{(P + \cdots + P)}_{n-times} - \underbrace{(P + \cdots + P)}_{n-times} = O$. Thus for any divisor of the form $D = n[P] - n[O]$ for any point P on E, there exists a function f_P such that $div(f_P) = n[P] - n[O]$.

With the result of this important theorem, the construction of the function in terms of its zeros and poles and the definition of divisors on elliptic curves, and we can now define the Weil Pairing.

9.2 A DARING DEFINITION

Definition 9.2.1 (Weil Pairings)**.** Let $E(\mathbb{F}_p)[n]$ be the set of points on $E(\mathbb{F}_p)$ whose order divides n and let $P, Q \in E(\mathbb{F}_p)[n]$. Let f_P and f_Q be rational functions on E such that

$$div(f_P) = n[P] - n[O] \quad \text{and} \quad div(f_Q) = n[P] - n[O]$$

The Weil Pairing of P and Q is the quantity

$$w_n(P, Q) = \frac{f_P(Q + S)}{f_P(S)} \Big/ \frac{f_Q(P - S)}{f_Q(-S)}$$

such that S is an arbitrary point on $E(\mathbb{F}_p)$ where $S \notin \{O, P, -Q, P - Q\}$

Thus we can understand the Weil Pairing in terms of the relation of arbitrary functions with specific divisors evaluated at points of the elliptic curve. This quantity qualifies as a bilinear pairing because it takes two inputs and produces one output. In this case, what

makes the Weil Pairing unique is its extremely interesting output: an n-th root of unity in the field F_p. These interesting properties of the Weil Pairing along with their proofs of are defined below. However before they can be proven we need the following property: Weil Reciprocity.

9.2.1 *Properties*

Theorem 9.2.2 (Weil Reciprocity)**.** *Let f_P and f_Q be functions over an elliptic curve $E(\mathbb{F}_p)$ such that their divisors have disjoint supports, then*

$$f_P(div(f_Q)) = f_Q(div(f_P))$$

Proof. This theorem is proven in terms of Riemann Surfaces which is outside of the scope of this textbook so it will be excluded, however the reader is directed to [33]. □

Thus we can now define and prove the properties of the Weil Pairing

Theorem 9.2.3 (Properties of Weil Pairings)**.** *Let P and Q be arbitrary points in $E(F_p)[n]$ then the Weil Pairing $w_n(P, Q)$ has the following properties:*

1. *The Weil Pairing is an n-th root of unity*

$$w_n(P, Q)^n = 1$$

2. *The Weil Pairing is Bilinear:*

$$w_n(P_1 + P_2, Q) = w_n(P_1, Q)w_n(P_2, Q)$$

3. *The Weil Pairing is Alternating:*

$$w_n(P, P) = 1 \rightarrow w_n(P, Q) = w_n(Q, P)^{-1}$$

4. *The Weil Pairing is nondegenerate:*

$$\text{if } w_n(P, Q) = 1 \text{ for all } Q \in E(\mathbb{F}_p) \text{ then } P = O$$

Proof. 1 : The Weil Pairing is an n-th root of unity

Let P and Q be points in the set $E(\mathbb{F}_p)[n]$ and consider $w_n(P, Q)$. By definition

$$w_n(P, Q) = \frac{f_P(Q + S)}{f_P(S)} \Big/ \frac{f_Q(P - S)}{f_Q(-S)}$$

Consider $\left(\frac{f_P(Q+S)}{f_P(S)}\right)^n$ and notice

$$\left(\frac{f_P(Q+S)}{f_P(S)}\right)^n = f_P(Q+S)^n f_P(S)^{-n}$$

Since $div(f_P) = n[P] - n[O]$, $div(f_P(Q+S)) = n[(Q+S)] - n[S]$ by definition of Weil Pairings. Thus the product $f_P(Q+S)^n f_P(S)^{-n}$ has the form of the function f_P evaluated at the divisor $n[(Q+S)] - n[S]$ and can be written as $f_P(n(Q+S) - n(S))$ by definition and by bilinearity (which is proven below). Also notice that $n(Q+S) - n(S)$ is the divisor of the function f_Q evaluated at $(X - S)$ and so

$$f_P(n(Q+S) - m(S)) = f_P(div(f_Q(X-S))) = f_Q(div(f_P(X-S))) = f_Q(n[P-S] - n[-S])$$

by Weil Reciprocity. By similar logic since the product $f_Q((P-S)^n f_Q(-S)^{-n})$ has the form of the function f_Q evaluated at the divisor $n[P-S] - n[-S]$, it follows that

$$f_Q(n[P-S] - n[-S]) = f_Q((P-S)^n f_Q(-S)^{-n}) = \left(\frac{f_Q(P-S)}{f_Q(-S)}\right)^n$$

Thus, $\left(\frac{f_P(Q+S)}{f_P(S)}\right)^n = \left(\frac{f_Q(P-S)}{f_Q(-S)}\right)^n$ and by substitution we see that

$$w_n(P,Q)^n = \left(\frac{f_P(Q+S)}{f_P(S)}\right)^n \Big/ \left(\frac{f_Q(P-S)}{f_Q(-S)}\right)^n = \left(\frac{f_Q(P-S)}{f_Q(-S)}\right)^n \Big/ \left(\frac{f_Q(P-S)}{f_Q(-S)}\right)^n = 1$$

By definition $w_n(P,Q)$ is an n-th roots of unity.

□

Proof. 2 : The Weil Pairing is Bilinear

Let P_1, P_2 and Q be points in the set $E(\mathbb{F}_p)[n]$ and consider $w_n(P_1+P_2, Q)$ set equal to $w_n(P_1,Q)w_n(P_2,Q)$. By definition this equals

$$\frac{f_{P_1+P_2}(Q+S)}{f_{P_1+P_2}(S)} \Big/ \frac{f_Q(P_1+P_2-S)}{f_Q(-S)} = \left(\frac{f_{P_1}(Q+S)}{f_{P_1}(S)} \Big/ \frac{f_Q(P_1-S)}{f_Q(-S)}\right)\left(\frac{f_{P_2}(Q+S)}{f_{P_2}(S)} \Big/ \frac{f_Q(P_2-S)}{f_Q(-S)}\right)$$

By rearranging and cross multiplying we get

$$\frac{f_{P_1+P_2}(Q+S)}{f_{P_1}(Q+S)f_{P_2}(Q+S)}\Big/\frac{f_{P_1+P_2}(S)}{f_{P_1}(S)f_{P_2}(S)}=\frac{f_Q(P_1+P_2-S)f_Q(-S)}{f_Q(P_1-S)f_Q(P_2-S)}$$

Consider the left side of this equation and consider a function F_{P_1,P_2} such that

$$F_{P_1,P_2}(X)=\frac{f_{P_1+P_2}(X)}{f_{P_1}(X)f_{P_2}(X)}$$

We can define this since, by definition, a function is defined in the Weil Pairing if their divisors have the form $div(f_X)=n[X]-n[X]$ and since

$$div(F_{P_1,P_2})=(n[P_1+P_2]-n[O])-((n[P_1]-n[O])-(n[P_2]-n[O])$$

F_{P_1,P_2} has this form. Notice this equals $m([P_1+P_2]-[P_1]-[P_2]+[O])$. Now let G_{P_1,P_2} be a function such that $div(G_{P_1,P_2})=[P_1+P_2]-[P_1]-[P_2]+[O]$ which implies that $(m)div(G_{P_1,P_2})$ is defined and $div(F_{P_1,P_2})=(m)div(G_{P_1,P_2})$. Thus $F_{P_1,P_2}(X)=G_{P_1,P_2}(X)^m$ by their construction.

$$\frac{f_{P_1+P_2}(Q+S)}{f_{P_1}(Q+S)f_{P_2}(Q+S)}\Big/\frac{f_{P_1+P_2}(S)}{f_{P_1}(S)f_{P_2}(S)}=\frac{F_{P_1,P_2}(Q+S)}{F_{P_1,P_2}(S)}=\frac{G_{P_1,P_2}(Q+S)^n}{G_{P_1,P_2}(S)^n}$$

This equals the product $(G_{P_1,P_2}(Q+S)^n)(G_{P_1,P_2}(S)^{-n})$ which is of the form of G_{P_1,P_2} evaluated at the divisor of $f_Q(X-S)$ by definition of Weil Pairings. So

$$(G_{P_1,P_2}(Q+S)^n)(G_{P_1,P_2}(S)^{-n})=G_{P_1,P_2}(div(f_Q(X-S))=f_Q(div(G_{P_1,P_2}(X-S))$$

Notice

$$f_Q(div(G_{P_1,P_2}(X-S))=f_Q([P_1+P_2-S]-[P_1-S]-[P_2-S]+[-S])$$

which equals

$$\frac{f_Q(P_1+P_2-S)f_Q(-S)}{f_Q(P_1-S)f_Q(P_2-S)}$$

This is precisely the right side of our original expanded equation, thus

$w_n(P_1+P_2,Q)=w_n(P_1,Q)w_n(P_2,Q)$ and Weil Pairings are bilinear. □

Remark 9.2.4. By this result if we consider $w_n(kP,lQ)$ for some integers k and l we see that

$$w_n(kP,lQ)=w_n(\underbrace{P+\cdots+P}_{k-times},\underbrace{Q+\cdots+Q}_{l-times})=\underbrace{w_n(P,Q)\cdots w_n(P,Q)}_{kl-times}=w_n(P,Q)^{lk}$$

This result is intrinsic to one of the most common applications of the Weil Pairing, the Tripartite Diffie-Hellman Problem, which will be defined later in this chapter

Proof. 3 : The Weil Pairing is Alternating

Let $P \in E(\mathbb{F}_p)[n]$ and consider $w_n(P,P)$. By definition this equals

$$w_n(P,P) = \frac{f_P(P+S)}{f_P(S)} \Big/ \frac{f_Q(P-S)}{f_Q(-S)}$$

Since S is arbitrary by definition we can let $S = O$ and the equation becomes

$$w_n(P,P) = \frac{f_P(P)}{f_P(O)} \Big/ \frac{f_Q(P)}{f_Q(O)} = 1$$

Furthermore, by this property for some point $Q \in E(\mathbb{F}_p)[n]$, $w_n(P+Q,P+Q) = 1$. Thus by bilinearity

$$\begin{aligned} w_n(P+Q,P+Q) &= w_n(P,P)w_n(P,Q)w_n(Q,P)w_n(Q,Q) \\ 1 &= (1)w_n(P,Q)w_n(Q,P)(1) \\ &= w_n(P,Q)w_n(Q,P) \\ w_n(P,Q) &= w_n(Q,P)^{-1} \end{aligned}$$

By definition of inverses. Thus the Weil Pairing is Alternating □

Proof. 4 : The Weil Pairing is nondegenerate The proof of this property is through another definition of the Weil Pairing and thus is not included here but the reader is referred to [34]. □

Thus the Weil Pairing is defined and its properties understood, but how can this complex quantity be calculated? Multiple methods have been developed, from calculating the quantity directly from the definition (which is strenuous and easily erred) to programs like Millers Algorithm that calculate the quantity quickly and efficiently. The purpose behind these calculations is to find the functions that satisfy the desired divisors.

9.3 COMPLEX CALCULATION

9.3.1 *Direct Calculation*

We can calculate a Weil Pairing w_2 directly from its definition using the elliptic curve addition algorithm 7.1.3.

As stated previously, an elliptic curve $E(\mathbb{F}_p)[2]$ for some prime p has the form

$$E : Y^2 = X^3 + AX + B = (X - \alpha_1)(X - \alpha_2)(X - \alpha_3)$$

such that the zeros of the curve are the points

$$P_1 = (X - \alpha_1) \quad P_2 = (X - \alpha_2) \quad P_3 = (X - \alpha_3)$$

Note that $\alpha_1 + \alpha_2 + \alpha_3 = 0$ since the elliptic curve equation has no x^2 term and also note that P_1, P_2, P_3 all have order 2.

Consider $w_2(P_1, P_2)$. By definition there exists functions f_{P_1} and f_{P_2} such that $div(f_{P_1}) = 2[P_1] - 2[O]$ and $div(f_{P_2}) = 2[P_2] - 2[O]$. Notice that the functions $(X - \alpha_1)$ and $(X - \alpha_2)$ are functions with these divisors so let $f_{P_1} = X - \alpha_1$ and $f_{P_2} = X - \alpha_2$. So by definition and substitution it follows that

$$w_2(P_1, P_2) = \frac{X(P_2 + S) - \alpha_1}{X(S) - \alpha_1} \Big/ \frac{X(P_1 - S) - \alpha_2}{X(-S) - \alpha_1}$$

for a point $S = (x, y) \in E(\mathbb{F}_p)[2]$, where $X(P_1 - S)$ denotes the x-coordinate of the point $P_1 - S$.

By the addition algorithm this equals

$$\begin{aligned} X(P_1 - S) &= \left(\frac{0 - y}{x - \alpha_1}\right)^2 - \alpha_1 - x \\ &= \frac{y^2 - (x - \alpha_1)^2(x + \alpha_1)}{(x - \alpha_1)^2} \\ &= \frac{(x - \alpha_1)(x - \alpha_2)(x - \alpha_3) - (x - \alpha_1)^2(x + \alpha_1)}{(x - \alpha_1)^2} \end{aligned}$$

Since $Y^2 = (X - \alpha_1)(X - \alpha_2)(X - \alpha_3)$ this equals

$$= \frac{(x - \alpha_2)(x - \alpha_3) - (x - \alpha_1)(x + \alpha_1)}{(x - \alpha_1)}$$
$$= \frac{x(-\alpha_2 - \alpha_3) + \alpha_2\alpha_3 + \alpha^1}{(x - \alpha_1)}$$

Since $\alpha_1 + \alpha_2 + \alpha_3 = 0$ this equals

$$= \frac{\alpha_1 x + \alpha_2\alpha_3 + \alpha_1}{(x - \alpha_1)}$$

So we see that

$$X(P_1 - S) = \frac{\alpha_1 x + \alpha_2\alpha_3 + \alpha_1^2}{(x - \alpha_1)}$$

and by similar logic

$$X(P_2 - S) = \frac{\alpha_2 x + \alpha_1\alpha_3 + \alpha_2^2}{(x - \alpha_2)}$$

Therefore when we substitute these quantities into the definition of the Weil Pairing it becomes

$$\begin{aligned} w_2(P_1, P_2) &= \frac{X(P_2 + S) - \alpha_1}{X(S) - \alpha_1} \Big/ \frac{X(P_1 - S) - \alpha_2}{X(-S) - \alpha_1} \\ &= \frac{\frac{\alpha_2 x + \alpha_1\alpha_3 + \alpha_2^2}{(x - \alpha_2)} - \alpha_1}{x - \alpha_1} \Big/ \frac{\frac{\alpha_1 x + \alpha_2\alpha_3 + \alpha_1^1}{(x - \alpha_1)} - \alpha_2}{x - \alpha_1} \\ &= \frac{x(\alpha_2 - \alpha_1) + \alpha_1\alpha_3 + \alpha_2^2 + \alpha_1\alpha_2}{x(\alpha_1 - \alpha_2) + \alpha_2\alpha_3 + \alpha_1^2 + \alpha_1\alpha_2} \\ &= \frac{x(\alpha_2 - \alpha_1) + \alpha_2^2 - \alpha_1^2}{x(\alpha_1 - \alpha_2) + \alpha_1^2 - \alpha_2^2} \\ &= -1 \end{aligned}$$

Thus we see that for any points P and Q on an elliptic curve $E(\mathbb{F}_2)$, the Weil Pairing outputs an 2nd root of unity.

Although this method stems directly from the definition it is tedious even to calculate it when $n = 2$ thus it becomes even more tedious to find functions that satisfy the necessary divisors and calculate them as the value of n increases. To avoid this cryptographers developed multiple computer algorithms that find functions that satisfy the

required divisors and calculate the quantity. One such algorithm, Millers Algorithm, is pictured but will not be computed here as it features a definition not used in this chapter and thus is outside of the scope of this text. For a more in depth look at the how exactly this algorithm is computed through the necessary programs, the reader is referred to [35]

Algorithm 4.1 Miller's Algorithm [2]

Input: $P \in E(\mathbb{F}_{q^k})[r], D_Q \sim (Q) - (\mathcal{O})$ with support disjoint from $(f_{r,P})$, and $r = (r_{n-1}...r_1r_0)_2$ with $r_{n-1} = 1$.

Output: $f_{r,P}(D_Q) \leftarrow f$.

1: $R \leftarrow P, f \leftarrow 1$.
2: **for** $i = n - 2$ down to 0 **do**
3: Compute the line function $\ell_{R,R}$.
4: $R \leftarrow [2]R$.
5: Compute the line function ν_R.
6: $f \leftarrow f^2 \times \frac{\ell_{R,R}}{\nu_R}(D_Q)$.
7: **if** $r_i = 1$, **then**
8: Compute the line function $\ell_{R,P}$.
9: $R \leftarrow R + P$.
10: Compute the line function ν_R.
11: $f \leftarrow f \times \frac{\ell_{R,P}}{\nu_R}(D_Q)$.
12: **end if**
13: **end for**
14: **return** f.

Figure 13.: Miller's Algorithm [35]

9.4 ASTONISHING APPLICATIONS

Now that we have defined and calculated the Weil Pairing we can move on to the last and most important section of this chapter, their cryptological applications. This section will focus on the two major applications: the Tripartite DH Key Exchange Cryptosystem and the MOV Attack which attempts to break the security of the cryptosystems we've spent so many pages defining.

9.4.1 *Tripartite DHP Problem*

This method was developed by Antoine Joux. It allows the Diffie-Hellman Key Exchange to occur between 3 parties relies heavily on the bilinearity property of the Weil Pairing.

Definition 9.4.1 (Tripartite Diffie-Hellman Public Key Exchange Protocol)**.** Suppose 3 parties, Alice, Bob, and Charles wish to establish a secure symmetric key while conducting all communication over insecure channels. The following algorithm creates this key.

Step 1. Alice, Bob, and Charles agree on an elliptic curve E, a finite field $\mathbb{F}_p$, and an integer n. They also decide on points P and Q on E. Thus, $E(\mathbb{F}_p)[n]$, P, and Q are all public information.

Step 2. Alice decides on a secret integer, a and computes $P_A = aP$ and $Q_A = aQ$. Similarly Bob decides on a secret integer b and computes $P_B = bP$ and $Q_B = bQ$ while Charles decides on a secret integer c and computes $P_C = cP$ and $Q_C = cQ$. The values a, b, c are private keys and thus are not public knowledge. Alice, Bob, and Charles transmit the public information $P_A, Q_A, P_B, Q_B, P_C, Q_C$ over insecure channels.

Step 3. Upon receiving P_B and Q_C, Alice computes $K = w_n(P_B, Q_C)^a$. Thus by properties of Weil Pairings,

$$K = w_n(P_B, Q_C)^a = w_n(bP, cQ)^a = (w_n(P, Q)^{bc})^a = w_n(P, Q)^{abc}$$

Similarly, upon receiving P_A and Q_C, Bob computes $K' = w_n(P_A, Q_C)^b$ which equals

$$K' = w_n(P_A, Q_C)^b = w_n(aP, cQ)^b = (w_n(P, Q)^{ac})^b = w_n(P, Q)^{abc}$$

And lastly, upon receiving P_A and Q_B, Charles computes $K'' = w_n(P_A, Q_B)^c$ which equals,

$$K'' = w_n(P_A, Q_B)^c = w_n(aP, bQ)^c = (w_n(P, Q)^{ab})^c = w_n(P, Q)^{abc}$$

Thus, since $K = K' = K''$ Alice, Bob, and Charles now share a secret key K

Remark 9.4.2. The security of this cryptosystem, as with the bipartite Diffie-Hellman Key Exchange, lies with the difficulty of the Elliptic Curve Discrete Logarithm Problem since one would have to solve the ECDLP to discern the secret keys a, b, c. Since this is extremely hard, the secret key that relies on this is secure. However, what happens if solving the ECDLP were to become less difficult? The security of this cryptosystem and others would be threatened.

9.4.2 *MOV Attack*

Following the previous remark, another application of the Weil Pairing is the MOV Attack Algorithm. Developed by Alfred Menezes, Scott Vanstone, and Tatsuaki Okamoto this algorithm reduces the difficulty of the Elliptic Curve Discrete Logarithm Problem 7.2.1, making it quicker to solve and thus easier to obtain the secret keys created within the Bipartite and Tripartite DHP and other cryptosystems. The algorithm reduces the DLP from the elliptic curve $E(\mathbb{F}_p)$ to one in $\mathbb{F}_{p^m}*$ for an integer m. This allows the DLP to be solved much quicker and therefore threatens a cryptosystems security. The algorithm is computed as follows:

Definition 9.4.3 (MOV Attack Algorithm)**.** Given an elliptic curve E over $\mathbb{F}_p$ and points P, Q on $E(\mathbb{F}_P)$, let $|P| = N$ such that $\gcd(N, p) = 1$, the MOV attack finds an integer k such that $Q = kP$ in order to reduce the Discrete Logarithm problem on the curve $E(\mathbb{F}_p)$ to the the Discrete Logarithm problem in $\mathbb{F}^*_{p^m}$ for an integer m.

Complete the following steps for $1 \leq i \leq k$ until $\text{lcm}(d_1, d_2 \cdot d_k) = N$.

Step 1. Select a random point L_i on $E(\mathbb{F}_{p^m})$

Step 2. Let $|L_i| = M_i$ for some integer M_i. Compute M_i

Step 3. Let $d_i = \gcd(M_i, N)$ and $L'_i = \frac{M_i}{d_i} L_i$. Thus we know that $|L'_i| = d_i$ and $L'_i \in E[N]$

Step 4. Compute $w_n(P, L'_i) = \alpha$ and $w_n(Q, L'_i) = \beta$. Since $|L'_i| = d_i$, α_i and β_i are dth roots of unity by definition of the Weil Pairing. By the definition of roots of unity since $\mu_d \subseteq \mathbb{F}_{p^m}*$, $\alpha_i, \beta_i \in \mathbb{F}_{p^m}*$

Step 5. Solve the Discrete Logarithm problem $\beta_i = \alpha_i^{k_i}$ in $\mathbb{F}_{p^m}*$, which returns a $k_i \bmod d_i$.

Continue these steps until $\text{lcm}(d_1, d_2 \cdot d_k) = N$ for all d_i such that $1 \leq i \leq k$. Using these k_i values modulo d_i we can compute k modulo N using the Chinese Remainder Theorem such that $k \equiv k_i \bmod d_i$ for all $1 \leq i \leq k_i$. Thus we have found $k \bmod N$ such that $Q = kP$.

Proof. Let $P, Q \in E(\mathbb{F}_p)$ and consider the MOV algorithm. The first three steps are trivial. Let $\gamma = w_n(R, L'_i)$ such that R is an arbitrary point in $E(\mathbb{F}_{p^m})$ and L'_i is defined as it is in Step 3 of the algorithm. Thus if we consider γ^d for an integer d. It follows that

$$\gamma^d = w_n(R, L'_i)^d = w_n(R, dL'_i) = w_n(R, M_i L_i) = w_n(R, O) = 1$$

Therefore, by definition of roots of unity, γ is a d-th root of unity and thus for all $i, 1 \leq i \leq k$ γ_i is an element of μ_d and thus is an element of $\mathbb{F}_{p^m}$. So we can solve $\gamma_{2i} = \gamma_i^{k_i}$ in $\mathbb{F}_{p^m}$.

Now let $kP = Q$, let $l_i \equiv k \pmod{d_i}$, and consider $w_n(kP, L_i')$. By substitution this equals

$$w_n(kP, L_i') = w_n(Q, L_i) \text{ which by bilinearity implies } w_n(P, L_i')^k = w_n(Q, L_i)$$

Thus $\gamma_{2i} = \gamma_i^k$, which implies that $\gamma_i^{li} \equiv \gamma_{2i} \pmod{d_i}$ since γ is a $d-$th root of unity. Since $\gamma_i^{li} \equiv \gamma_{2i}$ and$\gamma_{2i} = \gamma_i^{k_i}$, $l_i \equiv k_i \pmod{d_i}$. So $l_i \equiv k_i \equiv k \pmod{d_i}$. Therefore when we calculate the correct k_i we find our desired k. □

The ECDLP can be solved much faster in each divisor d_i, than in N and through the Chinese Remainder Theorem we can quickly determine the value its value mod N

Although the Weil Pairing has a strenuous definition, confusingly proven properties, and a tedious calculation its applications are extremely useful in the world of cryptography due to its bilinearity and the reciprocity of its functions and divisors. However its uses both aid those who wish to secure the secrecy and privacy cryptosystems provide and those who wish to gain access to privileged information.

10

MULTIVARIATE CRYPTOSYSTEMS

The RSA cryptosystem is ubiquitous today, it's secure against current attacks, and against current technology. But RSA also has its weaknesses. The security of RSA relies on the fact that we do not have any fast algorithms for factoring large integers. Recent developments pertaining to integer factorization, such as the number field sieve and algorithms based on elliptic curves, have forced RSA to use increasingly larger parameters in order to maintain a necessary level of security. At current standards, a secure RSA scheme requires that N, the product of two distinct primes p and q, has at least 1000 binary digits [36]. Numbers with such large bit sizes, require a huge amounts of calculations, slowing down encryption and decryption and making the entire process inefficient and more costly.

On top of this quantum computers have emerged as a threat to current RSA schemes. In 1999 Peter Shor discovered an algorithm that can be used to factor integers in polynomial time on a quantum computer. So once we have quantum computers that have a large number of quantum bits, the RSA cryptosystem can no longer be considered secure, because N can now be factored with much greater efficiency. This threat should be taken seriously, as the first small-scale but real quantum computer built in 2001, factored 15 into 3 and 5 using Shor's algorithm [37]. This provides the motivation to search for efficient and secure alternatives to RSA.

The RSA public key cryptosystem is based on a singular modular equation. A natural extension of this idea is to consider systems of several modular equations in several variables. In 1988 Tsutomu Mastumoto and Hideki Imai created that very extension of the RSA system [38].

By using a set of quadratic polynomials over a finite field, Mastumoto and Imai created the blue print for what today is called multivariate public key cryptosystems, or MPKCs. As time has progressed many new systems have been introduced, creating a family of multivariate cryptosystems. The main security of the system is backed by the NP-hardness to solve multiple nonlinear equations over a finite field [37]. However as we will soon see dispute this fact not all multivariate systems are secure, and the study of multivariate systems is still very much evolving.

The family of multivariate public key cryptosystems is ever growing and changing. In their current form MPKCs are viable digital signature schemes and based off current research, MPCKs are a potential contender for quantum secure cryptosystems. This chapter explores the general construction of MPKC schemes, the first implementation, the MI or Mastumoto-Imai scheme, the way in which MI was broken, and an alternative to MI.

10.1 FIELD EXTENSIONS AND POLYNOMIALS

Ultimately we want to be able to use polynomials in a versatile way if we are going to build a cryptosystem. Our goal is to create a cryptosystem using polynomials, in order to do this we need an understanding of some of the mathematics behind polynomials.

Recall that a ring, is like a group but with a second operation that distributes over the first, and that a ring stops just short of being a field since it's elements under the second operation needn't meet all the requirements of a group.

We can construct a ring containing polynomials as its elements over a ring as follows.

Definition 10.1.1 (Ring of Polynomials over ring R)**.** Let R be a commutative ring. The set of formal symbols

$$R[x] = \{a_n x^n + a_{n-1} + x^{n-1} + \cdots + a_1 x + a_0 | a_i \in R, \text{n is a nonnegative integer}\}$$

Is called the *ring of polynomials over* R in the indeterminate x. For two elements in $R[x]$

$$p_1(x) = a_n x^n + a_{n-1} + x^{n-1} + \cdots + a_1 x + a_0$$

and

$$p_2(x) = b_x^m + b_{m-1} + x^{m-1} + \cdots + b_1 x + b_0$$

we say $p(x)_1 = p(x)_2$ if and only if $a_i = b_i$ for all nonnegative integers i.

For the following 3 definitions let D be an integral domain with elements $a, b, c \in D$ and the unit $u \in D$.

Definition 10.1.2 (Associates)**.** Elements a and b are associates if $a = ub$, where u is a unit.

Definition 10.1.3 (Irreducibles)**.** a nonzero element a is called irreducible if a is not a unit and, whenever $a = bc$, then b or c is a unit.

Definition 10.1.4 (Primes)**.** a is prime if it is not a unit and $a|bc$ implies $a|b$ or $a|c$.

Definition 10.1.5 (Unique Factorization Domain (UFD))**.** an integral domain D is a *Unique Factorization Domain* if

1. Every nonzero element of D that is not a unit can be written as a product of irreducibles of D
2. The factorization is unique up to associates and the order in which the factors appear.

We are now able to define the Euclidean algorithm for polynomials over fields.

Definition 10.1.6 (Euclidean Algorithm for Polynomials)**.** Let $f(x)$ and $g(x)$ be polynomials with coefficients in an integral domain D. A greatest common divisor of $f(x)$ and $g(x)$ is a polynomial $d(x)$ that divides $f(x)$ and $g(x)$ and such that every common divisor of $f(x)$ and $g(x)$ also divides d.

Since MPKCs are based off quadratic polynomials over a finite field, we first need to understand how to incorporate polynomials into finite fields. This will require introducing some infrastructure from abstract algebra in order to construct our system. To begin we need to be able to explore and utilize fields more deeply. Starting with a concept from linear algebra and applying it to abstract algebra.

Definition 10.1.7 (Vector Space)**.** We call a set V a *vector space* over a field $\mathbb{F}$ if V is an Abelian group under addition and, if for each $a \in \mathbb{F}$ and $v \in V$, there is an element $av \in V$ such that the following conditions hold true for all $a, b \in \mathbb{F}$ and for all $u, v \in V$.

1. $a(v + u) = av + au$
2. $(a + b)v = av + bv$
3. $a(bv) = a(b)v$
4. $1v = v$

Before moving on to the next definition, we can also redefine vector and scalars in a field context.

Remark 10.1.8. The elements or members of a vector space, are called *vectors*, and the elements of a field are called *scalars*. We define the operation between vector spaces and fields as *scalar multiplication*.

Lets now define the analogs of subspaces and Linear dependence.

Definition 10.1.9 (Subspace)**.** Let V be a vector space over a field $\mathbb{F}$ and let U be a subset of V. If U is also a vector space over $\mathbb{F}$ under the operations of V, then U is a subspace of V.

Definition 10.1.10 (Linear Independence)**.** Let S be a set of vectors. S is said to be *linearly dependent* over a field $\mathbb{F}$ if there are vectors $v_1, v_2, \ldots, v_n$ from S and elements $a_1, a_2, \ldots, a_n$ from $\mathbb{F}$, not all zero, such that $a_1v_1 + a_2v_2 + \ldots a_nv_n = 0$. A set of vectors that is not linearly dependent, is called *linearly independent*.

Definition 10.1.11 (Basis)**.** Let V be a vector space over $\mathbb{F}$. A subset U of V is called a basis for V if U is linearly independent over $\mathbb{F}$ and every element of V is a linear combination of elements of U.

Sometimes polynomials in $\mathbb{F}[x]$, do not have roots, or solutions in $\mathbb{F}$, and we need to add extra elements to our field, in order to solve our polynomial. These extra elements come in the form of an Extension Field.

Definition 10.1.12 (Extension Field)**.** A field E is an extension field of a field $\mathbb{F}$, if $\mathbb{F} \subseteq E$ and the operations of $\mathbb{F}$ are those of E restricted to $\mathbb{F}$.

Definition 10.1.13 (Splitting Field)**.** Let E be an extension field of $\mathbb{F}$ and let $f(x) \in \mathbb{F}[x])$, where $\deg f(x) > 1$. We say $f(x)$ splits in E if there are elements $a \in \mathbb{F}$ and $a_1, a_2, \ldots, a_n \in E$ such that

$$f(x) = a(x - a_1)(x - a_2)\ldots(x - a_n)$$

We call E a splitting field for $f(x)$ over $\mathbb{F}$ if the extension field is the field adjoined with the set $\{a_1, a_2, \ldots, a_n\}$.

$$E = \mathbb{F}(a_1, a_2, \ldots, a_n)$$

A splitting field is the smallest extension of field where a polynomial in $\mathbb{F}$ factors.

Extension fields have their own types and structures. A particular important type of extension is an algebraic extension.

Definition 10.1.14 (Algebracic Element)**.** Let E be an extension of $\mathbb{F}$ and let $a \in E$. a is *algebraic over* $\mathbb{F}$ if a is the zero for some nonzero polynomial in $\mathbb{F}[x]$.

Definition 10.1.15 (Algebracic Extension)**.** An extension E is an algebraic extension, if every element in E is algebraic over $\mathbb{F}$

Definition 10.1.16 (Degree of an Extension)**.** Let E be an extension field over $\mathbb{F}$. E has *degree* n over $\mathbb{F}$ and write $[E : F] = n$ if E has dimension n as a vector space over F. If $[E : F]$ is finite, then E is called a finite extension.

All these definitions were adapted from the text book *Contemporary Abstract Algebra* [39].

10.2 THE DISCRETE LOGARITHM PROBLEM FOR MPKCS

Multivariate public-key cryptosystems are schemes that use multivariate polynomial functions as public keys. Thus the security of MPKC is based on the one-wayness of the multivariate polynomial maps. The one-wayness of multivariate polynomial maps resides in the difficulty to find solutions of a system of multivariate polynomial equations [36]. If the multivariate polynomials involved in a MP problem consist only of quadratic polynomials, the problem is called $\mathcal{MQ}$ problem. Let $\mathbb{F}$ be a field, we define a set of polynomials in $\mathbb{F}[x]$ as

$$\begin{aligned}
f_1(x_1, x_2, \ldots, x_n) &= \sum_{1\le i\le j\le n} a_{ij}^{(1)} x_i x_j + \sum_{1\le i\le n} b_i^{(1)} x_i + c^{(1)} \\
f_2(x_1, x_2, \ldots, x_n) &= \sum_{1\le i\le j\le n} a_{ij}^{(2)} x_i x_j + \sum_{1\le i\le n} b_i^{(2)} x_i + c^{(2)} \\
&\vdots \\
f_m(x_1, x_2, \ldots, x_n) &= \sum_{1\le i\le j\le n} a_{ij}^{(m)} x_i x_j + \sum_{1\le i\le n} b_i^{(m)} x_i + c^{(m)}
\end{aligned}$$

where m is the number of equations, n the number of variables, and $a_{ij}^{(1)}, b_i^{(1)}, c^{(1)}, \ldots, a_{ij}^{(m)}, b_i^{(m)}, c^{(m)}$ are all elements in a finite field $\mathbb{F}$.

We can think of this problem as given m multivariate quadratic polynomials $f_1(x), \ldots, f_m(x)$, find a vector $\overline{X} = (x_1, \ldots x_n)$ such that

$$f_1(\overline{X}) = \ldots = f_m(\overline{X}) = 0.$$

Complexity theory tells us this problem has been proven to be NP hard.

10.3 GENERAL CONSTRUCTION

The general construction is as follows. Let k be a finite field, the cipher is given as a function $\overline{F}$, where $\overline{F} : k^n \to k^m$, for vector spaces k^n and k^m. We are sending vectors from k^n to k^m as

$$\overline{F}(x_1, \ldots, x_n) = (\overline{f}_1, \ldots, \overline{f}_m).$$

Each $\overline{f_i}$ is a polynomial in $k[x_1, \ldots, x_n]$. General construction begins by defining the map F from k^n to k^m. This map must meet the following two properties.

1. $F(x_1, \ldots, x_n) = (f_1, \ldots, f_m)$, where $f_i \in k[x_1, \ldots, x_n]$
2. For any equation of the form $F(x_1, \ldots, x_n) = (y'_1, \ldots, y'_m)$, we can easily find the pre-image $F^*(y'_1, \ldots, y'_m)$ for a given map F.

Remark 10.3.1. Note that finding F^* only means we have found the pre-image of $(y'_1, \ldots, y'_m)$, this does not require that our function F has an inverse.

And to properly define linear transformations recall the definition.

Definition 10.3.2 (Linear Transformation)**.** Let V and W be vector spaces over the same field $\mathbb{K}$. A function $f : V \to W$ is said to be a linear map if for any two vectors $\mathbf{u}, \mathbf{v} \in V$ and any scalar $c \in \mathbb{K}$ the following two conditions are satisfied:

$$f(\mathbf{u} + \mathbf{v}) = f(\mathbf{u}) + f(\mathbf{v})$$

and

$$f(c\mathbf{u}) = cf(\mathbf{u})$$

The map F serves as our trap door, enabling us to easily decrypt a message. F is often called the central map, and we want to hide it from public view. We hide F in $\overline{F}$ by composing it with two linear transformations, such that

$$\overline{F} = S \circ F \circ T$$

Where S is a randomly chosen invertible affine transformation from k^m to k^m, and similarly T is a transformation from k^n to k^n. This construction forms the basic cipher see figure 10.3.

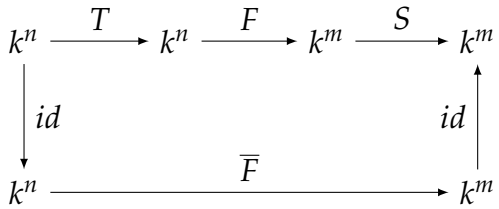


Figure 14.: This figure shows the basic function digram for the general version of all MPKCs [37].

In this construction the private information is S, T and depending on the system F, and the public information is $\overline{F}$. We can now describe the general encryption and decryption process.

Encryption Alice chooses a message vector $X' = (x'_1, \ldots, x'_n)$ and computes $\overline{F}(X')$ to produce a $Y' = (y'_1, \ldots, y'_m)$.

Decryption Bob has access to the secret information, so they can decrypt the message. First bob computes $a = S^{-1}(Y')$, then $b = F^*(a)$ and finally $T^{-1}(b)$ to produce the original message.

Alice and Bob verify the decryption process by adding a digital signature. The much like real signatures, digital signature are legal forms of verification and are used to ensure that people are indeed who the claim to be. In this case Bob can sign a message by finding a solution set to $F(X) = Y'$, denoted $X' = (x'_1, \ldots, x'_n)$. Alice can verify the signature by checking if

$$F(X) = Y'.$$

Once this process has been completed Alice and Bob now know who they are sending their sensitive information to.

10.3.1 *Quadratic Constructions*

In many multivariate schemes, the public key is given as a set of polynomials of total degree two. Thus the quadratic polynomials have the form

$$\sum_{i \leq j} a_{ij} x_i x_j = \sum b_i x_i + c$$

The public polynomials have this form for two main reasons. Those being for efficiency and for security considerations. For a fixed n, the number of possible terms in a polynomial of degree d in n variables is expressed using binomial theorem

$$\binom{n+d}{d} = \frac{(n+d)!}{(n+d-d)!d!} = \frac{(n+d)!}{n!d!}$$

As d increases, the total number of terms grows rapidly. If d becomes large then the public key size will be to great to be computed and stored efficiently. Furthermore any system of large total degree polynomials can always be converted into a larger system with total degree two. Thus the increase in security gained by increasing the total degree is not a good trade off due to the loss in efficiency. These considerations have lead to the majority of multivariate schemes to be bipolar and quadratic. We will now introduce the first system built using these ideas.

10.4 THE MATSUMOTO-IMAI CRYPTOSYSTEM

The first multivariate cryptosystem created, denoted C^* or MI, was built by Japanese mathematicians Matsumoto and Imai. Their system has unique in that it utilized both

the vector space and hidden field structure of k^n, where k is a finite field. Rather than finding invertible functions for the vector space k^n directly, they searched for invertible maps over K, an n degree extension of the field k. This map could then be transformed into a map over k^n [38].

The MI system gained a lot of attention in Japan, even so far as becoming a candidate for security standards to be used by the Japanese government. Unfortunately MI was successfully broken by Jacques Patarin in 1995 using what is now called a linearizion attack. This particular method take advantage of the hidden algebraic structures within k[36].

It would be easy to assume this is the end of the Mi system, or ever this type of encryption. This is not the case, in fact the opposite is true. The MI system became seen as a fundamental breakthrough as it brought through mathematical ideas into cryptology, and consequentiality has been explored, expanding the field of cryptology [36].

First we must construct a public key similarly to how we created one in the perviousness section. Let K be a finite field of characteristic two. Let $g(x)$ be an irreducible polynomial in $k[x]$, with degree n. Define the extension field K as the factor field $K = k[x]/g(x)$. Define $\phi : K \to k^n$ as a k-linear isomorphism, mapping elements from K to the vector space k^n. The function ϕ sends polynomials in K to a vector of their coefficients in k^n, such that for any given polynomial in K we

$$\phi(a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}) = (a_0, a_1, \ldots, a_{n-1})$$

Using the function ϕ, we can embed the subfield k into our vector space k^n. We treat all elements in k as constant polynomials and the function works as follows

$$\phi(a) = (a, 0, \ldots, 0), \quad \forall \quad a \in k$$

Now we choose a θ such that $0 < \theta < n$, and most importantly

$$\gcd(q^\theta + 1, q^n - 1) = 1$$

This is used to define the function $\widetilde{F}$ over K as

$$\widetilde{F}(X) = X^{q^\theta + 1}.$$

The way we have defined θ ensures that $\widetilde{F}$ has an inverse. If t is an integer such that $t(1 + q^\theta) \equiv 1 mod(q^n - 1)$, Then we can compute $\widetilde{F}^{-1}$ simply as

$$\widetilde{F}^{-1}(X) = X^t.$$

We now define the central map as the central map F over k^n as $F = \phi \circ \widetilde{F} \circ \phi^{-1}$, where

$$F(x_1, \ldots, x_n) = \phi \circ \widetilde{F} \circ \phi^{-1}(x_1, \ldots, x_n) = (\overline{f_1}, \ldots, \overline{f_n}),$$

Where $\overline{f_1}, \ldots, \overline{f_n} \in k[x_1, \ldots, x_n]$. To embed this map in the same fashion as the general construction. We select a S,T to again be linear transformations over k^n. See the figure below for a diagram displaying the MI construction.

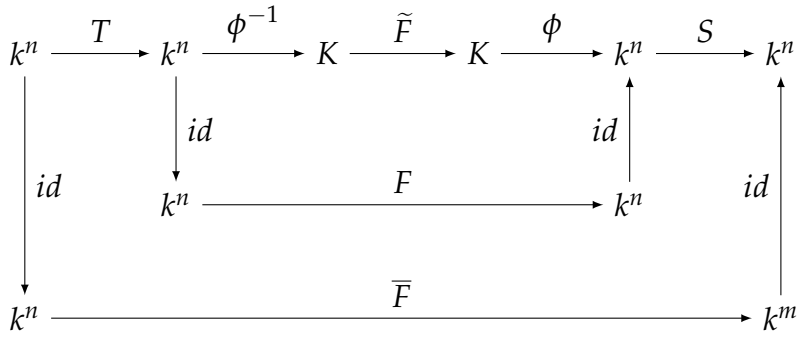


Figure 15.: The construction of the MI cryptosystem [37].

The public key includes the following information:

The field k

The n polynomials $\overline{f_1}, \ldots, \overline{f_n} \in k[x_1, \ldots, x_n]$

The private key contains the following information:

The linear transformations S, T

The parameter θ

Remark 10.4.1. We do not have to keep the parameter θ private. Since there are fewer than n choices of θ, it is likely an attacker could figure out what θ is given that n is relatively small.

We now turn towards the encryption, and decryption process. Alice or anyone for that matter can encrypt a plain text message $x'_1, \ldots, x'_n$, to produce a ciphertext $y'_1, \ldots, y'_n$. This is accomplished with the public map $\overline{F}$. A given element in the cipher text is described as $y'_i = \overline{f}_i(x'_1, \ldots, x'_n)$, for $i = 1, \ldots, n$.

Bob decrypts the message by computing $\overline{F}^{-1}(y'_1, \ldots, y'_n)$. Which begins first by calculating the inverses to S and T,

$$\overline{F}^{-1}(y'_1, \ldots, y'_n) = S^{-1} \circ F^{-1} \circ T^{-1}$$

Bob then computes the inverse of $\widetilde{F}$,

$$\overline{F}^{-1}(y'_1, \ldots, y'_n) = S^{-1} \circ \phi^{-1} \circ \widetilde{F}^{-1} \circ \phi \circ T^{-1}$$

Generally the components of $\overline{F}^{-1}$ (polynomials) have very high degree, so the actual implemented practice of decoding the message is as follows.

1 First Bob calculates $(z'_1, \ldots, z'_n) = T^{-1}(y'_1, \ldots,)$

2 Then they find $(\overline{z}_1, \ldots, \overline{z}_n) = \phi \circ \widetilde{F} \circ \phi^{-1}$

3 To produce the original message the compute $(x'_1, \ldots, x'_n) = S^{-1}(\overline{z}_1, \ldots, \overline{z}_n)$

The process for both Alice and Bob is outlined in the following figure.

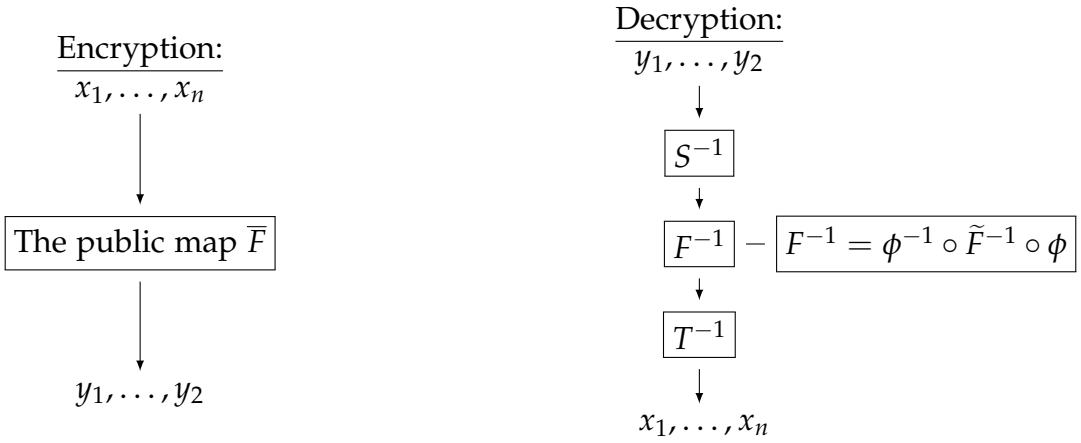


Figure 16.: The encryption and decryption process for the MI system [37].

Therefore if the system is secure the only person or people who can decrypt the information successfully are those who have access to the secret key. The following example is what is called a toy example. It illustrates the mechanics of the MI system for a very small finite field, so small that it would not be suitable for actual use.

Example 10.4.2. We first must pick our field, that we are working in. Let k represent our finite field , such that $|k| = 2^2 = 4$. We represent the field with four elements as $k = \{0, 1, \alpha, \alpha^2\}$, where 0 is the additive identity, and 1 is the multiplicative identity. The two tables below are Cayley diagrams showing how the operations act over all the elements.

$+$	0	1	α	α^2
0	0	1	α	α^2
1	1	0	α^2	α
α	α	α^2	0	1
α^2	α^2	α	1	0

$\times$	0	1	α	α^2
0	0	0	0	0
1	0	1	α	α^2
α	0	α	α^2	1
α^2	0	α^2	1	α

Figure 17.: The operations of multiplication and addition within the finite field K

Notice in the second table that the element α generates the whole group. We now have to find an irreducible polynomial in $k[x]$. we find that $g(x) = x^3 + x + 1$ is irreducible in $k[x]$. Notice that the degree of $g(x)$ is 3, so $n = 3$. Recall that we must find a θ such that

$\gcd(q^\theta + 1, q^n - 1) = 1$. In this case θ can either be 1 or 2. we will set $\theta = 2$, and we see that it satisfy the previous condition

$$\gcd(q^\theta + 1, q^n - 1) = \gcd(17, 63) = 1.$$

We now create our extension field K as $k[x]/g(x)$. Next we must compute $\widetilde{F}$ and its inverse. We find that

$$\widetilde{F} = X^{q^\theta + 1} = X^{17}$$

We find the inverse $\widetilde{F}^{-1}$ by solving the equation

$$t * 17 \equiv 1 \pmod{63}$$

We find that $t \equiv 1 * 17^{-1} \pmod{63}$, and thus we find that $t \equiv 26 \pmod{63}$, so $t = 26$, and we have found the inverse to be

$$\widetilde{F}^{-1} = X^t = X^{26}.$$

The linear transforms S and T have to invertible 3×3 matrices. Recall in our construction of the MI scheme that S, T where maps from k^n to k^n. In this example S and T are defined as

$$S(x_1, x_2, x_3) = \begin{pmatrix} \alpha^2 & \alpha & \alpha \\ \alpha & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} + \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ \alpha \end{pmatrix}$$

and

$$T(x_1, x_2, x_3) = \begin{pmatrix} 1 & 0 & \alpha \\ 0 & 1 & \alpha \\ 1 & \alpha & 0 \end{pmatrix} + \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} \alpha \\ \alpha^2 \\ \alpha^2 \end{pmatrix}$$

To derive the public key polynomials in terms of the plaintext message variables x_1, x_2, x_3. We start by calculating $\phi^{-1} \circ T(x_1, x_2, x_3)$. Recall that to produce the public key we want to compute $\overline{F}(X)$.

Also recall that ϕ sends coefficient of polynomials in K to the vector space k^n, and the inverse sends the components of vectors in k^n to polynomials in K. We now compute $\phi^{-1} \circ T(x_1, x_2, x_3)$. The output of ϕ^{-1} is the polynomial

$$(\alpha + x_1 + \alpha x_3) + (\alpha^2 + x_2 + \alpha x_3)x + (\alpha^2 + x_1 + \alpha x_2)x^2.$$

Lets denote this polynomial as X, moving along the chain we now compute $\widetilde{F}(X) = X^{16}$. At first the polynomial to the power 16 may seem frightening, but recall that k has characteristic two, and therefore cannot have polynomials of degree greater than 2. Thus $\tilde{F}(x)$ is equal to

$$\begin{gathered} 1 + \alpha^2 x_1 + \alpha x_2 + x3 + x_1 x_2 + \alpha^2 x_2 x_3 \\ +(\alpha + \alpha x_1 + x_2 + \alpha^2 x_3 + x_1^2 + \alpha^2 x_1 x_2 + x_2^2 + x_2 x_3)x \\ +(\alpha^2 + \alpha^2 x_1 + \alpha x_2 + \alpha x_3 + x_1^2 + x_1 x_2 + \alpha x_1 x_3 + \alpha^2 x_2^2 + \alpha x_2 x_3 + \alpha^2 x_3^2)x^2 \end{gathered}$$

The final component to compute is $S \circ \phi(X)$, to obtain the public key polynomials

$$\overline{f}_1(x_1,x_2,x_3) = 1 + x_3 + \alpha x_1 x_3 + \alpha^2 x_2^2 + \alpha^2 x_2 x_3 + x_3^2$$
$$\overline{f}_2(x_1,x_2,x_3) = 1 + \alpha^2 x_1 + \alpha x_2 + x_3 + x_1^2 + x_1 x_2 + \alpha^2 x_1 x_3 + x_2^2$$
$$\overline{f}_3(x_1,x_2,x_3) = \alpha^2 x_3 + x_1^2 + \alpha^2 x_2^2 + x_2 x_3 + \alpha^2 x_3^2$$

These polynomials are now used to encrypt plain text messages. Suppose that the plain text message Alice wishes to send is $(x_1', x_2', x_3') = (1, \alpha, \alpha^2)$, then we compute

$$y_1' = \overline{f}_1(1, \alpha, \alpha^2)$$

evaluating $\overline{f}_1$ we see

$$\overline{f}_1(1, \alpha, \alpha^2) = 1 + (\alpha^2) + \alpha(1 * \alpha^2) + \alpha^2(\alpha)^2 + \alpha^2(\alpha * \alpha^2) + (\alpha^2)^2$$

Applying the operations we find that

$$y_1' = \overline{f}_1 = 0,$$

and repeating this process for y_2' and y_3' we see that

$$y_1' = \overline{f}_1 = 0$$
$$y_2' = \overline{f}_2 = 1$$
$$y_3' = \overline{f}_3 = 1$$

to get the ciphertext $(0, 0, 1)$

Bob, knowing S^{-1}, T^{-1} and $\widetilde{F}^{-1}$ begins the decryption process by finding $S^{-1}(0, 0, 1)$. Where

$$S^{-1} = \begin{pmatrix} \alpha^2 & 1 & 1 \\ 1 & \alpha^2 & \alpha \\ \alpha^2 & 1 & 0 \end{pmatrix} + \begin{pmatrix} y_1 - 0 \\ y_2 - 1 \\ y_3 - \alpha \end{pmatrix}$$

and Bob finds that $S^{-1}(0, 0, 1) = \begin{pmatrix} \alpha \\ \alpha \\ 1 \end{pmatrix}$. Letting $X = \alpha + \alpha x + x^2$, Bob finds that

$$\widetilde{F}(X) = X^{26} = \alpha + x^2,$$

which is can easily be computed by the binary method, or the square and multiply method. Continuing with the decryption process, Bob sets $(\overline{z}_1, \overline{z}_2, \overline{z}_3) = (\alpha, 0, 1)$, and using T^{-1}, they can find the oringal plain text.

Bob finds

$$T^{-1}(\alpha, 0, 1) = \begin{pmatrix} 1 \\ \alpha \\ \alpha^2 \end{pmatrix},$$

Which is the original plain text.

10.5 LINEARIZATION EQUATION ATTACK

The base *MI* system was successfully broken in 1995, by Jacques Patarin using an algebraic attack. Specifically Patarin used linearization equations. We start by defining the notion of a linearization equation.

Definition 10.5.1 (Linearization Equation)**.** Let $\mathcal{G} = \{q_1, \ldots, g_m\}$ be any set of m polynomials in $k[x_1, \ldots, x_n]$. A linearization equation for $\mathcal{G}$ is any polynomial in $k[x_1, \ldots, x_n, y_1, \ldots, y_n]$ of the form

$$\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}x_i y_j + \sum_{i=1}^{n} b_i x_i + \sum_{j=1}^{m} c_j y_j + d,$$

such that we obtain the zero function upon substituting the polynomial g_i for y_i, for $j = 1, \ldots, n$. The linearization equation is any equation of the form

$$\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}x_i g_j(x_1, \ldots, x_n) + \sum_{i=1}^{n} b_i x_i + \sum_{j=1}^{m} c_j g_j(x_1, \ldots, x_n) + d = 0$$

Which holds for all $x'_1, \ldots, x'_2$.

Using linearization equations allows us to find what are called linear equations. These will be used in the attack on the *MI* system, and We can build a system of linear equations to solve for the plaintext.

Definition 10.5.2 (Linear Equation)**.** let k be a field, a linear equation is an equation that may be put in the form

$$a_1 x_1 + \cdots + a_n x_n + c = 0,$$

where $x_1, \ldots, x_n$ are the variables or unknowns, and $c, a_1, \ldots, a_n$ are elements in k. In other words, a linear equation is obtained by equating to zero a polynomial of degree 1.

The set of linearization equations, denoted $\overline{\mathcal{L}}$, form a k-vector space, this is called the *linearization equation space* of $\mathcal{G}$.

Patarin noticed that the linearization equation space of the cipher $\overline{F}$ in the *MI*, could be used to attack the cryptosystem. The general attack is as follows, given a set of components $\{\overline{f}_1, \ldots, \overline{f}_n\}$, that make up $\overline{F}$, and suppose we find a linearization equation in given form in definition 10.5.1.For some given ciphertext $(y'_1, \ldots, y'_n)$, we substitute y'_i in for each $\overline{f}_i$, in the hopes that it will produce a linear equation in the variables $x_1, \ldots, x_n$, whose solution set contains the plaintext.

With enough linearization equations, we can hope to produce enough linear equations such that the resulting system has the plaintext as its unique solution. Even if we cannot

find directly the plaintext from these linear equations for a given ciphertext, as long as the LEs can produce enough linearly independent linear equations for the corresponding plaintext, these linear equations can then be plugged into the quadratic public equations derived from the public key and the ciphertext to reduce the number of variables and make it much easier to solve it [40].

In order to see if this attack will be possible, we must determine the number of linearly independent linear equations. We can derive from the linearization space of the components $\overline{F}$. Using the *MI* cryptosystem, the following theorem tells us the lower bound on the number of linearly independent linear equations.

Theorem 10.5.3 (The lower bound for the number of LEs of $\overline{F}$)**.** *let $\{\overline{f}_1, \ldots, \overline{f}_n\}$ be the public key for an implementation of the MI cryptosystem. Suppose that we have an arbitrary cipher text $Y' = (y'_1, \ldots, y'_n) \in k^n$, let $\overline{\mathcal{L}}$ be linearization space of $\{\overline{f}_1, \ldots, \overline{f}_n\}$. If $\overline{\mathcal{L}}_{Y'}$ is the space of equations that are derived when substituting in y'_i for y_i, for $i = 1, \ldots, n$, in each equation of $\overline{\mathcal{L}}$, then the number of linearly independent linear equations in $\overline{\mathcal{L}}_{Y'}$ is at least*

$$n - gcd(n, \theta) \geq \frac{2n}{3}.$$

This thereon tells us we can construct at least $\frac{2n}{3}$ linearization equations. In order to prove this theorem, we need the following lemmas.

Lemma 10.5.4. *Let the cipher $\overline{F} = S \circ F \circ T$ be same as in the construction of the MI cryptosystem. Let $\mathcal{L}$ be the linearization space of $\{f_1, \ldots, f_n\}$ and let $\overline{\mathcal{L}}$ be the linearization space of $\{\overline{f_1}, \ldots, \overline{f_n}\}$. Then the two vector spaces have the same dimension,*

$$\dim_k \mathcal{L} = \dim_k \overline{\mathcal{L}}.$$

Proof. Let the cipher $\overline{F} = S \circ F \circ T$ be same as in the construction of the *MI* cryptosystem. Let $\mathcal{L}$ be the linearization space of $\{f_1, \ldots, f_n\}$ and let $\overline{\mathcal{L}}$ be the linearization space of $\{\overline{f_1}, \ldots, \overline{f_n}\}$. Now Suppose that the transformation T, is an identity matrix. Thus we send plain text through $\overline{F}$ we get the following result

$$\overline{f}_i(x_1, \ldots, x_n) = \sum_{i=1}^{n} \alpha_{ij} f_i(x_1, \ldots, x_n) + \beta_j,$$

Where $\alpha, \beta \in k$. Then applying definition 10.5.1, we find that

$$0 = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i \overline{f}_j + \sum_{i=1}^{n} b_i x_i + \sum_{j=1}^{n} c_j \overline{f}_j + d$$

and using substitution we replace $\overline{f}$, resulting in

$$0 = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i\Big(\sum_{l=1}^{n}\alpha_{jl}f_l + \beta_j\Big) + \sum_{i=1}^{n} b_i x_i + \sum_{j=1}^{n} c_j\Big(\sum_{l=1}^{n}\alpha_{jl}f_l + \beta_j\Big) + d$$

Simplifying we find that

$$0 = \sum_{i=1}^{n}\sum_{j=1}^{n} a'_{ij}x_i f_j + \sum_{i=1}^{n} b'_i x_i + \sum_{j=1}^{n} c'_j f_j + d'$$

Which by definition is a linearization equation for the set $f_1, \ldots, f_n$. By looking at $F = S^{-1} \circ \overline{F}$ and starting with a linearization equation for $f_1, \ldots, f_n$, we can derive a linearization equation for $\{\overline{f_1}, \ldots, \overline{f_n}\}$. From this bijection we see that the dimension of the linearization equation spaces for F and $S \circ F$ are the same. Now suppose that S is the identity, and let

$$\overline{x}_j = \sum_{i=1}^{n} \alpha_{ij}x_i + \beta_j$$

therefore it follows that $\overline{f}_i(x_1, \ldots, x_n) = f_i(\overline{x}_1, \ldots, \overline{x}_n)$. Then the linearization equation

$$0 = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i f_j(x_1, \ldots, x_n) + \sum_{i=1}^{n} b_i x_i + \sum_{j=1}^{n} c_j f_j(x_1, \ldots, x_n) + d,$$

Which can be expressed as

$$0 = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\overline{x}_i f_j(\overline{x}_1, \ldots, \overline{x}_n) + \sum_{i=1}^{n} b_i \overline{x}_i + \sum_{j=1}^{n} c_j f_j(\overline{x}_1, \ldots, \overline{x}_n) + d,$$

since the inevitable change of variables amounts to a permutation on k^n. And again by substitution we find that

$$0 = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\overline{x}_i \overline{f}_j(x_1, \ldots, x_n) + \sum_{i=1}^{n} b_i \overline{x}_i + \sum_{j=1}^{n} c_j \overline{f}_j(x_1, \ldots, x_n) + d,$$

which again reduces to

$$0 = \sum_{i=1}^{n}\sum_{j=1}^{n} a'_{ij}\overline{x}_i \overline{f}_j + \sum_{i=1}^{n} b'_i \overline{x}_i + \sum_{j=1}^{n} c'_j \overline{f}_j + d',$$

which is a linearization equation for $\{\overline{f}_1, \ldots, \overline{f}_n\}$. Looking at $F = \overline{F} \circ T^{-1}$, we start with an LE for $\{\overline{f}_1, \ldots, \overline{f}_n\}$, and we can derive a linearization equation for $f_1, \ldots, f_n$. From this bijection we can see that the dimensions of the LEs for F and $F \circ T$ are the same. Therefore $dim_k \mathcal{L} = dim_k \overline{\mathcal{L}}$. □

Lemma 10.5.4 showed us that the dimension of the linearization equation space of the polynomial components in F, is equal to the dimension of the linearization equation

space of the polynomial components in $\overline{F}$. The second lemma required in the proof of theorem 10.5.3, shows that dimension of the linearization equation space over the ciphertext Y' it equal to the dimension of linearization equation space of the inverse transformation of $S^{-1}(Y')$.

Lemma 10.5.5. *Let $\mathcal{L}$ and $\overline{\mathcal{L}}$ be defined in the same as in 10.5.4. Let $Y' = (y'_1, \ldots, y'_n) \in k^n$ be an arbitrary ciphertext, and let $Z = S^{-1}(Y') = (z_1, \ldots, z_n)$. Let $\mathcal{L}_z$ be the space of linear equations that arise from substituting in z_i for y_i in each linearization equation in $\mathcal{L}$, and let $\overline{\mathcal{L}}_{Y'}$ be the space of linear equations that arise from substituting in zy'_i for y_i in each linearization equation in $\overline{\mathcal{L}}$. Then these two vector spaces have the same dimension,*

$$\dim_k \mathcal{L}_Z = \dim_k \overline{\mathcal{L}}_{Y'}.$$

Proof. Let $\mathcal{L}$ and $\overline{\mathcal{L}}$ be defined in the same as in lemma 10.5.4. Let $Y' = (y'_1, \ldots, y'_n) \in k^n$ be an arbitrary ciphertext, and let $Z = S^{-1}(Y') = (z_1, \ldots, z_n)$. Let $\mathcal{L}_z$ be the space of linear equations that arise from substituting in z_i for y_i in each linearization equation in $\mathcal{L}$, and let $\overline{\mathcal{L}}_{Y'}$ be the space of linear equations that arise from substituting in zy'_i for y_i in each linearization equation in $\overline{\mathcal{L}}$.The same bijection between $\mathcal{L}$ and $\overline{\mathcal{L}}$ in the proof lemma 10.5.4, induces a bijection between $\mathcal{L}_Z$ and $\overline{\mathcal{L}}_{Y'}$. Therefore using the same argument in the last lemma applies here and

$$dim_k \mathcal{L}_Z = dim_k \overline{\mathcal{L}}_{Y'}.$$

□

Lemma 10.5.6. *For any two positive integers a, b we have*

$$\gcd(q^a - 1, q^b - 1) = q^{\gcd(a,b)} - 1.$$

Proof. Let q, a, b be integers. let the $\gcd(q^a - 1, q^b - 1) = d$ and let $\gcd(a, b) = c$, where $d, c \in \mathbb{Z}$. By properties of the gcd we know that $q^a \equiv 1 \pmod{d}$, and $q^b \equiv 1 \pmod{d}$. Another property allows us to express the $\gcd(a, b)$ as linear combination of a and b, so that

$$c = ax + by,$$

where $x, y \in \mathbb{Z}$. Raising q to the power of c, we find that $q^c = q^{ax+by} = q^{ax}q^{bx}$. Taking the expression modulo d we find that $q^{ax}q^{bx} \equiv 1^x 1^y \equiv 1 \pmod{d}$, since as stated before we have $q^a \equiv 1 \pmod{d}$, and $q^b \equiv 1 \pmod{d}$. Using the definition of modular arithmetic, we find that $d | q^c - 1$, and it is also clear that $q^{\gcd(a,b)} - 1 | d$, thus $\gcd(q^a - 1, q^b - 1) = q^{\gcd(a,b)} - 1$. □

We now move to outline the proof of theorem 10.5.3. Suppose we have construed an *MI* cryptosystem. Recall in the construction of the *MI* scheme we have the $\widetilde{F} =$

$\phi \circ F \circ \phi^{-1}$, and the the function ϕ, utilized an extension field $K = k[x]/g(x)$, where $g(x)$ is irreducible in $k[x]$. consider two polynomials in $X, Y \in K$ such that

$$Y = \widetilde{F}(X)$$

and from the definition of $\widetilde{F}(X)$ in the last section, we know that

$$Y = \widetilde{F}(X) = X^{q^\theta+1}$$

Raising both sides of the equation to $q^\theta - 1$ we find that

$$Y^{q^{\theta-1}} = (X^{q^\theta-1})^{q^\theta-1}$$

Which reduces to

$$Y^{q^{\theta-1}} = X^{q^{2\theta}-1}$$

We can multiply both sides by XY, finding that

$$XY^{q^\theta} = X^{q^{2\theta}}Y$$

or that

$$XY^{q^\theta} = X^{q^{2\theta}}Y$$

Using the above equation, we define $\widetilde{R}(X, Y) \in K[X, Y]$, where

$$\widetilde{R}(X, Y) = XY^{q^\theta} - X^{q^{2\theta}}Y,$$

and from this point on the proof goes beyond the scope of the chapter. The remaining compnets of the proof can be found in *Multivariate Public Key Cryptosystems*. Suffice to say, $\widetilde{R}$ is used to find the n number of linearization equations of the polynomial compents of F. With this information it is possible to determine total number of solutions to $\widetilde{R}$ which is around $\gcd(q^\theta - 1, q^n - 1)$. Then applying lemma 10.5.6 and lemma 10.5.5, it is possible to show that the minimum number of linearization equations for a given MI system is

$$n - gcd(\theta, n) \geq \frac{2n}{3}$$

If the $\gcd(\theta, n) = 1$, then it an MI scheme can be broken, through linearization equations alone. Generally speaking this shows that the system is not very secure, because there are least $\frac{2n}{3}$ linear equations for a given ciphertext, which are satisfied by the plaintext. This is analogous to leaking $2/3$ of the information, and these equations can be used to eliminate $2/3$ of the variables from public quadratic equations, derived from the cipher text and public key, making the system much easier to solve[37].

A question we have not yet considered is how linearization equations are actually obtained. One aproach to take when generating these equations, is to use plaintext-ciphertext pairs. This is easily done, since the encryption method is public, we can create as many pairs as we would like.

10.5.1 *Plaintext-Ciphertext Pairs*

As stated earlier using the public key, we generate plaintext-cipher text pairs. Each pair is given as $\overline{F}(x'_1, \ldots, x'_n) = (y'_1, \ldots, y'_n)$, we can substitute in these pairs into the general linearization equation

$$\sum a_{ij} x_i y_i + \sum b_i x_i + \sum c_j y_j d = 0,$$

and we produce a linear equation of $(n+1)^2$ unknowns in $a_{ij}, b_i, c_j \in K$. Therefore if we generate $(n+1)^2$ plaintext-ciphertext pairs, then we should be able to solve the system of unknown coefficients.

Using example 10.4.2 as our encryption scheme, we will now attack the MI cryptosystem using linearization equations.

Example 10.5.7. The plaintext is given by $n = 5$ variables so $(x_1, x_2, x_3, x_4, x_5) \in k^5$. To make the public key more compact we introduce $x_0 = 1$, so that we can write the key as a sum of quadratic terms. Using the row vector $X = (x_0, x_1, x_2, x_3, x_4, x_5)$, the public key is expressed with the following matrices, where X^T, is the transpose of X.

$$y_1 = X \begin{pmatrix} 0 & 0 & \alpha & 1 & 1 & 1 \\ 0 & \alpha & \alpha & \alpha^2 & \alpha & 0 \\ 0 & 0 & 1 & \alpha^2 & 0 & \alpha \\ 0 & 0 & 0 & \alpha^2 & \alpha & \alpha \\ 0 & 0 & 0 & 0 & \alpha & \alpha^2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} X^T,$$

$$y_2 = X \begin{pmatrix} \alpha & 0 & 0 & 0 & \alpha^2 & 1 \\ 0 & \alpha & 0 & \alpha^2 & \alpha^2 & 1 \\ 0 & 0 & 1 & \alpha^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & 0 & 1 & \alpha^2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} X^T,$$

$$y_3 = X \begin{pmatrix} 1 & \alpha^2 & \alpha & \alpha^2 & 1 & \alpha^2 \\ 0 & \alpha^2 & 0 & 0 & 1 & \alpha^2 \\ 0 & 0 & \alpha^2 & 0 & 1 & 1 \\ 0 & 0 & 0 & \alpha^2 & \alpha & \alpha \\ 0 & 0 & 0 & 0 & 0 & \alpha^2 \\ 0 & 0 & 0 & 0 & 0 & \alpha^2 \end{pmatrix} X^T,$$

$$y_4 = X \begin{pmatrix} 1 & \alpha^2 & 1 & \alpha^2 & 0 & 0 \\ 0 & \alpha^2 & \alpha & \alpha^2 & \alpha^2 & \alpha^2 \\ 0 & 0 & 1 & 0 & \alpha^2 & \alpha \\ 0 & 0 & 0 & 1 & \alpha^2 & \alpha \\ 0 & 0 & 0 & 0 & 0 & \alpha^2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} X^T,$$

$$y_5 = X \begin{pmatrix} \alpha^2 & \alpha^2 & \alpha & 1 & 1 & \alpha^2 \\ 0 & 0 & 0 & \alpha^2 & \alpha^2 & 0 \\ 0 & 0 & \alpha^2 & \alpha & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & \alpha^2 \\ 0 & 0 & 0 & 0 & \alpha & \alpha \\ 0 & 0 & 0 & 0 & 0 & \alpha^2 \end{pmatrix} X^T$$

Lets assume a plaintext message produced the following ciphertext, $(1, 0, 0, 0, 1)$. We can now recover the original plaintext, with linearization equations. We now introduce y_0 which is a 6×6 identity matrix, this allows us to express the public key as the vector $Y = (y_0, y_1, y_2, y_3, y_4, y_5)$. Using the equation for linearization equations in definition 10.5.1, where we set $m = 5$, and $n = 5$, we can now write them in matrix form

$$xAy^t = 0 \tag{1}$$

where A is a 6×6 matrix with unknown coefficients $A_{i,j}$, $i, j = 0 \ldots 5$. To denote our system of equations, we will employ array notation from computer science, where an element $A_{i,j}$ in a matrix corresponds to an index of an array $A[(m+1)i + j]$, so generally speaking we find that

$$A_{i,j} = A[(m+1)i + j]$$

So for our unknown filed elements we express the coefficients of the linearization as follows,

$$a_{ij} = A[(m+1)i + j]$$
$$b_i = A[(m+1)i]$$
$$c_j = A[j]$$
$$d = A[0],$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. We can substitute in our public into equation 1, producing a homogeneous polynomial with 56 terms, we obtain a system of equations with 36 unknowns $A[0]$ through $A[35]$

$$A[0] = \alpha^2 A[33] + \alpha^2 A[32] + A[34] + A[35]$$

$$A[1] = A[29] + \alpha^2 A[32] + A[33] + A[34] + A[35]$$
$$A[2] = \alpha A[29] + A[32] + A[35]$$
$$A[3] = A[29] + \alpha A[32] + \alpha^2 A[33] + \alpha^2 A[34] + \alpha A[35] +$$

$$A[4] = \alpha A[32] + \alpha^2 A[33] + \alpha A[34] + A[35]$$
$$A[5] = \alpha A[32]$$
$$\vdots$$
$$A[31] = A[31] + \alpha^2 A[33] + A[34] + \alpha^2 A[35]$$

Although not all equations are picture here, every equation has is comprised of the free variables $A[29], A[32], A[33], A[34], A[35]$. These values and the given cipher text are subsisted back into equation 1, and the coefficients of the free parameters are set to zero to produce the following equations for the plaintext,

$$\alpha x_1 + x_2 + x_4 + \alpha^2 = 0,$$
$$\alpha^2 x_2 + x_3 + \alpha x_4 + x_5 + \alpha = 0,$$
$$\alpha x_1 + x_2 + \alpha^2 x_3 + x_5 + 1 = 0,$$
$$\alpha x_1 + \alpha x_3 + x_4 + \alpha^2 x_2 = 0,$$
$$\alpha^2 x_1 + \alpha^2 x_2 + x_3 + \alpha x_4 = 0.$$

This system of equations has the following solution in terms of x_5

$$x_1 = \alpha x_5 + \alpha^2,$$
$$x_2 = x_5 + \alpha,$$
$$x_3 = x_5 + \alpha^2,$$
$$x_4 = \alpha x_5.$$

We now try all possible values of $x_5 \in k$ in order to find our which of the possible plaintext vectors is related to the ciphertext we have, that being $(1, 0, 0, 0, 1)$. All of the different values for x_5 come form the set of equations above. Recall that our field in this example only had 4 elements, so the computation is fairly simple.

Table 2.: Table shows us trying all values to determine which plaintext created the cipher text.

plaintext		ciphertext
$(\alpha^2, \alpha, \alpha^2, 0, 0)$	$\iff$	$(1, 0, 0, 0, 1)$
$(1, \alpha^2, \alpha, \alpha, 1)$	$\iff$	$(0, \alpha, 0, \alpha^2, \alpha)$
$(0, 0, 1, \alpha^2, \alpha)$	$\iff$	$(\alpha, 1, 0, \alpha, \alpha^2)$
$(\alpha, 1, 0, 1, \alpha)$	$\iff$	$(\alpha^2, \alpha^2, 0, 1, 0)$

As we can see from table 2, there is only one plain text the matches our given ciphertext and therefor the orignal message must have been $(\alpha^2, \alpha, \alpha^2, 0, 0)$. Both examples in this chapter were adapted from the book titled *Multivariate Public Key Cryptosystems*[37].

10.6 PERTURBATIONS TO MI

Perturbations are slight modifications to a base system that change the system enough to prevent what ever attack was used. We will now go over the minus perturbation, denoted $MI-$, perturbation of the MI system. The minus method a perturbation designed to prevent linearization attacks. In part it was created by Patarin. The method deletes a few polynomial components from the public key. Typically in the MI system , the public is a map such that $\overline{F} : k^n \to k^n$, with the minus method, the map is changed to

$$\overline{F}^- : k^n \to k^{n-r},$$

where we have deleted r number components from the key. The function is defined by

$$\overline{F}^-(x_1, \ldots, x_n) = (\overline{f}_1, \ldots, \overline{f}_{n-r}).$$

The cryptosystem works as follows.

The Public Key

The field structure of k

The set of polynomials $(\overline{f}_1, \ldots, \overline{f}_{n-r}) \in k[x_1, \ldots, x_n]$

The Private Key

Same as the original cryptosystem

Signature Generation

The document is $Y'- = (y'_1, \ldots, y'_{n-r})$, a vector in k^{n-r}. A user chooses $n - r$ random elements $y'_{n-r+1}, \ldots, y'_n \in k$, which are added to $Y'-$, producing the following vector $Y' = (y'_1, \ldots, y'_n) \in k^n$. Then the user produces the following plain text

$$X' = (x' - 1, \ldots, x_n) = \overline{F}(Y'),$$

using the same decryption process as the original MI scheme. This produces X' which is the signature of the document $Y'-$.

Signature Verification

Anyone who has access to X',the signature and the document $Y'-$ can authenticate the signature. The accomplish this by verifying the following equality is true,

$$(\overline{f}_1(X'), \ldots, \overline{f}_{n-r}(X')) = Y'.$$

If the equality is true then the signature is accepted.

10.7 OTHER SCHEMES AND CURRENT DIRECTIONS

There are four major families of schemes being reached currently. All in their base form, without perturbations, have successfully be attacked. Perturbations have been introduced like $MI-$ which fix various attacks. Research continues on multivariate schemes given their efficiency and resistance towards quantum computer based attacks. The security of the system is not found within the $\mathcal{MQ}$ problem, but people are actively exploring which systems work and what they can be used for. Recently a variation of the $MI-$ method called $Sflash$, was accepted be the European Union, as viable cryptosystem for use in low-end smart cards. Below is a figure that shows the basic schemes and how they are related to one another

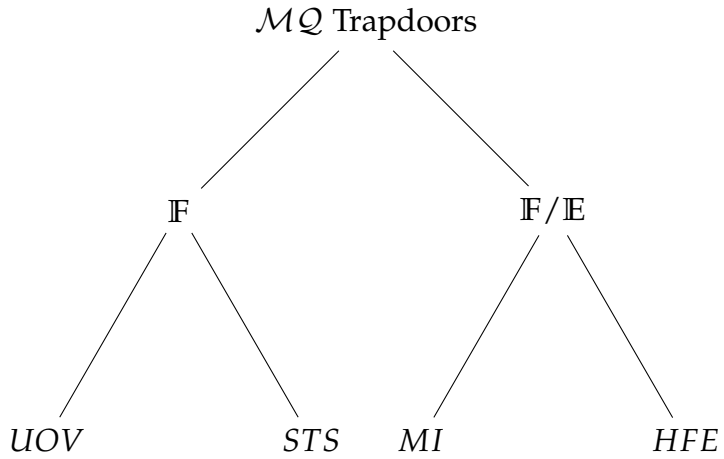


Figure 18.: The four basic schemes from left to right are Unbalanced oil and vinegar, the step-transition scheme, Matsumoto-Imai, and Hidden Field Equations. $\mathbb{F}$ denotes single field construction, and $\mathbb{F}/$, denotes mixed field construction [36].

It is now time to explore other quantum secure encryption schemes and there variations. The next chapter of the book outlines lattice based cryptography, another system that is resistance to quantum computer based factorization attacks.

11

LATTICES IN CRYPTOLOGY – IAN GILBERT

Similar to the Chapter 10, lattices offer a new set of mathematical problems that are resistant to quantum computing. These problems have multiple potential benefits over the current standards. On the one hand, there are simply more of them, leading to a more varied set of cryptosystems, each with their own unique challenges. Additionally, as we will see, lattices involve almost exclusively linear operations, which computers can calculate significantly faster than the modular arithmetic heavy systems like RSA and Diffie-Helman. This effect is even greater when you consider that not only can CPUs perform linear calculations faster than modular ones, GPUs are specifically designed for this. I mean this quite literally. The one, and only one, reason that GPUs exist is so that they can perform linear operations as fast as we can possibly make them. This means that we can use larger key sizes without slowing the computer down, leading to more secure systems. The vast majority of this chapter is based on [1].

11.1 A BRIEF REVIEW OF LINEAR ALGEBRA

Before we begin our discussion of lattices, we need to review some properties of linear algebra – particularly those about vector spaces. For our purposes, we can restrict our attention to vector spaces over $\mathbb{R}^n$. I will forgo any proofs in this section, as I assume you have already taken a linear algebra course in which all of this has been covered. Simply let this serve as a refresher before we explore the directly related field of lattices.

Definition 11.1.1. A vector space V over $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ that is closed under vector addition and scalar multiplication. In other words, for any $\mathbf{v}_1, \mathbf{v}_2 \in V$ and $\alpha \in \mathbb{R}$, V has the property that $\alpha\mathbf{v}_1 \in V$ and $\mathbf{v}_1 + \mathbf{v}_2 \in V$.

Definition 11.1.2. Given a set of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ in a vector space V, a linear combination is a vector $\mathbf{w} \in V$ such that $\alpha_1\mathbf{v}_1 + \cdots + \alpha_k\mathbf{v}_k = \mathbf{w}$ where $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$.

Definition 11.1.3. Given a set of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$, the span of $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ is the set of all linear combinations $\{\alpha_1\mathbf{v}_1 + \cdots + \alpha_k\mathbf{v}_k \mid \alpha_1, \ldots, \alpha_k \in \mathbb{R}\}$.

Definition 11.1.4. A set of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k \in V$ is linearly independent if $\alpha_1\mathbf{v}_1 + \cdots + \alpha_k\mathbf{v}_k = 0$ necessarily implies that $\alpha_1 = \cdots = \alpha_k = 0$. Otherwise the set of vectors is called linearly dependent.

Definition 11.1.5. A basis of a vector space V is a set of linearly independent vectors that span V. In other words, $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ is a basis of V if any vector $\mathbf{w} \in V$ can be written in the form $\alpha_1\mathbf{v}_1 + \cdots + \alpha_k\mathbf{v}_k = \mathbf{w}$ for some *unique* choice of $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$.

Remark 11.1.6. There are infinitely many basis for a given vector space, however there will always be the same number of elements in each basis. This is known as the dimension of the vector space.

Theorem 11.1.7. *Let V be a vector space over $\mathbb{R}^n$, $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be a basis for V, and let $\mathbf{w}_1, \ldots, \mathbf{w}_n$ be another set of vectors in V. Now let B_1 be the matrix with rows $\mathbf{v}_1, \ldots, \mathbf{v}_n$ and B_2 be the matrix with rows $\mathbf{w}_1, \ldots, \mathbf{w}_n$. Then $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ is a basis of V if and only if there is an invertible matrix A such that $B_1 = B_2A$.*

Now we explain the important notion of the length of a vector in $\mathbb{R}^n$ using the dot product and the Euclidean norm.

Definition 11.1.8. Let $\mathbf{v}, \mathbf{u} \in V \subseteq \mathbb{R}^n$, and write $\mathbf{v}$ and $\mathbf{u}$ using coordinates $\mathbf{v} = (x_1, \ldots, x_n)$ and $\mathbf{u} = (y_1, \ldots, y_n)$. The dot product of $\mathbf{v}$ and $\mathbf{u}$ is given by

$$\mathbf{v} \cdot \mathbf{u} = x_1y_1 + \cdots + x_ny_n$$

There are many ways to measure the length of a vector, but we will use the most common: the Euclidean Norm.

Definition 11.1.9. The Euclidean Norm of a vector $\mathbf{v} = (x_1, \ldots, x_n)$ is given by

$$\|\mathbf{v}\| = \sqrt{x_1^2 + \cdots + x_n^2}$$

Notice that the dot product and Euclidean Norm are related by $\|\mathbf{v}\|^2 = \mathbf{v} \cdot \mathbf{v}$.

Theorem 11.1.10. *Let $\mathbf{v}, \mathbf{u} \in V$, and let θ denote the angle between $\mathbf{v}$ and $\mathbf{u}$. Then the dot product is given by*

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

Corollary 11.1.11 (Cauchy-Schwarz Inequality)**.** *Let $\mathbf{v}, \mathbf{u} \in V$, then $|\mathbf{v} \cdot \mathbf{u}| \leq \|\mathbf{v}\| \, \|\mathbf{u}\|$.*

Definition 11.1.12. Two vectors are orthogonal if their dot product is zero.

Definition 11.1.13. An orthogonal basis for a vector space V is a basis $\{v_1, \ldots, v_n\}$ with the property that

$$\mathbf{v}_i \cdot \mathbf{v}_j = 0 \quad \text{for all } i \neq j$$

An *orthonormal basis* has the additional property that $\|\mathbf{v}_i\| = 1$ for all $i \in 1, \ldots, n$.

Remark 11.1.14. Many processes become much simpler when using an orthogonal or orthonormal basis. For example, if $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ is an orthogonal basis and $\mathbf{w} = \alpha_1\mathbf{v}_1 + \cdots + \alpha_n\mathbf{v}_n$ is a linear combination of basis vectors, then

$$\begin{aligned}\|\mathbf{w}\|^2 &= \mathbf{w} \cdot \mathbf{w} \\ &= (\alpha_1\mathbf{v}_1 + \cdots + \alpha_n\mathbf{v}_n) \cdot (\alpha_1\mathbf{v}_1 + \cdots + \alpha_n\mathbf{v}_n) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j(\mathbf{v}_i \cdot \mathbf{v}_j) \\ &= \sum_{i=1}^{n} \alpha_i^2 \|\mathbf{v}_i\|^2 \quad \text{since } \mathbf{v}_i \cdot \mathbf{v}_j = 0 \text{ for } i \neq j\end{aligned}$$

And if the basis is orthonormal, then it simplifies even further to $\|\mathbf{w}\|^2 = \sum \alpha_i^2$.

Since this is such a desirable property, we may want to know when we it is possible create an orthogonal or orthonormal basis, and how to find it. It turns out that this is always possible, and given any basis, an orthogonal and orthonormal one can be found using the well-known Gram-Schmidt Algorithm.

Theorem 11.1.15 (Gram-Schmidt Algorithm)**.** *Let $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be a basis for V. Then the following algorithm returns an orthogonal/orthonormal basis for V:*

1. *Set $\mathbf{v}_1^* = \mathbf{v}_1$.*
2. *For each $i \in 2, \ldots, n$*

   $$\text{calculate } \mu_{ij} = \frac{\mathbf{v}_i \cdot \mathbf{v}_j^*}{\left\|\mathbf{v}_j\right\|^2} \quad \text{for all } 1 \leq j < i,$$

   $$\text{and set } \mathbf{v}_i^* = \mathbf{v}_i - \sum_{j=1}^{i-1} \mu_{ij}\mathbf{v}_j^*$$

3. *$\{\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*\}$ is an orthogonal basis for V.*
4. *$\left\{\mathbf{v}_1^*/\|\mathbf{v}_1^*\|, \ldots, \mathbf{v}_n^*/\|\mathbf{v}_n^*\|\right\}$ is an orthonormal basis for V.*

Definition 11.1.16. Let V be a vector space, and let W be a vector subspace of V. The *orthogonal complement* of W in V is the set

$$W^\perp = \{\mathbf{v} \in V \mid \mathbf{v} \cdot \mathbf{w} = 0 \quad \text{for all } \mathbf{w} \in W\}$$

Lemma 11.1.17. *$W^\perp$ is a subspace of V and every vector $\mathbf{v} \in V$ can be written as a sum $\mathbf{v} = \mathbf{w} + \mathbf{w}'$ for some unique vectors $\mathbf{w} \in W$ and $\mathbf{w}' \in W^\perp$.*

11.2 DEFINITION AND PROPERTIES OF LATTICES

With that out of the way, we begin our discussion of lattices in earnest. In this section we will go over the definition of a lattice, as well as examine some of the important properties of lattices. The necessity for the review of vector spaces becomes clear after seeing the definition of a lattice. In many ways, lattices are to vector spaces what the integers are to the reals.

Definition 11.2.1. Let $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^n$ be a set of linearly independent vectors. Then the lattice generated by $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is the set of linear combinations of $\mathbf{v}_1, \ldots, \mathbf{v}_k$ with integer coefficients:

$$\mathcal{L} = \{a_1\mathbf{v}_1 + \cdots + a_k\mathbf{v}_k \mid a_1, \ldots a_k \in \mathbb{Z}\}$$

Remark 11.2.2. Just like with vector spaces, we can define the basis to be any linearly independent set of vectors that generates $\mathcal{L}$, and the dimension as the number of vectors in the basis. For our purposes, we will only consider full-rank lattices, meaning that if $\mathcal{L} \subset \mathbb{R}^n$, then $\dim(\mathcal{L}) = n$.

Lemma 11.2.3. *A basis for a full-rank lattice over $\mathbb{R}^n$ is also a basis for $\mathbb{R}^n$.*

Proof. Let $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be the basis of some full-rank lattice over $\mathbb{R}^n$. The definition of linear independence is not changed between a vector space and a lattice, and therefore $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are also linearly independent in $\mathbb{R}^n$. Since any set of n linearly independent vectors in $\mathbb{R}^n$ is a basis of $\mathbb{R}^n$, we have that $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ is a basis of $\mathbb{R}^n$. □

Remark 11.2.4. This lemma tells us that if we have a basis for a full-rank lattice, we can write any vector in R^n as a (not necessarily integral) linear combination of the basis vectors. This will become very useful throughout this chapter.

Note that the converse of this lemma is false – any basis of R^n is not necessarily a basis of $\mathcal{L}$. It will be the basis of *some* lattice, just not necessarily this one.

Now we introduce the concept of a change of basis in a lattice. In order to do this, however, we need the following definition.

Definition 11.2.5 (Unimodular Matrix)**.** A unimodular matrix is a square matrix with integer elements and a determinant equal to ± 1.

Lemma 11.2.6. *If U is a unimodular matrix, then so is U^{-1}.*

Proof. Let U be a unimodular matrix. By the definition of the inverse and determinant of a matrix, we have that

$$\det(UU^{-1}) = \det(U)\det(U^{-1}) = \pm\det(U^{-1}) = \det(I) = 1$$

Therefore $\det(U^{-1}) = \pm 1$. Recall that $U^{-1} = \frac{1}{\det(U)}\text{Adj}(U) = \pm\text{Adj}(U)$. Since U is an integer matrix, $\text{Adj}(U)$ is also an integer matrix, which implies that U^{-1} is an integer matrix. Thus U^{-1} is unimodular. □

Theorem 11.2.7. *Let $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be a basis for $\mathcal{L}$, and let $\mathbf{w}_1, \ldots, \mathbf{w}_n$ be another set of vectors in $\mathcal{L}$. Just as with vector spaces, let B_1 be the matrix with rows $\mathbf{v}_1, \ldots, \mathbf{v}_n$ and B_2 be the matrix with rows $\mathbf{w}_1, \ldots, \mathbf{w}_n$. Then $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ is a basis for $\mathcal{L}$ if and only if $B_1 = UB_2$ for some unimodular matrix U.*

Proof. Define B_1 and B_2 as described above.

($\Rightarrow$) Suppose that $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ is also a basis for $\mathcal{L}$. Then we can write each $\mathbf{v}_i$ in terms of this basis:

$$\begin{aligned}
\mathbf{v}_1 &= a_{11}\mathbf{w}_1 + a_{12}\mathbf{w}_2 + \cdots + a_{1n}\mathbf{w}_n \\
\mathbf{v}_2 &= a_{21}\mathbf{w}_1 + a_{22}\mathbf{w}_2 + \cdots + a_{2n}\mathbf{w}_n \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
\mathbf{v}_n &= a_{n1}\mathbf{w}_1 + a_{n2}\mathbf{w}_2 + \cdots + a_{nn}\mathbf{w}_n
\end{aligned}$$

Note that every a_{ij} is an integer. Now we create the matrix

$$U = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

It is clear that $B_1 = UB_2$. Similarly, we can find a matrix V such that $B_2 = VB_1$, and by substitution, $B_2 = VB_1 = VUB_2$. By properties of the transpose of a matrix, we can say that $B_2^T B_2 = B_2^T (VU)^T (VU) B_2$, and it follows that

$$\det(B_2^T B_2) = \det(VU)^2 \det(B_2^T B_2) \;\Rightarrow\; \det(VU) = \det(V)\det(U) = \pm 1$$

Therefore U is a unimodular matrix.

($\Leftarrow$) Suppose that $B_1 = UB_2$ for some unimodular matrix U. To show that $\mathbf{w}_1, \ldots, \mathbf{w}_n$ are linearly independent, note that since $\det(B_1)$ and $\det(U)$ are both non-zero, $\det(B_2)$ must also be non-zero.

Now since $B_1 = UB_2$, we know that we can write every $\mathbf{v}_i$ as interger multiples of the $\mathbf{w}_j$ vectors. This means that the lattice generated by $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ is a subset

of the lattice generated by $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$. Similarly, since $B_2 = U^{-1}B_1$, we know that the lattice generated by $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ is a subset of the lattice generated by $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. Therefore these are the same lattices, and, in particular, $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ is a basis for $\mathcal{L}$.

Each statement has been shown to imply the other, and thus the proposition is true. □

Remark 11.2.8. Given a basis of a lattice, this theorem tells us that we can find a new basis by simply multiplying the vectors of the given basis by a unimodular matrix. This will become very useful in Sections 11.5 and 11.6.

Example 11.2.9. Consider the 3-dimensional lattice $\mathcal{L} \subset \mathbb{R}^3$ generated by $\mathbf{v}_1 = (2, 1, 3)$, $\mathbf{v}_2 = (1, 2, 0)$, and $\mathbf{v}_3 = (2, -3, -5)$, and create the matrix:

$$B_1 = \begin{pmatrix} 2 & 1 & 3 \\ 1 & 2 & 0 \\ 2 & -3 & -5 \end{pmatrix}$$

Now we need to create a unimodular matrix. We can do this by performing arbitrary row operations on the identity matrix. For example, take the following matrix:

$$U = \begin{pmatrix} 1 & 0 & 1 \\ 1 & -1 & 2 \\ 1 & 2 & 0 \end{pmatrix}$$

We will take the vectors $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$ to be the rows of the following matrix:

$$B_2 = UB_1 = \begin{pmatrix} 4 & -2 & -2 \\ 5 & -7 & -7 \\ 4 & 5 & 3 \end{pmatrix}$$

so $\mathbf{w}_1 = (4, -2, -2)$, $\mathbf{w}_2 = (5, -7, -7)$, and $\mathbf{w}_3 = (4, 5, 3)$.

Now we see that

$$B_1 = U^{-1}B_2 = \begin{pmatrix} 4 & -2 & -1 \\ -2 & 1 & 1 \\ -3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 4 & -2 & -2 \\ 5 & -7 & -7 \\ 4 & 5 & 3 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 3 \\ 1 & 2 & 0 \\ 2 & -3 & -5 \end{pmatrix}$$

Therefore, since $\det(U^{-1}) = -1$, Theorem 11.2.7 says that $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$ is a new basis for $\mathcal{L}$.

Remark 11.2.10. The matrices U and U^{-1} show us how to express each basis in terms of the other. So in the previous example, we can use the rows of U to say that $\mathbf{w}_1 = \mathbf{v}_1 + \mathbf{v}_3$, $\mathbf{w}_2 = \mathbf{v}_1 - \mathbf{v}_2 + 2\mathbf{v}_3$, and $\mathbf{w}_3 = \mathbf{v}_1 + 2\mathbf{v}$. Similarly, we can use the rows of U^{-1} to say that $\mathbf{v}_1 = 4\mathbf{w}_1 - 2\mathbf{w}_2 - \mathbf{w}_3$, $\mathbf{v}_2 = -2\mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3$, and $\mathbf{v}_3 = -3\mathbf{w}_1 + 2\mathbf{w}_2 + \mathbf{w}_3$.

It is often easier to talk about lattices where the elements of the vectors themselves are integers as well, as in the example above.

Definition 11.2.11. An integral (or integer) lattice is a lattice over $\mathbb{Z}^n$. In other words, an integral lattice is a lattice generated by vectors in $\mathbb{Z}^n$.

Definition 11.2.12. A subset L of $\mathbb{R}^n$ is an additive subgroup if it is closed under addition and subtraction. It is called a discrete additive subgroup if there is a positive constant $\epsilon > 0$ such that for every $\mathbf{v} \in L$,

$$L \cap \left\{ \mathbf{w} \in \mathbb{R}^m \;\middle|\; \|\mathbf{v} - \mathbf{w}\| < \epsilon \right\} = \{\mathbf{v}\}$$

Those familiar with topology may recognize this as saying that there is some $\epsilon > 0$ such that the open ball in L centered at $\mathbf{v}$ of radius ϵ, contains only $\mathbf{v}$.

Theorem 11.2.13. *A subset of $\mathbb{R}^n$ is a lattice if and only if it is a discrete additive subgroup.*

Lattices are often easier to understand in 2-dimensions as orderly points in the $\mathbb{R}^2$ plane. Each point would be the tip of a vector in the lattice. An example of this is shown in Figure 19.

Definition 11.2.14. Let $\mathcal{L}$ be a lattice of dimension n, and let $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be a basis for $\mathcal{L}$. The fundamental domain (or fundamental parallelepiped[1]) for $\mathcal{L}$ corresponding to this basis is the set

$$\mathcal{F}(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \{t_1\mathbf{v}_1 + \cdots + t_n\mathbf{v}_n \mid 0 \leq t_i < 1\}$$

For a geometric understanding of a fundamental domain, see the 2-dimensional example in Figure 19. The importance of the fundamental domain can be seen in the following theorem.

Theorem 11.2.15. *Let $\mathcal{L}$ be a lattice of dimension n, and let $\mathcal{F}$ be a fundamental domain for $\mathcal{L}$. Then every vector $\mathbf{w} \in \mathbb{R}^n$ can be written in the form $\mathbf{w} = \mathbf{t} + \mathbf{v}$ for a unique $\mathbf{t} \in \mathcal{F}$ and a unique $\mathbf{v} \in \mathcal{L}$.*

Proof. Let $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be a basis of $\mathcal{L}$, and let $\mathcal{F}$ be the associated fundamental domain. Consider some $\mathbf{w} \in \mathbb{R}^n$. By Lemma 11.2.3, we can write $\mathbf{w} = \alpha_1\mathbf{v}_1 + \cdots + \alpha_n\mathbf{v}_n$ where each $\alpha_i \in \mathbb{R}$. It is clear that each α_i can be written as $\alpha_i = t_i + a_i$ where $0 \leq t_i < 1$ and $a_i \in \mathbb{Z}$, and so we can write $\mathbf{w}$ in the form

$$\mathbf{w} = (t_1 + a_1)\mathbf{v}_1 + \cdots + (t_n + a_n)\mathbf{v}_n = \underbrace{t_1\mathbf{v}_1 + \cdots + t_n\mathbf{v}_n}_{\mathbf{t} \in \mathcal{F}} + \underbrace{a_1\mathbf{v}_1 + \cdots + a_n\mathbf{v}_n}_{\mathbf{v} \in \mathcal{L}}$$

1 Say parallelepiped five times fast.

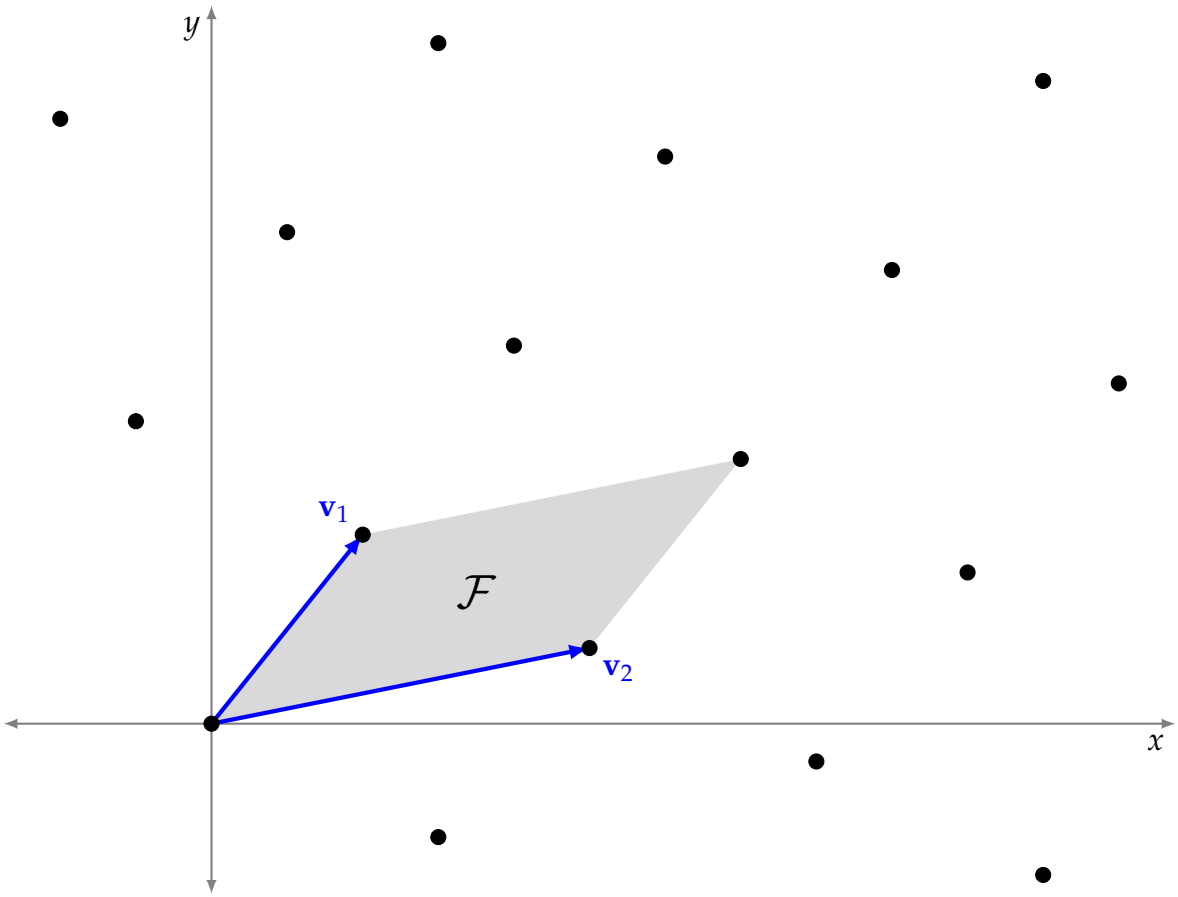


Figure 19.: This is a lattice in $\mathbb{R}^2$ generated by $\{\mathbf{v_1}, \mathbf{v}_2\}$. The fundamental domain corresponding to this basis is the shaded region $\mathcal{F}$.

Now we must show that $\mathbf{t}$ and $\mathbf{v}$ are unique. Suppose that $\mathbf{w} = \mathbf{t} + \mathbf{v} = \mathbf{t}' + \mathbf{v}'$ for some vectors $\mathbf{t}, \mathbf{t}' \in \mathcal{F}$ and $\mathbf{v}, \mathbf{v}' \in \mathcal{L}$. Then we can write $\mathbf{w}$ in the form

$$\mathbf{w} = (t_1 + a_1)\mathbf{v}_1 + \cdots + (t_n + a_n)\mathbf{v}_n = (t'_1 + a'_1)\mathbf{v}_1 + \cdots + (t'_n + a'_n)\mathbf{v}_n$$

Since $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are linearly independent, this implies that each $t_i + a_i = t'_i + a'_i$. Further, this implies that $t_i - t'_i = a_i - a'_i$. Since it is clear that $a_i - a'_i$ is an integer, it follows that $t_i - t'_i$ must also be an integer, however since $0 \leq t_i, t'_i < 1$, this can only happen if $t_i = t'_i$. Therefore

$$\mathbf{v} = \mathbf{w} - \mathbf{t} = \mathbf{w} - \mathbf{t}' = \mathbf{v}'$$

and so $\mathbf{v}$ and $\mathbf{t}$ are unique.

Thus every vector $\mathbf{w} \in \mathbb{R}^n$ can be written in the form $\mathbf{w} = \mathbf{t} + \mathbf{v}$ for a unique $\mathbf{t} \in \mathcal{F}$ and a unique $\mathbf{v} \in \mathcal{L}$. □

Remark 11.2.16. An equivalent way to view this theorem is to say that the union of translations of $\mathcal{F}$ by vectors in $\mathcal{L}$ exactly covers $\mathbb{R}^n$. That is,

$$\bigcup_{\mathbf{v}\in\mathcal{L}} \mathcal{F}+\mathbf{v} = \bigcup_{\mathbf{v}\in\mathcal{L}} \{\mathbf{t}+\mathbf{v} \mid \mathbf{t}\in\mathcal{F}\} = \mathbb{R}^n$$

An example of this is shown in Figure 20.

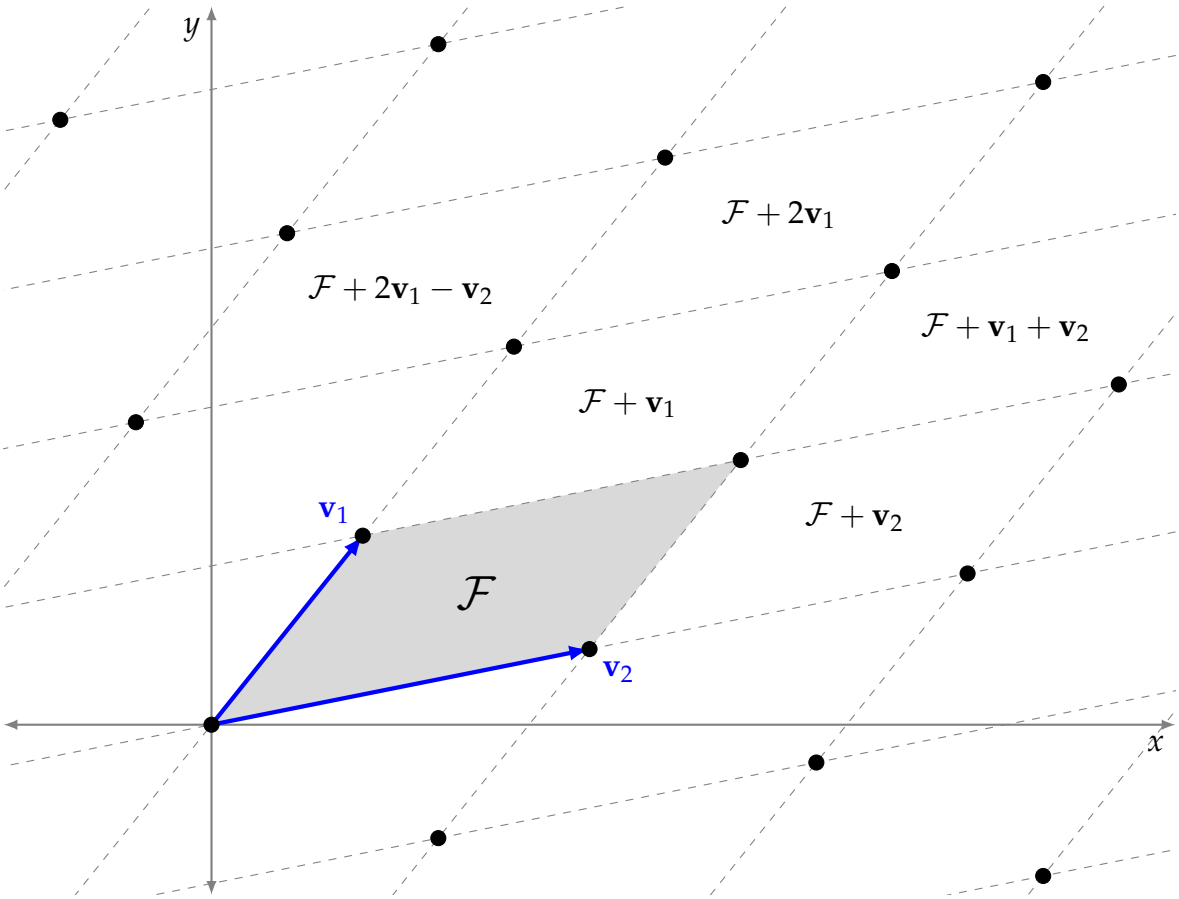


Figure 20.: This shows how translations of $\mathcal{F}$ by vectors in the lattice from 19 exactly covers $\mathbb{R}^2$.

Definition 11.2.17. Let $\mathcal{L}$ be a lattice of dimension n, and let $\mathcal{F}$ be a fundamental domain for $\mathcal{L}$. Then the determinant of $\mathcal{L}$, denoted $\det(\mathcal{L})$, is given by the n-dimensional volume of the fundamental domain, $\mathrm{Vol}(\mathcal{F})$.

It may be a little confusing how to use this, however the following theorem helps to clear this up.

Theorem 11.2.18. *Let $\mathcal{L}$ be a lattice of dimension n, let $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be a basis for $\mathcal{L}$, and let $\mathcal{F}$ be the associated fundamental domain. Write each vector in the basis as $\mathbf{v}_i = (r_{i1}, r_{i2}, \ldots, r_{in})$ and use the coordinates of each vector as the rows of a matrix:*

$$F = F(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nn} \end{pmatrix}$$

Then the dimension of the lattice is given by $\dim(\mathcal{L}) = \mathrm{Vol}(\mathcal{F}) = \left|\det(F)\right|$.

Proof. $\mathcal{F}$ is just a region in the $\mathbb{R}^n$ plane, and so we can calculate the volume of $\mathcal{F}$ as the integral of the constant function 1 over the region $\mathcal{F}$:

$$\mathrm{Vol}(\mathcal{F}) = \int_{\mathcal{F}} dx_1 \cdots dx_n$$

Now we can make a change of variables from $\mathbf{x} = (x_1, \ldots, x_n)$ to $\mathbf{t} = (t_1, \ldots, t_n)$ for $\mathbf{t} \in \mathcal{F}$ according to the formula

$$(x_1, \ldots, x_n) = t_1\mathbf{v}_1 + \cdots + t_n\mathbf{v}_n = \mathbf{t}F$$

The Jacobian matrix of this change of variables is F, and the fundamental domain $\mathcal{F}$ is the image under F of the unit cube $C_n = [0,1]^n$, so the change of variables formula for integrals gives us

$$\begin{aligned} \mathrm{Vol}(\mathcal{F}) &= \int_{\mathcal{F}} dx_1 \cdots dx_n = \int_{FC_n} dx_1 \cdots dx_n = \int_{C_n} \left|\det(F)\right| \, dt_1 \cdots dt_n \\ &= \left|\det(F)\right| \mathrm{Vol}(C_n) = \left|\det(F)\right| \end{aligned}$$

Thus the dimension of a lattice is given by $\dim(\mathcal{L}) = \mathrm{Vol}(\mathcal{F}) = \left|\det(F)\right|$. □

Remark 11.2.19. Calculating the determinant of this matrix is much easier than computing the n-dimensional volume of the fundamental domain directly, and this is why it is useful to talk about $\det(\mathcal{L})$ over $\mathrm{Vol}(\mathcal{F})$.

Example 11.2.20. The lattice described in Example 11.2.9 has determinant

$$\det(\mathcal{L}) = \left|\det(U)\right| = \left|\det \begin{pmatrix} 2 & 1 & 3 \\ 1 & 2 & 0 \\ 2 & -3 & -5 \end{pmatrix}\right| = |-36| = 36$$

Corollary 11.2.21. *Let $\mathcal{L}$ be a lattice of dimension n. Then every fundamental domain for $\mathcal{L}$ has the same volume.*

Proof. Let $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ and $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ be two bases for a lattice $\mathcal{L}$, with associated fundamental domains $\mathcal{F}(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ and $\mathcal{F}(\mathbf{w}_1, \ldots, \mathbf{w}_n)$. Let $F(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ and $F(\mathbf{w}_1, \ldots, \mathbf{w}_n)$ be the matrices as described in Theorem 11.2.18. By Theorem 11.2.7, we can write $F(\mathbf{v}_1, \ldots, \mathbf{v}_n) = F(\mathbf{w}_1, \ldots, \mathbf{w}_n)U$ for some unimodular matrix U. Now we can use Theorem 11.2.18 to say that

$$\begin{aligned}\operatorname{Vol}\big(\mathcal{F}(\mathbf{v}_1, \ldots, \mathbf{v}_n)\big) &= \Big|\det\big(F(\mathbf{v}_1, \ldots, \mathbf{v}_n)\big)\Big| = \Big|\det\big(F(\mathbf{w}_1, \ldots, \mathbf{w}_n)U\big)\Big| \\ &= \Big|\det\big(F(\mathbf{w}_1, \ldots, \mathbf{w}_n)\big)\Big|\,\Big|\det(U)\Big| = \Big|\det\big(F(\mathbf{w}_1, \ldots, \mathbf{w}_n)\big)\Big| \\ &= \operatorname{Vol}\big(\mathcal{F}(\mathbf{w}_1, \ldots, \mathbf{w}_n)\big)\end{aligned}$$

Thus every fundamental domain for $\mathcal{L}$ has the same volume. □

Remark 11.2.22. A useful implication of this corollary is that $\det(\mathcal{L})$ is an invariant of the lattice $\mathcal{L}$, independent of the particular basis used.

One final property of lattices and fundamental domains begins to relate the dimension of a lattice to the important concept of orthogonality.

Lemma 11.2.23 (Hadamard's Inequality)**.** *Let $\mathcal{L}$ be a lattice of dimension n, let $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be a basis for $\mathcal{L}$, and let $\mathcal{F}$ be the associated fundamental domain. Then*

$$\det(\mathcal{L}) = \operatorname{Vol}(\mathcal{F}) \leq \|\mathbf{v}_1\|\,\|\mathbf{v}_2\| \cdots \|\mathbf{v}_n\|$$

In particular, as the basis becomes more orthogonal, this becomes closer to an equality.

This lemma gives us a way to measure the orthogonality of a basis.

Definition 11.2.24 (Hadamard Ratio)**.** The Hadamard ratio of the basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ for a lattice $\mathcal{L}$ of dimension n is the term

$$\mathcal{H}(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \left(\frac{\det(\mathcal{L})}{\|\mathbf{v}_1\|\,\|\mathbf{v}_2\| \cdots \|\mathbf{v}_n\|}\right)^{1/n}$$

The more orthogonal the basis, the closer this value is to 1, while less orthogonal bases will approach 0. The reciprocal of the Hadamard ratio is called the *orthogonality defect*.

11.3 THE MERKLE-HELLMAN KNAPSACK CRYPTOSYSTEM

Our first foray into lattices and cryptography is actually not a lattice-based cryptosystem, but rather a cryptosystem that was broken using lattices. The Merkle-Hellman Knapsack Cryptosystem was the first attempt at a cryptosystem based on an *NP*-complete problem.

The knapsack problem goes like this: there is a jewelry thief[2] who has just broken into a jewelry store. He wants to smuggle out as many jewels as he can, however the jewelry store has many different kinds of jewels, all worth different amounts, and the thief can't fit all of them into his knapsack. What is the highest cumulative worth that the thief can steal?

Even in the special case (called the subset-sum problem) where the value of each object is equal to it's size, this is not an easy problem to solve. In fact, there is not even guaranteed to be a solution at all.

So how do we turn this into a cryptosystem? First of all, we need to formalize this subset-sum problem.

Definition 11.3.1 (Generalized Subset-Sum Problem)**.** Suppose that you are given some ordered list of integers $H = (h_1, \ldots, h_n)$ and another integer S such that it is the sum of some number of elements from H. Find a subset of H such that the elements sum to exactly S (there may be more than one solution).

Example 11.3.2. Let $H = (2, 3, 4, 9, 14, 23)$ and $S = 27$. With some trial and error, we determine that the only solution is $\{4, 9, 14\}$. If, however, we set $S = 29$, we would find that $\{2, 4, 23\}$ and $\{2, 4, 9, 14\}$ are both solutions.

There is another, more cryptographic, way to describe this problem. Let the set $H = (h_1, \ldots, h_n)$ be public knowledge. Bob wants to send a message to Alice, and so he chooses some binary vector $\mathbf{m} = (m_1, \ldots, m_n)$ and computes the sum

$$S = \sum_{i=1}^{n} m_i h_i$$

and sends S to Alice. Note that the binary vector $\mathbf{m}$ simply determines which elements will appear in the sum. The subset-sum problem then asks Alice to either find $\mathbf{m}$ or some other binary vector that satisfies S.

If Alice wants to brute-force this problem, she will have to check all 2^n binary vectors of length n. This seems like a lot of work though, so she looks for a faster solution.

Theorem 11.3.3. *Let $H = (h_1, \ldots, h_n)$ and let (H, S) be a subset-sum problem. For all sets of integers*

$$I = \left\{ i \in \mathbb{Z} \;\middle|\; 1 \leq i \leq \frac{1}{2}n \right\} \qquad \text{and} \qquad J = \left\{ j \in \mathbb{Z} \;\middle|\; \frac{1}{2}n < j \leq n \right\}$$

compute and make a list of the values

$$A_I = \sum_{i \in I} h_i \qquad \text{and} \qquad B_J = S - \sum_{j \in J} h_j$$

2 Let's call him Dan [reference the dan steals things quote].

Then there exists a pair of sets I_0 and J_0 such that $A_{I_0} = B_{J_0}$. These sets give a solution to the subset-sum problem:

$$S = \sum_{i \in I_0} h_i + \sum_{j \in J_0} h_j$$

Proof. Suppose that $\mathbf{m}$ is the solution to the subset-sum problem (H, S). Then it is clear that we can write

$$S = \sum_{i=1}^{n} m_i h_i = \sum_{1 \le i \le \frac{1}{2}n} (m_i h_i) + \sum_{\frac{1}{2} < i \le n} (m_i h_i) \;\Rightarrow\; \sum_{1 \le i \le \frac{1}{2}n} (m_i h_i) = S - \sum_{\frac{1}{2} < i \le n} (m_i h_i)$$

Therefore the only sets I_0 and J_0 such that $A_{I_0} = B_{J_0}$ will be sets such that

$$\sum_{i \in I_0} h_i = \sum_{1 \le i \le \frac{1}{2}n} (m_i h_i) \quad \text{and} \quad \sum_{j \in J_0} h_j = \sum_{\frac{1}{2} < i \le n} (m_i h_i)$$

Thus $\sum_{i \in I_0} h_i + \sum_{j \in J_0} h_j$ will be a solution to the subset-sum problem. □

Remark 11.3.4. The number of entries in each list is at most $2^{n/2}$, and so we have cut the exponent of the time it takes to solve the problem in half. This is an example of a collision algorithm.

While Theorem 11.3.3 definitely reduces the time it takes to brute-force the subset-sum problem, it is still unfeasible when n is large. If this is to become the basis for a cryptosystem, we will need to find some sort of trapdoor that Alice can use to quickly solve her own subset-sum problem.

Imagine that there was some special class of subset-sum problems that are very easy to solve, and imagine that there was some way to disguise this problem so that other people couldn't see how to solve it. Wouldn't that be great?

Fortunately, such a system exists! We now define this special class, which are called superincreasing sequence subset-sum problems.

Definition 11.3.5. A superincreasing sequence of integers is an ordered list of positive integers $\mathbf{r} = (r_1, \ldots, r_n)$ with the property that $r_{i+1} \ge 2r_i$ for all $1 \le i \le n-1$.

The name superincreasing comes from the following lemma.

Lemma 11.3.6. *Let $\mathbf{r} = (r_1, \ldots, r_n)$ be a superincreasing sequence. Then*

$$r_k > r_{k-1} + \cdots + r_2 + r_1 \quad \textit{for all } 2 \le k \le n$$

Proof. Let $\mathbf{r} = (r_1, \ldots, r_n)$ be a superincreasing sequence. We proceed by induction:

Base Case: Suppose that $k = 2$. Then by the definition of a superincreasing sequence, $r_2 \geq 2r_1 > r_1$.

Induction Hypothesis: Suppose that $r_k > r_{k-1} + \cdots + r_2 + r_1$, and consider r_{k+1}. By the definition of a superincreasing sequence and our induction hypothesis, we find

$$r_{k+1} \geq 2r_k = r_k + r_k > r_k + (r_{k-1} + \cdots + r_2 + r_1)$$

Thus $r_k > r_{k-1} + \cdots + r_2 + r_1$ for all $2 \leq k \leq n$. □

A subset-sum problem in which the integers in H form a superincreasing sequence is very easy to solve, as the following theorem shows.

Theorem 11.3.7. *Let $(\mathbf{r}, S)$ be a subset-sum problem in which the integers in $\mathbf{r}$ form a superincreasing sequence. Assuming that a solution exists, it is unique and may be computed by the following fast algorithm:*

1. *Set $S_n = S$.*
2. *For each $i \in n, \ldots, 1$ (note that i starts at n and decreases):*

   If $S_i \geq r_i$, set $m_i = 1$ and set $S_{i-1} = S_i - r_i$.

   Otherwise, set $m_i = 0$ and $S_{i-1} = S_i$.
3. *The binary vector $\mathbf{m} = (m_1, \ldots, m_n)$ is the solution to the subset-sum problem.*

Proof. Let $(\mathbf{r}, S)$ be a subset-sum problem in which the integers in $\mathbf{r}$ form a superincreasing sequence, and let $\mathbf{x}$ be a solution to the subset-sum problem. We want to show that the vector $\mathbf{m}$ produced by the algorithm is equal to $\mathbf{x}$, both proving that the algorithm works and that the solution is unique in one step. We proceed by backwards induction:

Base Case: Consider m_n. There are two cases:

(Case 1) Suppose that $x_n = 1$, which implies that $S \geq r_n$. Therefore, step 2 of the algorithm would return $m_n = 1$.

(Case 2) Suppose that $x_n = 0$. This implies that $S \leq r_1 + \cdots + r_{n-1}$, which, by Lemma 11.3.6, implies that $S < r_n$. Therefore, step 2 of the algorithm would return $m_n = 0$.

In either case, we see that $x_n = m_n$.

Induction Hypothesis: Suppose that $x_i = m_i$ for all $k < i \leq n$. At this stage of the algorithm, we find that S_k has been reduced to

$$S_k = S - \sum_{i=k+1}^{n} m_i r_i = \sum_{i=1}^{n} x_i r_i - \sum_{i=k+1}^{n} m_i r_i = \sum_{i=1}^{k} x_i r_i$$

There are two possibilities for the next step in the algorithm:

(Case 1) Suppose that $x_k = 1$, which implies that $S \geq r_k$. Therefore, step 2 of the algorithm would return $m_k = 1$.

(Case 2) Suppose that $x_k = 0$. This implies that $S \leq r_1 + \cdots + r_{k-1}$, which, by Lemma 11.3.6, implies that $S < r_k$. Therefore, step 2 of the algorithm would return $m_k = 0$.

In either case, we see that $x_k = m_k$. Thus, we see that the algorithm works and that the solution is unique. □

Example 11.3.8. The set $\mathbf{r} = (35, 73, 157, 320, 644)$ is superincreasing. We will now use Theorem 11.3.7 to solve the subset-sum problem where $S = 836$.

1. $S_5 = 836$. Since $836 \geq r_5 = 644$, we set $m_5 = 1$ and $S_4 = S_5 - r_5 = 836 - 644 = 192$.
2. Since $S_4 = 192 < r_4 = 320$, we set $m_4 = 0$ and $S_3 = S_4 = 192$.
3. Now $S_3 = 192 > r_3 = 157$, and so we set $m_3 = 1$ and $S_2 = S_3 - r_3 = 192 - 157 = 35$.
4. Since $S_2 = 35 < r_2 = 73$, we set $m_2 = 0$ and $S_1 = S_2 = 35$.
5. Finally, we see that $S_1 = r_1 = 35$, and so we set $m_1 = 1$. Notice that $S_0 = S_1 - r_1 = 35 - 35 = 0$, which is a good sign.

The binary vector $\mathbf{m} = (1, 0, 1, 0, 1)$ should be the solution to the subset-sum problem. We can check this with

$$\sum_{i=1}^{5} m_i r_i = 1 \cdot 35 + 0 \cdot 73 + 1 \cdot 157 + 0 \cdot 320 + 1 \cdot 644 = 35 + 157 + 644 = 836 = S$$

The cryptosystem proposed by Merkle and Hellman revolves around Alice creating a subset-sum problem with a superincreasing sequence, and then converting that into a non-superincreasing sequence that is made public. Bob can attach a message to this sequence and send it back to Alice. Eve then cannot use the fast algorithm from Theorem 11.3.7 and must brute-force the system in order to attack it. Alice, on the other hand, retains secret information that allows her to convert the sequence she gets from Bob back into a superincreasing sequence and solves it with ease. We now outline the Merkle-Hellman Knapsack Public Key Cryptosystem:[3]

- **Key Creation**: Alice chooses a superincreasing sequence $\mathbf{r} = (r_1, \ldots, r_n)$. She also chooses two integers A and B such that $2 < A < B$ and $B > 2r_n$ and $\gcd(A, B) = 1$. Now for each $i \in 1, \ldots, n$, she sets $h_i \equiv Ar_i \pmod{B}$ to create the non-superincreasing sequence $H = (h_1, \ldots, h_n)$. The sequence H is her public key, while the combination of $(\mathbf{r}, A, B)$ is her private key.

3 Or MHKPKC if you want to be extra confusing.

- **Encryption**: Bob chooses a message and writes it in the form of a binary vector $\mathbf{m} = (m_1, \ldots, m_n)$. Using Alice's public key, H, he computes the ciphertext,

$$S = \sum_{i=1}^{n} m_i h_i$$

and sends this back to Alice.

- **Decryption**: Alice uses her knowledge of A and B, and that $\gcd(A, B) = 1$, to compute $S' \equiv A^{-1}S \pmod{B}$. Notice that

$$S' \equiv A^{-1}S \equiv A^{-1}\sum_{i=1}^{n} m_i h_i \equiv A^{-1}\sum_{i=1}^{n} m_i A r_i \equiv \sum_{i=1}^{n} m_i r_i \pmod{B}$$

By our assumption that $B > 2r_n$ and Lemma 11.3.6, Alice knows

$$\sum_{i=1}^{n} m_i r_i \leq \sum_{i=1}^{n} r_i = \sum_{i=1}^{n-1} (r_i) + r_n < r_n + r_n = 2r_n < B$$

which means that $S' = \sum_{i=1}^{n} m_i r_i$ is an exact equality, not a modular congruence.

This is a subset-sum problem $(\mathbf{r}, S')$ with a superincreasing sequence and solution $\mathbf{m}$. Therefore she can use Theorem 11.3.7 to quickly solve for $\mathbf{m}$ and receive the message from Bob.

Remark 11.3.9. There are a few variations on the basic system that make it slightly more secure. First of all, Alice can create more than one A, and reduce her superincreasing sequence multiple times. She can also take the resulting sequence, and permute it by some function.

Example 11.3.10. Let $\mathbf{r} = (35, 73, 157, 320, 644)$ be Alice's superincreasing sequence, and suppose that she chooses $A = 113$ and $B = 1297$. Note that $1297 > 2 \cdot 644 = 1288$ and that $\gcd(113, 1297) = 1$. Then her public key is

$$\begin{aligned} H &= (113 \cdot 35, 113 \cdot 73, 113 \cdot 157, 113 \cdot 320, 113 \cdot 644) \pmod{1297} \\ &= (64, 467, 880, 1141, 140) \end{aligned}$$

Note that even if you were to rearrange H so that the terms were increasing, this would still not be a superincreasing sequence.

Bob wants to send the message $\mathbf{m} = (1, 0, 1, 0, 1)$ to Alice, and so he computes

$$S = \sum_{i=1}^{5} m_i h_i = 1 \cdot 64 + 0 \cdot 467 + 1 \cdot 880 + 0 \cdot 1141 + 1 \cdot 140 = 64 + 880 + 140 = 1084$$

and sends this to Alice.

Alice receives S and computes

$$S' \equiv A^{-1}S \equiv 264 \cdot 1084 \equiv 836 \pmod{1297}$$

Notice that $(\mathbf{r}, S')$ is the subset-sum problem from Example 11.3.8, where we found the solution $\mathbf{m} = (1,0,1,0,1)$. This is the message from Bob!

So what does any of this have to do with lattices? Suppose that Eve wants to intercept the message from Bob, but Alice and Bob use a large enough dimension subset-sum problem that using the collision algorithm from Theorem 11.3.3 is infeasible. Being the pesky little hacker she is, she might use the public sequence H to form the following matrix.

$$\begin{pmatrix} 2 & 0 & \cdots & 0 & h_1 \\ 0 & 2 & \cdots & 0 & h_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 2 & h_n \\ 1 & 1 & \cdots & 1 & S \end{pmatrix}$$

The important aspect of this matrix are the rows, as listed below:

$$\begin{aligned} \mathbf{v}_1 &= (2,0,\ldots,0,h_1) \\ \mathbf{v}_2 &= (0,2,\ldots,0,h_2) \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ \mathbf{v}_n &= (0,0,\ldots,2,h_n) \\ \mathbf{v}_{n+1} &= (1,1,\ldots,1,S) \end{aligned}$$

Since these vectors are all linearly independent, we can use them to generate a lattice of dimension $n+1$.

Suppose that $\mathbf{m}$ is a solution to the subset-sum problem that Bob sent to Alice. Then this lattice will contain the vector

$$\mathbf{t} = \sum_{i=1}^{n}(m_i\mathbf{v}_i) - \mathbf{v}_{n+1} = (2m_1 - 1, 2m_2 - 1, \ldots, 2m_n - 1, 0)$$

Now, because of some security concerns when the numbers in the sequence $\mathbf{r}$ are too small, we usually require that $r_1 > 2^n$, which ends up implying that $h_1, \ldots, h_n, S$ are all within a factor of 2^{2n}. This means that $\|\mathbf{v}_i\| \approx 2^{2n+1}$ for each vector in the basis of $\mathcal{L}$.

Since each m_i is either a 1 or a 0, it follows that the first n elements of $\mathbf{t}$ will all be ± 1, which means that $\|\mathbf{t}\| = \sqrt{n}$. When all of the basis vectors of a lattice are large, it is very uncommon for them to produce a vector in the lattice substantially smaller than them, and so it very likely that $\mathbf{t}$ is the shortest vector in the lattice. So if Eve has a way to search through the lattice for the shortest vector, then she will be able to break this encryption scheme. This can be done using lattice reduction algorithms, like LLL, which we will go over in Section 11.6. But for now, finding the shortest vector in a lattice is a fundamental problem for the security of this cryptosystem.

11.4 SHORTEST AND CLOSEST VECTOR PROBLEMS

One of the things that makes lattice-based cryptography so exciting is the flexibility of lattices. Within one field of mathematics, we can define a variety of different fundamental problems that can be used as the basis for a cryptosystem, either independently or in conjunction with one another. This serves not only to make the cryptosystem more complex, but if one problem is broken, it is not too difficult to simply replace it with another one. In this section we will introduce the some of the most common such problems in lattices.

We begin with the two most common problems:

Definition 11.4.1 (Shortest Vector Problem (SVP))**.** Find a shortest non-zero vector in a lattice, i.e. find a non-zero vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\|$ is minimized.

Definition 11.4.2 (Closest Vector Problem (CVP))**.** Given a vector $\mathbf{w} \in \mathbb{R}^n$ that is not in $\mathcal{L}$, find a vector $\mathbf{v} \in \mathcal{L}$ that is closest to $\mathbf{w}$, i.e. such that $\|\mathbf{w} - \mathbf{v}\|$ is minimized.

Remark 11.4.3. Note that there may be more than one solution to either problem. For example, in $\mathbb{Z}^2$, the vectors $(1, 0)$ and $(0, 1)$ are both shortest vectors of the lattice. SVP and CVP simply ask for one possible solution.

These problems are both considered to be NP-hard, although SVP is only NP-hard on a certain family of lattices. The majority of the lattice cryptosystems are based on at least one of these problems or one of their variants. Of particular note are the approximate versions of SVP and CVP:

Definition 11.4.4 (Approximate Shortest Vector Problem (apprSVP))**.** Let $\mathcal{L}$ be a lattice of dimension n, and let $\psi(n)$ be a function of n. Find a vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{V}\| \leq \psi(n) \|\mathbf{v}_{shortest}\|$ for some shortest vector $\mathbf{v}_{shortest}$.

Definition 11.4.5 (Approximate Closest Vector Problem (apprCVP))**.** Let $\mathcal{L}$ be a lattice of dimension n, and let $\psi(n)$ be a function of n. Given a vector $\mathbf{w} \in \mathbb{R}^n$ not in $\mathcal{L}$, find a vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{V}\| \leq \psi(n)\|\mathbf{w} - \mathbf{v}_{closest}\|$ for some closest vector $\mathbf{v}_{closest}$.

Roughly speaking, the approximate versions of SVP and CVP simply ask you to solve SVP or CVP to within some small factor. So it doesn't necessarily have to be the exact shortest vector in the lattice, just within that factor.

We also mention one more fundamental problem in lattices – the shortest basis problem.

Definition 11.4.6 (Shortest Basis Problem (SBP))**.** Given a lattice $\mathcal{L}$ of dimension n, find a basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$ that is shortest in some sense. The most common measurements are

- minimize the longest vector, i.e. minimize $\max_{1 \leq i \leq n} \|\mathbf{v}_i\|$
- minimize the cumulative length, i.e. minimize $\sum_{i=1}^{n} \|\mathbf{v}_i\|^2$

SBP is not going to be used in this text, however some of the more cutting-edge systems are beginning to favor SBP over the other SVP and CVP.

11.4.1 *Babai's Algorithm*

Before we move on to our next cryptosystem, we need to discuss Babai's algorithm and good/bad bases for a lattice. A good basis is defined as a basis that is at least reasonably orthogonal, while a bad basis is one that is not. The reason for this is that SVP and CVP become trivial when using an orthogonal basis, as is shown in the following examples.

Example 11.4.7 (SVP with an Orthogonal Basis). Let $B = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be an orthogonal basis for a lattice $\mathcal{L}$. Then, as we saw in Remark 11.1.14, the length of any vector in the lattice is given by

$$\|a_1\mathbf{v}_1 + \cdots + a_n\mathbf{v}_n\|^2 = a_1^2 \|\mathbf{v}_1\|^2 + \cdots + a_n^2 \|\mathbf{v}_n\|^2$$

This is minimized when $a_i = \pm 1$ for $\mathbf{v}_i = \min(B)$ and $a_j = 0$ for all $j \neq i$. In other words, the shortest vector in the lattice is simply the shortest vector in the basis, which is significantly easier to find.

Example 11.4.8 (CVP with an Orthogonal Basis). Similarly, suppose we want to solve CVP in $\mathcal{L}$ for some $\mathbf{w} \in \mathbb{R}^n$. First, let $\mathbf{w} = t_1\mathbf{v}_1 + \cdots + t_n\mathbf{v}_n$ for some $t_1, \ldots, t_n \in \mathbb{R}$. Then for $\mathbf{v} \in \mathcal{L}$ such that $\mathbf{v} = a_1\mathbf{v}_1 + \cdots + a_n\mathbf{v}_n$, we have

$$\|\mathbf{w} - \mathbf{v}\|^2 = (t_1 - a_1)^2 \|\mathbf{v}_1\|^2 + \cdots + (t_n - a_n)^2 \|\mathbf{v}_n\|$$

Since $a_i \in \mathbb{Z}$, this is clearly minimized when $a_i = \lfloor t_i \rceil$.

We commonly generalize Example 11.4.8 in the form of Babai's algorithm, shown below.

Theorem 11.4.9 (Babai's Closest Vertex Algorithm)**.** *Let $\mathcal{L}$ be a lattice with a basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, and let $\mathbf{w} \in \mathbb{R}^n$ be an arbitrary vector such that $\mathbf{w} = t_i\mathbf{v}_1 + \cdots + t_n\mathbf{v}_n$ for $t_1, \ldots, t_n \in \mathbb{R}$. Then the closest vector in $\mathcal{L}$ will be $\mathbf{v} = a_1\mathbf{v}_1 + \cdots + a_n\mathbf{v}_n$ where $a_i = \lfloor t_i \rceil$.*

Babai's algorithm does not specify that the basis be orthogonal, but instead the algorithm simply stops working once the basis becomes bad enough. In general, we say that

Babai's algorithm solves some version of apprCVP if the basis is reasonably orthogonal, but if the basis is not very orthogonal, then it will return a vector very far from $\mathbf{w}$. To see what is happening, we need to understand the geometry of Babai's algorithm a little bit better.

By Theorem 11.2.15, we know that translations of the fundamental domain by vectors exactlyattice covin the ler $\mathbb{R}^n$. Babai's algorithm then selects the unique translation of the fundamental domain that contains $\mathbf{w}$, and returns the closest vertex of that translation of the fundamental domain. When the basis is highly orthogonal, Figure 21 shows how the closest vertex is in fact the closest lattice point to $\mathbf{w}$. However when the basis is highly non-orthogonal, as in Figure 22, we see that this closest vertex is not the closest lattice point. It is also worth noting that this effect gets worse as the dimension increases.

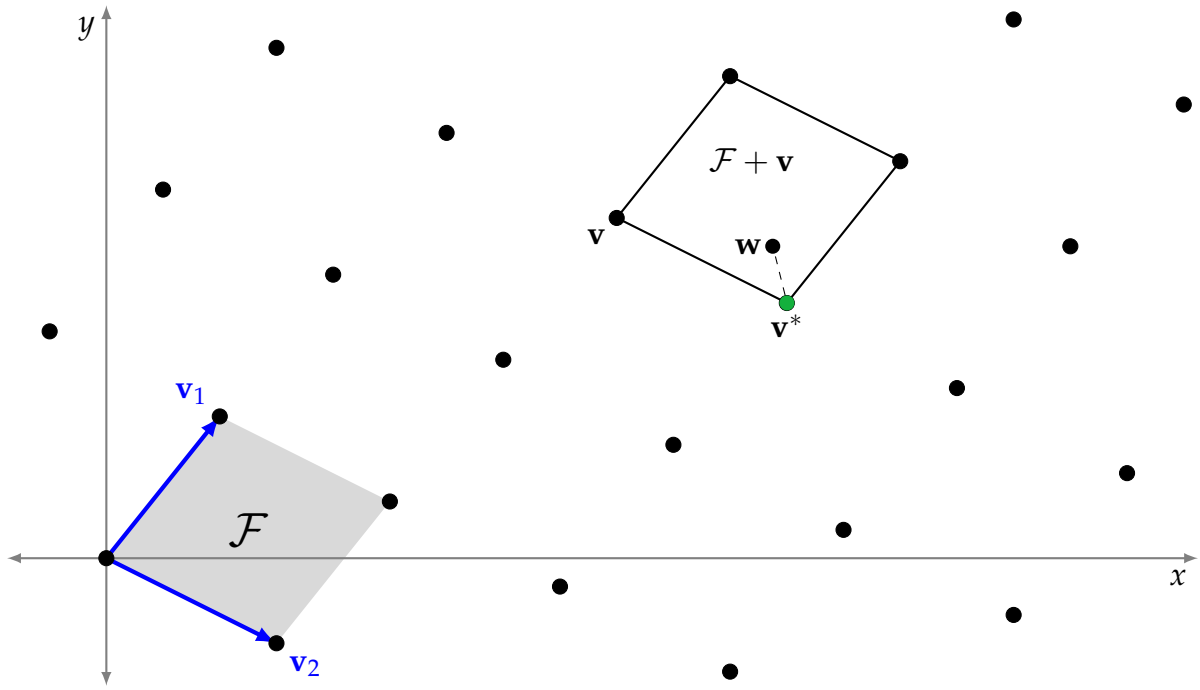


Figure 21.: Babai's algorithm works very well on a good basis. Here we see that $\mathcal{F}+\mathbf{v}$ is the translation of $\mathcal{F}$ that contains $\mathbf{w}$. The closest vertex of this translation, $\mathbf{v}^*$ is in fact the closest vector in the lattice.

Example 11.4.10. Consider the lattice $\mathcal{L} \subset \mathbb{R}^2$ generated by the vectors $\mathbf{v}_1 = (137, 312)$ and $\mathbf{v}_2 = (215, -187)$. We see that the Hadamard ratio defined in Definition 11.2.24 is

$$\mathcal{H}(\mathbf{v}_1, \mathbf{v}_2) = \left(\frac{\det(\mathcal{L})}{\|\mathbf{v}_1\| \, \|\mathbf{v}_2\|}\right)^{1/2} = \left(\frac{92699}{(340.75)(284.95)}\right)^{1/2} \approx 0.977$$

which is reasonably close to 1. Therefore this is a good basis, and we expect Babai's algorithm to work very well.

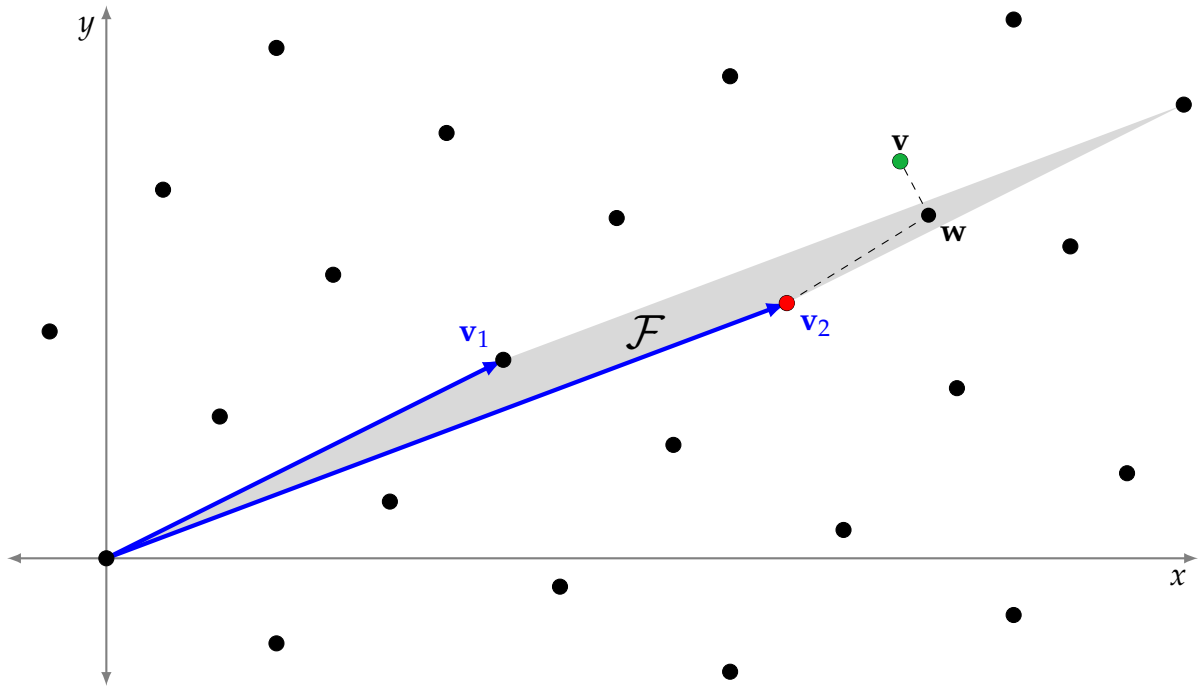


Figure 22.: Here we see that Babai's algorithm does not work when the basis is bad. The closest vector in the lattice to $\mathbf{w}$ is $\mathbf{v}$, however this is not a vertex of the translation of the fundamental domain that contains $\mathbf{w}$ (which in this case is just $\mathcal{F}$). Instead, the algorithm will return $\mathbf{v}_2$, which is incorrect.

We want to use Babai's algorithm to find the vector $\mathbf{w} = (53172, 81743)$. The first step is to express $\mathbf{w}$ in terms of the basis. This means we need to find $t_1, t_2 \in \mathbb{R}$ such that $\mathbf{w} = t_1\mathbf{v}_1 + t_2\mathbf{v}_2$. We see that by plugging in our vectors we get the linear equation

$$(53172, 81743) = (t_1, t_2)\begin{pmatrix} 137 & 312 \\ 215 & -187 \end{pmatrix}$$

It is easy to solve this by simply inverting this matrix and multiplying it by $\mathbf{w}$. This gives us the values $t_1 \approx 296.85$ and $t_2 \approx 58.15$. Babai's algorithm then says to round t_1 and t_2 to the nearest integer, and computing

$$\mathbf{v} = \lfloor t_1 \rceil \mathbf{v}_1 + \lfloor t_2 \rceil \mathbf{v}_2 = 297(137, 312) + 58(215, -187) = (53159, 81818)$$

It is clear that $\mathbf{v}$ is in $\mathcal{L}$, and so we compute

$$\|\mathbf{w} - \mathbf{v}\| \approx 76.12$$

Given that $\|\mathbf{v}_1\| = 340.75$ and $\|\mathbf{v}_2\| = 284.95$, we see that $\mathbf{v}$ is reasonably close to $\mathbf{w}$ and solves apprCVP for a reasonable factor.

If, however, we repeat the measurement with the new basis $\mathbf{v}_1' = 5\mathbf{v}_1 + 6\mathbf{v}_2 = (1975, 438)$ and $\mathbf{v}_2' = 19\mathbf{v}_1 + 23\mathbf{v}_2 = (7548, 1627)$, we will get a very different result.

Using the Hadamard ratio for this new basis,

$$\mathcal{H}(\mathbf{v}_1', \mathbf{v}_2') = \left(\frac{\det(\mathcal{L})}{\|\mathbf{v}_1'\|\|\mathbf{v}_2'\|}\right)^{1/2} = \left(\frac{92699}{(2022.99)(7721.36)}\right)^{1/2} \approx 0.077$$

we see that this basis is not very orthogonal, and so we expect Babai's algorithm to output a vector much further away from $\mathbf{w}$.

Following the same process as before, we get the linear equation

$$(53172, 81743) = (t_1', t_2')\begin{pmatrix} 1975 & 438 \\ 7548 & 1627 \end{pmatrix}$$

which gives the solution $t_1' \approx 5722.66$ and $t_2' \approx -1490.34$. Therefore Babai's algorithm outputs the vector

$$\mathbf{v}' = 5723\mathbf{v}_1' - 1490\mathbf{v}_2' = (56405, 82444)$$

We now check

$$\|\mathbf{w} - \mathbf{v}'\| \approx 3308.12$$

which is substantially larger than our previous calculation. This means that the algorithm found a vector that is much further vector from $\mathbf{w}$ than when the basis was more orthogonal.

11.5 THE GGH PUBLIC KEY CRYPTOSYSTEM

The notion of a good and bad basis, along with Babai's algorithm, are the basis[4] of the GGH cryptosystem.

- **Key Creation:** Alice chooses a set of of linearly independent, reasonably orthogonal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. One easy way to do this is to choose a random integer d, and choose the elements of the vectors by randomly switching between $\pm d$. She can compute the Hadamard ratio for the resulting basis to ensure that it is reasonably orthogonal. If it is not, then she has been quite unlucky, and she tries again. Let V be the matrix with rows $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

  Next, Alice chooses a unimodular matrix U. Again, an easy way to do this is to multiply a large number of random elementary matrices together. She then computes $W = UV$. by Theorem 11.2.7, the rows of W are another basis for $\mathcal{L}$ that is very likely to be highly non-orthogonal, simply because most bases are. If this is not the case, then again she has been quite unlucky and she tries again with a new matrix U. The good basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ is her private key, while the bad basis $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ is her public key.

4 See what I did there?

- **Encryption:** If Bob wants to send a message to Alice, he writes his message in the form of a small lattice vector $\mathbf{m}$. This might be a binary vector, but any small lattice vector will do. He then also selects a small random perturbation vector $\mathbf{r}$ that is not in the lattice. An easy way to do this is to choose the coordinates of $\mathbf{r}$ randomly between $\pm\delta$, where δ is a small real number. Note that $\mathbf{r} \in \mathcal{F}$, although $\mathbf{r}$ we require that $\mathbf{r}$ be smaller than just some arbitrary vector in the fundamental domain. He then computes the ciphertext

$$\mathbf{c} = \mathbf{m}W + \mathbf{r} = \sum_{i=1}^{n} (m_i \mathbf{w}_i) + \mathbf{r}$$

  Notice that while $\mathbf{c}$ is not a lattice point, it is very close to the lattice point $\mathbf{m}W$ if $\mathbf{r}$ is small enough. Bob then sends $\mathbf{c}$ to Alice.

- **Decryption:** Decryption is pretty straightforward. Since she has a good basis for the lattice, she simply performs Babai's algorithm to find the closest vector to $\mathbf{c}$. Assuming the parameters have all been chosen correctly, this will be $\mathbf{v} = \mathbf{m}W$. She then computes $\mathbf{v}W^{-1}$ to recover Bob's message $\mathbf{m}$.

Remark 11.5.1. One variation on GGH switches $\mathbf{m}$ and $\mathbf{r}$, so that Bob encrypts his message by calculating $\mathbf{c} = \mathbf{r}W + \mathbf{m}$. In this case, the vector Alice calculates from Babai's algorithm is actually $\mathbf{r}W$, and she can then find $\mathbf{m}$ by calculating $\mathbf{m} = \mathbf{c} - \mathbf{r}W$.

Example 11.5.2. Suppose Alice choose the good basis $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ for $\mathbf{v}_1 = (-97, 19, 19)$, $\mathbf{v}_2 = (-36, 30, 86)$, and $\mathbf{v}_3 = (-184, -64, 78)$. We can ensure that this is a relatively orthogonal basis by computing the Hadamard ratio

$$\mathcal{H}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) = \left(\frac{\det(\mathcal{L})}{\|\mathbf{v}_1\| \, \|\mathbf{v}_2\| \, \|\mathbf{v}_3\|} \right)^{1/3} \approx 0.746$$

While this certainly is not the most orthogonal basis, it will do, especially in just three dimensions. Suppose Alice comes up with the unimodular matrix

$$U = \begin{pmatrix} 4327 & -15447 & 23454 \\ 3297 & -11770 & 17871 \\ 5464 & -19506 & 29617 \end{pmatrix}$$

This allows her to compute her public bad basis

$$\begin{aligned} W = UV &= \begin{pmatrix} 4327 & -15447 & 23454 \\ 3297 & -11770 & 17871 \\ 5464 & -19506 & 29617 \end{pmatrix} \begin{pmatrix} -97 & 19 & 19 \\ -36 & 30 & 86 \\ -184 & -64 & 78 \end{pmatrix} \\ &= \begin{pmatrix} -4179163 & -1882253 & 583183 \\ -3184353 & -1434201 & 444361 \\ -5277320 & -2376852 & 736426 \end{pmatrix} \end{aligned}$$

which leads to the bad basis $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$ for $\mathbf{w}_1 = (-4179163, -1882253, 583183)$, $\mathbf{w}_2 = (-3184353, -1434201, 444361)$, and $\mathbf{w}_3 = (-5277320, -2376852, 736426)$.

We can again ensure that this basis is bad by computing the Hadamard ratio

$$\mathcal{H}(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) = \left(\frac{\det(\mathcal{L})}{\|\mathbf{w}_1\| \, \|\mathbf{w}_2\| \, \|\mathbf{w}_3\|} \right)^{1/3} \approx 0.0000208$$

which is clearly an extremely non-orthogonal basis.

Now suppose Bob wants to send the message $\mathbf{m} = (86, -35, -32)$ to Alice. He generates the small random perturbation vector $\mathbf{r} = (-4, -3, 2)$ and computes the ciphertext

$$\begin{aligned} \mathbf{c} = \mathbf{m}W + \mathbf{r} &= (86, -35, -32) \begin{pmatrix} -4179163 & -1882253 & 583183 \\ -3184353 & -1434201 & 444361 \\ -5277320 & -2376852 & 736426 \end{pmatrix} + (-4, -3, 2) \\ &= (-79081427, -35617462, 11035473) \end{aligned}$$

Upon receiving $\mathbf{c}$ from Bob, Alice sets about using Babai's algorithm to solve for $\mathbf{m}$. She first writes $\mathbf{c}$ in terms of the good basis

$$\mathbf{c} \approx 81878.97\mathbf{v}_1 - 292300.00\mathbf{v}_2 + 443815.04\mathbf{v}_3$$

Rounding each coefficient, she finds that Babai's algorithm outputs the vector

$$\mathbf{v} = 81879\mathbf{v}_1 - 292300\mathbf{v}_2 + 443815\mathbf{v}_3 = (-79081423, -35617459, 11035471)$$

Since she used a reasonably good basis, we expect that $\mathbf{v} = \mathbf{m}W$. Therefore to find the message $\mathbf{m}$, we simply need to calculate $\mathbf{v}W^{-1}$. This is equivalent to writing $\mathbf{v}$ in terms of the bad basis and reading off the coefficients. In other words, since we can find that $\mathbf{v} = 86\mathbf{w}_1 - 35\mathbf{w}_2 - 32\mathbf{w}_3$, we see that $\mathbf{m} = (86, -35, -32)$. This is the message sent by Bob, and so the cryptosystem worked!

Remark 11.5.3. Suppose that Eve wanted to break the above example. Since she does not have access to the good basis, she would have to use Babai's algorithm with the bad basis. She first writes $\mathbf{c}$ in terms of the bad basis

$$\mathbf{c} \approx 75.76\mathbf{w}_1 - 34.52\mathbf{w}_2 - 24\mathbf{w}_3$$

Rounding each coefficient to the nearest integer, she finds that Babai's algorithm outputs the vector

$$\mathbf{v}' = 75\mathbf{w}_1 - 35\mathbf{w}_2 - 24\mathbf{w}_3 = (-79508353, -35809745, 11095049)$$

This outputs the message $\mathbf{m}' = (76, -35, 24)$, which is not correct.

To see why this happened we simply check to see how well Babai's algorithm performed on the two bases. For the good basis, Babai's algorithm found a vector very close to $\mathbf{c}$,

$$\|\mathbf{c} - \mathbf{v}\| \approx 5.3852$$

while for the bad basis, the algorithm found a vector very far from $\mathbf{c}$

$$\|\mathbf{c} - \mathbf{v}'\| \approx 472000$$

Therefore it is clear that the security of GGH relies on Eve's ability to construct a good basis for a littice, given a bad one.

11.6 LATTICE REDUCTION ALGORITHMS

We have now seen multiple lattice cryptosystems whose security relies on some variation of SVP or CVP. Since these problems are easy to solve with a good basis for the lattice, we can extend the problem to finding a good basis for a given lattice. In this section, we describe the algorithms that do this, called Lattice Reduction Algorithms. In particular, we discuss the LLL algorithm, which is the most famous lattice reduction algorithm, and its aplications to Merkle-Helman and GGH, as well as its limitations.

11.6.1 *The Gaussian Lattice Reduction Algorithm in Dimension Two*

Before we talk about LLL, we introduce the concept of a lattice reduction algorithm by defining the Gaussian lattice reduction algorithm, which finds a good basis for a 2-dimensional lattice.

The general idea is to subtract multiples of one basis vector from the other until the basis cannot be improved. The naïve way to do this would be to follow in the footsteps of the Gram-Schmidt algorithm. If we assume that $\|\mathbf{v}_1\| < \|\mathbf{v}_2\|$, then we would simply replace $\mathbf{v}_2$ with the vector

$$\mathbf{v}_2^* = \mathbf{v}_2 - \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|^2}\mathbf{v}_1$$

The Gram-Schmidt algorithm proves that $\mathbf{v}_2^*$ is orthogonal to $\mathbf{v}_1$. $\mathbf{v}_2^*$ is called the projection of $\mathbf{v}_2$ onto the orthogonal compliment of $\mathbf{v}_1$, as shown in Figure 23.

I say the naïve way because this will rarely work for lattices. We are not guaranteed that $\mathbf{v}_1 \cdot \mathbf{v}_2 / \|\mathbf{v}_1\|^2$ is an integer, and so $\mathbf{v}_2^*$ is not guaranteed to be in the lattice. If, instead, we round this term to the nearest integer, we are guaranteed to get a lattice vector, and it should still be reasonably orthogonal to $\mathbf{v}_1$. This is formalized in the following theorem.

Theorem 11.6.1 (Gaussian Lattice Reduction)**.** *Let $\mathcal{L} \subset \mathbb{R}^2$ be a 2-dimensional lattice with basis $\{\mathbf{v}_1, \mathbf{v}_2\}$. Then the following algorithm returns a good basis for $\mathcal{L}$:*

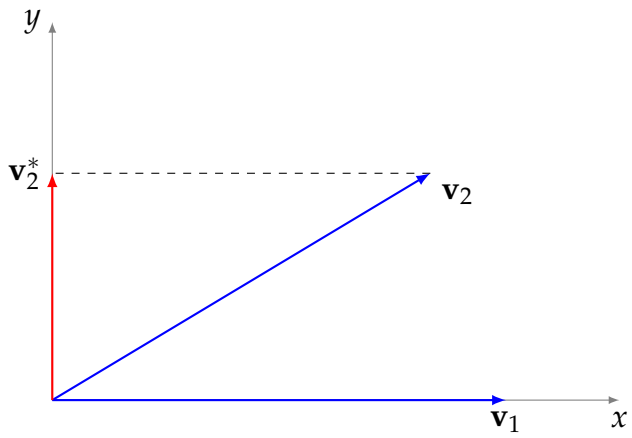


Figure 23.: This is an illustration of the projection of $\mathbf{v}_2$ onto the orthogonal compliment of $\mathbf{v}_1$.

1. *If* $\|\mathbf{v}_2\| < \|\mathbf{v}_1\|$, *swap* $\mathbf{v}_1$ *and* $\mathbf{v}_2$.
2. *Compute* $m = \left\lfloor \mathbf{v}_1 \cdot \mathbf{v}_2 / \|\mathbf{v}_1\|^2 \right\rceil$.
3. *If* $m = 0$, *return the basis vectors* $\mathbf{v}_1$ *and* $\mathbf{v}_2$.
4. *Otherwise, replace* $\mathbf{v}_2$ *with* $\mathbf{v}_2 - m\mathbf{v}_1$ *and repeat.*

More precisely, when the algorithm terminates, $\mathbf{v}_1$ *is a shortest non-zero vector in* $\mathcal{L}$*, so the algorithm solves SVP. Further, the angle* θ *between* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *satisfies* $|\cos\theta| \leq \|\mathbf{v}_1\| / 2\|\mathbf{v}_2\|$*, and therefore* $\frac{\pi}{3} \leq \theta \leq \frac{2\pi}{3}$.

Proof. First we show that after the algorithm terminates, $\mathbf{v}_1$ is a shortest vector. Once the algorithm has terminated, it will return the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ such that $\|\mathbf{v}_2\| \geq \|\mathbf{v}_1\|$ and that

$$\frac{|\mathbf{v}_1 \cdot \mathbf{v}_2|}{\|\mathbf{v}_1\|^2} \leq \frac{1}{2}$$

since this means that the algorithm cannot make $\mathbf{v}_2$ any smaller by subtracting multiples of $\mathbf{v}_1$ (m rounds to 0). Now suppose that $\mathbf{v} \in \mathcal{L}$ is some non-zero lattice vector. We can

write it in terms of the basis $\mathbf{v} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2$ for some integers a_1 and a_2. Now we see that the length of this vector is given by

$$\begin{aligned}
\|\mathbf{v}\|^2 &= \|a_1\mathbf{v}_1 + a_2\mathbf{v}_2\| \\
&= a_1^2\|\mathbf{v}_1\|^2 + 2a_1a_2(\mathbf{v}_1\cdot\mathbf{v}_2) + a_2^2\|\mathbf{v}_2\|^2 \\
&\geq a_1^2\|\mathbf{v}_1\|^2 - 2|a_1a_2||\mathbf{v}_1\cdot\mathbf{v}_2| + a_2^2\|\mathbf{v}_2\|^2 \\
&\geq a_1^2\|\mathbf{v}_1\|^2 - |a_1a_2|\,\|\mathbf{v}_1\|^2 + a_2^2\|\mathbf{v}_2\|^2 \qquad \left(\text{from } \frac{|\mathbf{v}_1\cdot\mathbf{v}_2|}{\|\mathbf{v}_1\|^2} \leq \frac{1}{2}\right) \\
&\geq a_1^2\|\mathbf{v}_1\|^2 - |a_1a_2|\,\|\mathbf{v}_1\|^2 + a_2^2\|\mathbf{v}_1\|^2 \qquad (\text{from } \|\mathbf{v}_2\| \geq \|\mathbf{v}_1\|) \\
&= (a_1^2 - |a_1|\,|a_2| + a_2^2)\,\|\mathbf{v}_1\|^2
\end{aligned}$$

Note that for any real numbers t_1 and t_2, the quantity

$$t_1^2 - t_1t_2 + t_2^2 = \left(t_1 - \frac{1}{2}t_2\right)^2 + \frac{3}{4}t_2^2 = \frac{3}{4}t_1^2 + \left(\frac{1}{2}t_1 - t_2\right)^2$$

is zero if and only if $t_1 = t_2 = 0$. Since we assumed that $\mathbf{v}$ was a non-zero lattice vector, a_1 and a_2 cannot be zero, and therefore $\|\mathbf{v}\|^2 \geq \|\mathbf{v}_1\|^2$, and therefore $\mathbf{v}_1$ is a shortest non-zero vector in $\mathcal{L}$.

Now to prove the restriction on the angle θ, note that since

$$\frac{|\mathbf{v}_1\cdot\mathbf{v}_2|}{\|\mathbf{v}_1\|^2} \leq \frac{1}{2}$$

we can say that $|\mathbf{v}_1\cdot\mathbf{v}_2| \leq \|\mathbf{v}_1\|^2/2$. By Theorem 11.1.10, this implies that

$$\|\mathbf{v}_1\|\,\|\mathbf{v}_2\|\,|\cos\theta| \leq \frac{\|\mathbf{v}_1\|^2}{2}$$

and rearranging gives us the desired result $|\cos\theta| \leq \|\mathbf{v}_1\|/2\,\|\mathbf{v}_2\|$. □

Example 11.6.2. Consider the lattice generated by the vectors $\mathbf{v}_1 = (66586820, 65354729)$ and $\mathbf{v}_2 = (6513996, 6393464)$. First, we compute $\|\mathbf{w}_1\|^2 \approx 8.71 \times 10^{15}$ and $\|\mathbf{v}_2\|^2 \approx 8.33 \times 10^{13}$. Since $\mathbf{v}_2$ is shorter than $\mathbf{v}_1$, we switch the two basis vectors so that $\mathbf{v}_1 = (6513996, 6393464)$ and $\mathbf{v}_2 = (66586820, 65354729)$.

Now we compute

$$m = \left\lceil \frac{\mathbf{v}_1\cdot\mathbf{v}_2}{\|\mathbf{v}_1\|^2} \right\rfloor = \lfloor 10.2221 \rceil = 10$$

and replace $\mathbf{v}_2$ with

$$\mathbf{v}_2' = \mathbf{v}_2 - m\mathbf{v}_1 = (1446860, 1420089)$$

Now we repeat. Since $\|\mathbf{v}_2'\| < \|\mathbf{v}_1\|$, we switch the two vectors, and calculate the new value of m. To cut back on repetition, we summarize the steps of this process in Table 3.

Step	$\mathbf{v}_1$	$\mathbf{v}_2$	m
1	$(6513996, 6393464)$	$(66586820, 65354729)$	10
2	$(1446860, 1420089)$	$(6513996, 6393464)$	5
3	$(-720304, -706981)$	$(1446860, 1420089)$	-2
4	$(6252, 6127)$	$(-720304, -706981)$	-115
5	$(-1324, -2376)$	$(6252, 6127)$	-3
6	$(2280, -1001)$	$(-1324, -2376)$	0

Table 3.: The steps of the Gaussian Lattice Reduction Algorithm for this example.

In the end, the algorithm returns the basis vectors $\mathbf{v}_1 = (2280, -1001)$ and $\mathbf{v}_2 = (-1324, -2376)$. To ensure that this is a good basis for the lattice, we calculate the Hadamard ratio

$$\mathcal{H}(\mathbf{v}_1, \mathbf{v}_2) = \left(\frac{\det(\mathcal{L})}{\|\mathbf{v}_1\| \, \|\mathbf{v}_2\|}\right)^{1/2} \approx \left(\frac{6742604}{(2490)(2720)}\right)^{1/2} \approx 0.998$$

which means that this is a very good basis indeed.

Remark 11.6.3. While this algorithm works very well for 2-dimensional lattices, it does not work for anything higher. However it does serve as a good introduction to the more complicated LLL algorithm that follows.

11.6.2 *The LLL Lattice Reduction Algorithm*

Similar to the Gaussian lattice reduction, LLL is essentially based on converting Gram-Schmidt into a lattice algorithm. If you recall from Theorem 11.1.15 that the Gram-Schmidt algorithm returns an orthogonal basis for the same vector space as our original basis, we have already mentioned that this is not necessarily a basis for the same lattice. We can still relate these two bases in a lattice context, however, through the following theorem.

Theorem 11.6.4. *Let $B = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be the basis for a lattice $\mathcal{L}$, and let $B^*\{\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*\}$ be the associated Gram-Schmidt orthogonal basis. Then*

$$\det(\mathcal{L}) = \prod_{i=1}^{n} \|\mathbf{v}_i^*\|$$

Proof. Let $F = F(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ be the matrix described in Theorem 11.2.18, and let $F^* = F(\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*)$ be the analogous matrix for the Gram-Schmidt basis. Note that since B^* is an orthogonal basis,

$$\left|\det(F^*)\right| = \prod_{i=1}^{n} \|\mathbf{v}_i^*\|$$

By the definition of the Gram-Schmidt algorithm, we know that

$$\mathbf{v}_i^* = \mathbf{v}_i - \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{v}_j^* \quad \text{where} \quad \mu_{i,j} = \frac{\mathbf{v}_i \cdot \mathbf{v}_j^*}{\|\mathbf{v}_j^*\|} \quad \text{for} \quad 1 \leq j \leq i-1$$

It then follows that $F = MF^*$ where M is defined as the following matrix:

$$M = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ \mu_{2,1} & 1 & 0 & \cdots & 0 & 0 \\ \mu_{3,1} & \mu_{3,2} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mu_{n-1,1} & \mu_{n-1,2} & \mu_{n-1,3} & \cdots & 1 & 0 \\ \mu_{n,1} & \mu_{n,2} & \mu_{n,3} & \cdots & \mu_{n,n-1} & 1 \end{pmatrix}$$

Since M is a lower diagonal matrix with 1's on the diagonal, we recall from linear algebra that this implies $\det(M) = 1$. By Theorem 11.2.18 and properties of the determinant, we find that

$$\det(\mathcal{L}) = \Big|\det(F)\Big| = \Big|\det(MF^*)\Big| = \Big|\det(M)\det(F^*)\Big| = \Big|\det(F^*)\Big| = \prod_{i=1}^{n} \|\mathbf{v}_i^*\|$$

Thus the proposition is true. □

Similar to the 2-dimensional Gaussian lattice reduction, we can use the notion of projecting onto the orthogonal compliment to explain the idea behind the Gram-Schmidt. In this case, we say that $\mathbf{v}_i^*$ is the projection of $\mathbf{v}_i$ onto $\text{Span}(\mathbf{v}_1, \ldots, \mathbf{v}_n)^\perp$.

As you may have noticed, throughout much of this chapter, we have used the notion of a good/bad basis, although the concept has not been very well defined. Well there is a reason for this – there is no one definition. A good basis is more orthogonal than a bad basis, but the boundary adapts to the situation at hand. For example, the definition of a good basis for Babai's algorithm may not be the same as for the Gaussian lattice reduction algorithm. Up until now, this has not been a particularly important distinction, however the LLL algorithm requires a very specific definition.

Definition 11.6.5. Let $B = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be a basis for a lattice $\mathcal{L}$, and let $B^* = \{\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*\}$ be the associated Gram-Schmidt orthogonal basis. The basis B is said to be *LLL reduced* if it satisfies the following properties:

$$\textbf{(Size Condition)} \qquad |\mu_{i,j}| = \frac{|\mathbf{v}_i \cdot \mathbf{v}_j^*|}{\|\mathbf{v}_j^*\|^2} \leq \frac{1}{2} \quad \text{for all} \quad 1 \leq j < i \leq n$$

$$\textbf{(Lovász Condition)} \qquad \|\mathbf{v}_i^*\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right)\|\mathbf{v}_{i-1}^*\|^2 \quad \text{for all} \quad 1 < i \leq n$$

Remark 11.6.6. There are multiple ways to state the Lovász condition. One equivalent statement is

$$\|\mathbf{v}_i^* + \mu_{i,i-1}\mathbf{v}_{i-1}^*\|^2 \geq \frac{3}{4}\|\mathbf{v}_{i-1}^*\|^2$$

and another one is to say that

$$\Big\|\text{Projection of } \mathbf{v}_i \text{ onto } \text{Span}(\mathbf{v}_1,\ldots,\mathbf{v}_{i-2})^\perp\Big\| \geq \frac{3}{4}\Big\|\text{Projection of } \mathbf{v}_{i-1} \text{ onto } \text{Span}(\mathbf{v}_1,\ldots,\mathbf{v}_{i-2})^\perp\Big\|$$

The following theorem describes some crucial properties of LLL reduced bases.

Theorem 11.6.7. *Let $\mathcal{L}$ be a lattice of dimension n. Any LLL reduced basis $\{\mathbf{v}_1,\ldots,\mathbf{v}_n\}$ for $\mathcal{L}$ has the following two properties:*

$$\prod_{i=1}^{n}\|\mathbf{v}_i\| \leq 2^{n(n-1)/4}\det(\mathcal{L})$$

$$\|\mathbf{v}_j\| \leq 2^{(i-1)/2}\|\mathbf{v}_i^*\| \quad \textit{for all} \quad 1 \leq j \leq i \leq n$$

Further, the initial vector in an LLL reduced basis satisfies

$$\|\mathbf{v}_1\| \leq 2^{(n-1)/4}\big|\det(\mathcal{L})\big|^{1/n} \qquad \textit{and} \qquad \|\mathbf{v}_1\| \leq 2^{(n-1)/2}\min_{0\neq\mathbf{v}\in\mathcal{L}}\|\mathbf{v}\|$$

Thus an LLL reduced basis solves apprSVP to within a factor of $2^{(n-1)/2}$.

Now we can finally define the LLL algorithm.

Theorem 11.6.8 (LLL Algorithm)**.** *Given a basis for a lattice $\{\mathbf{v}_1,\ldots,\mathbf{v}_n\}$ and the associated Gram-Schmidt orthogonal basis $\{\mathbf{v}_1^*,\ldots,\mathbf{v}_n^*\}$, the following algorithm returns a reasonably LLL reduced basis for $\mathcal{L}$:*

1. *Set $k = 2$, $\mathbf{w}_1 = \mathbf{v}_1$, and $\mathbf{v}_1^* = \mathbf{v}_1$.*
2. *While $k \leq n$,*

   Set $\mathbf{w}_k = \mathbf{v}_k - \lfloor\mu_{k,j}\rceil\,\mathbf{v}_j^*$ for all $1 \leq j \leq k-1$

   If $\|\mathbf{v}_k^*\|^2 \geq \left(\frac{3}{4} - \mu_{k,k-1}^2\right)\|\mathbf{v}_{k-1}^*\|^2$, then set $k = k+1$.

   Otherwise, swap $\mathbf{v}_{k-1}$ and $\mathbf{v}_k$ and set $k = \max(k-1, 2)$.
3. *The basis $\{\mathbf{w}_1,\ldots,\mathbf{w}_n\}$ is an LLL reduced basis for $\mathcal{L}$.*

For a proof of this algorithm, see [1], [41].

Example 11.6.9. We now demonstrate the LLL algorithm on the 6-dimensional lattice $\mathcal{L}$ generated by the rows of the following matrix:

$$M = \begin{pmatrix} 19 & 2 & 32 & 46 & 3 & 33 \\ 15 & 42 & 11 & 0 & 3 & 24 \\ 43 & 15 & 0 & 24 & 4 & 16 \\ 20 & 44 & 44 & 0 & 18 & 15 \\ 0 & 48 & 35 & 16 & 31 & 31 \\ 48 & 33 & 32 & 9 & 1 & 29 \end{pmatrix}$$

The smallest vector in this basis is $\mathbf{v}_2 = (15, 42, 11, 0, 3, 24)$, which has length $\|\mathbf{v}_2\| \approx 51.913$. The output from the algorithm gives the matrix

$$M^{LLL} = \begin{pmatrix} 7 & -12 & -8 & 4 & 19 & 9 \\ -20 & 4 & -9 & 16 & 13 & 16 \\ 5 & 2 & 33 & 0 & 15 & -9 \\ -6 & -7 & -20 & -21 & 8 & -12 \\ -10 & -24 & 21 & -15 & -6 & -11 \\ 7 & 4 & -9 & -11 & 1 & 31 \end{pmatrix}$$

We check that both matrices have the same determinant, which suggests that they are the basis for the same lattice, as discussed in Corollary 11.2.21. In fact they do

$$\det(M) = \det(M^{LLL}) = \pm 777406251$$

Further, we check the Hadamard ratio of the two basis to ensure that the algorithm has improved the orthogonality. Again, we see that the algorithm was successful, since

$$\mathcal{H}(M) = 0.46908 \leq \mathcal{H}(M^{LLL}) = 0.88824$$

FInally, we see that the shortest vector of the new basis is $\|\mathbf{v}_1\| = 26.739$ which, again, shows that the algorithm has worked.

We have already seen that an LLL reduced basis can solve apprSVP, but now we can see that we can also solve apprCVP to within a factor of C^n for some constant C.

Theorem 11.6.10 (LLL apprCVP Algorithm)**.** *There is a constant C such that for any lattice $\mathcal{L}$ of dimension n given by a basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, the following algorithm solves apprCVP to within a factor of C^n.*

1. *Apply LLL to the basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ to find an LLL reduced basis.*
2. *Apply Babai's algorithm using the LLL reduced basis.*

Remark 11.6.11. LLL works great in relatively low dimensions, but gets less accurate as the dimension increases. There are a few variations of the algorithm that improve the accuracy, but at the sake of computational time and memory.

Applying LLL to Knapsack Problems

As we saw at the end of Section 11.3, we can reformulate a subset-sum problem (H, S) with the matrix

$$\begin{pmatrix} 2 & 0 & \cdots & 0 & h_1 \\ 0 & 2 & \cdots & 0 & h_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 2 & h_n \\ 1 & 1 & \cdots & 1 & S \end{pmatrix}$$

Then the message from Bob will likely be the shortest vector in the lattice generated by the rows of this matrix.

We now show how to use the LLL algorithm to solve the subset-sum problem. Recall that in order to break the system, Eve must solve the non-superincreasing sequence subset-sum problem. Consider such a problem where $H = (89, 243, 212, 150, 245)$ and $S = 546$. First, we create the matrix

$$A_{H,S} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 89 \\ 0 & 2 & 0 & 0 & 0 & 243 \\ 0 & 0 & 2 & 0 & 0 & 212 \\ 0 & 0 & 0 & 2 & 0 & 150 \\ 0 & 0 & 0 & 0 & 2 & 245 \\ 1 & 1 & 1 & 1 & 1 & 546 \end{pmatrix}$$

where we have said that the rows of this matrix are the basis of a lattice. Now we run LLL on this basis to get the matrix

$$\begin{pmatrix} -1 & 1 & -1 & 1 & -1 & 0 \\ 1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 & 2 \\ 1 & -1 & -1 & -1 & -1 & 2 \\ -2 & -2 & 4 & 0 & -2 & 0 \\ -6 & -4 & -6 & -6 & 0 & -3 \end{pmatrix}$$

We know from the LLL algorithm that the first vector in the reduced basis is the short vector, and so we write $\mathbf{v}_1' = (-1, 1, -1, 1, -1, 0)$. Now we express this in terms of the rows of the matrix $A_{H,S}$

$$(-1, 1, -1, 1, -1, 0) = (-1, 0, -1, 0, -1, 1)A_{H,S}$$

This gives us a solution to the subset-sum problem

$$-89 - 212 - 245 + 546 = 0 \;\Rightarrow\; 89 + 212 + 245 = 546$$

Applying LLL to GGH

Now we show how LLL can be applied to the GGH cryptosystem by continuing Example 11.6.2. Recall that Alice's public key is the matrix

$$W = \begin{pmatrix} -4179163 & -1882253 & 583183 \\ -3184353 & -1434201 & 444361 \\ -5277320 & -2376852 & 736426 \end{pmatrix}$$

and that Bob's encrypted message is $\mathbf{c} = (-79081427, -35617462, 11035473)$.

In order to break this cryptosystem, Eve needs to find a vector in $\mathcal{L}$ that is close to $\mathbf{c}$. She first applies LLL to W to obtain the matrix

$$\begin{pmatrix} 36 & -30 & -86 \\ 61 & 11 & 67 \\ -10 & 102 & -40 \end{pmatrix}$$

This basis has a Hadamard ratio of $\mathcal{H} \approx 0.956$, which is even better than Alice's private basis. Therefore Eve can simply use Babai's algorithm to find the close lattice vector $\mathbf{v} = (79081423, 35617459, -11035471)$ that is very close to $\mathbf{c}$. Now she writes $\mathbf{v}$ in terms of the W basis to find $\mathbf{v} = -86\mathbf{w}_1 + 35\mathbf{w}_2 + 32\mathbf{w}_3$. Thus, Eve has found Bob's message $\mathbf{m} = (-86, 35, 32)$.

11.7 OTHER LATTICE CRYPTOSYSTEMS

In this chapter, we have barely scratched the surface of what is possible with lattices and cryptography. But before we finish,I would like to mention one of the most exciting potential applications of lattices in cryptology: Fully Homomorphic Encryption.

A fully homomorphic encryption scheme is an encryption scheme just like any other, however it has the property that you can apply functions to the encrypted messages, and get the same result as if you had decrypted the message, applied the function, and then re-encrypted the result. To formalize this, consider the following definition.

Definition 11.7.1 (Fully Homomorphic Encryption Scheme)**.** Suppose that for some cryptosystem, you have an arbitrary plaintext/ciphertext pair (m, c). Let E be the encryption function and D be the decryption function for the system. Then the system is fully homomorphic if for any well-defined function,

$$f(c) = E\Big(f\big(D(m)\big)\Big)$$

without giving up any private information about the system.

Remark 11.7.2. Just about every cryptosystem is *partially homomorphic*, that is, the property hold for *specific* functions, but not arbitrary ones. So for example, given two encrypted RSA messages c_1 and c_2, we can calculate

$$c_1 c_2 = m_1^e m_2^e = (m_1 m_2)^e)$$

without leaking information about the system. But this does not work for just any function.

Recently, researchers have been discovering ways to create lattice cryptosystems that are fully homomorphic. The only remaining question is if these systems are still secure when we add this fully homomorphic property. But if they are proven secure, then we will be able to implement lattice-based fully homomorphic encryption schemes.

So why does this matter? Well, fully homomorphic encryption has been considered the holy grail of cryptography essentially since its creation. Using a fully homomorphic system, we could communicate with banks and other secure servers without ever decrypting the sensitive information. This would also be useful in the world of digital signatures. Essentially anytime you don't need to know exactly what the information is, just that it is correct, or to alter it, fully homomorphic encryption is the answer.

To see why this is so important, consider the following scenario. You have a very sensitive message that you need to send, and so you painstakingly set up the worlds greatest cryptosystem ever created. Confident in the security of your system, you begin communicating your very sensitive information. Meanwhile, one of your interns accidentally downloaded some malware onto your servers that allowed a hacker to walk right in and steal the decrypted messages sitting on your servers.

That would suck. Like a lot. And it is not as unrealistic as you might think. Recently, researchers found that over half of USB drives found in corporate parking lots are plugged into company computers [42]. However if you only ever need the encrypted messages, then even a hacked computer will still be secure. How awesome is that?

A

FREQUENCY TABLES

Frequency of letters in English text, by decreasing frequency:

E	13.11%	M	2.54%
T	10.47%	U	2.46%
A	8.15%	G	1.99%
O	8.00%	Y	1.98%
N	7.10%	P	1.98%
R	6.83%	W	1.54%
I	6.35%	B	1.44%
S	6.10%	V	0.92%
H	5.26%	K	0.42%
D	3.79%	X	0.17%
L	3.39%	J	0.13%
F	2.92%	Q	0.12%
C	2.76%	Z	0.08%

Most common English bigrams (frequency per 1000 words)

th	he	an	re	er	in	on	at	nd	st	es	en	of	te	ed
168	132	92	91	88	86	71	68	61	53	52	51	49	46	46

Most common English double letters

L	S	E	O	T

B

ADDITIONAL WORK

Example B.0.1 (Continuing from Example 3.5.7). Next, consider the pair of congruences $x \equiv -1 \mod 59$ and $x \equiv -8 \mod 71$. Similarly, we see that $x_1 = -1$ is a solution to the first congruence, and then $x_2 = -1 + 59y$ is the general solution. Substituting x_2 into the second congruence, we find

$$\begin{aligned} -1 + 59y &\equiv -8 \mod 71 \\ 59y &\equiv -7 \mod 71 \\ y &\equiv 59^{-1} \cdot -7 \mod 71 \\ y &\equiv 65 \cdot -7 \mod 71 \\ y &\equiv 42 \mod 71 \end{aligned}$$

Substituting back into x_2, we know by the Chinese Remainder Theorem that $x_2 = -1 + (59 \cdot 42) = 2477 \mod 4189$ is the unique solution to our pair of simultaneous congruences.

Now, consider the pair of congruences $x \equiv -1 \mod 59$ and $x \equiv 8 \mod 71$. Using the same process, $x_1 = -1$ is a solution to the first congruence, and then $x_2 = -1 + 59y$ is the general solution. Substituting x_2 into the second congruence, we find

$$\begin{aligned} -1 + 59y &\equiv 8 \mod 71 \\ 59y &\equiv 9 \mod 71 \\ y &\equiv 59^{-1} \cdot 9 \mod 71 \\ y &\equiv 65 \cdot 9 \mod 71 \\ y &\equiv 17 \mod 71 \end{aligned}$$

Substituting back into x_2, we know by the Chinese Remainder Theorem that $x_2 = -1 + (59 \cdot 17) = 1002 \mod 4189$ is the unique solution to our pair of simultaneous congruences.

Lastly, consider the congruences $x \equiv 1 \mod 59$ and $x \equiv -8 \mod 71$. Using the same process, $x_1 = 1$ is a solution to the first congruence, and then $x_2 = 1 + 59y$ is the general solution. Substituting x_2 into the second congruence, we find

$$\begin{aligned} 1 + 59y &\equiv -8 \mod 71 \\ 59y &\equiv -9 \mod 71 \\ y &\equiv 59^{-1} \cdot -9 \mod 71 \\ y &\equiv 65 \cdot -9 \mod 71 \\ y &\equiv 54 \mod 71 \end{aligned}$$

Substituting back into x_2, we know by the Chinese Remainder Theorem that $x_2 = 1 + (59 \cdot 54) = 3187 \mod 4189$ is the unique solution to our pair of simultaneous congruences.

C

SEXY SPY BIRDS

They encrypt, they build nests, they do sexy dances.

Male Andean cock-of-the-rock (image from [43])

The Andean cock-of-the-rock (or *Rupicola peruvianus*) is a resident of the cloud forests of the Andes. It is the national bird of Peru, but ranges from Venezuela to Bolivia (it does not cohabit with the similar looking Guianan Cock-of-the-rock). The males gather in leks (communal breeding area, typically on the ground) and display to observing females. The males jump up and down on carefully chosen display branches and emit low, guttural croaks ("look how sexy I am, I'm hopping aggressively"). Despite their bright colors, males are hard to spot away from the leks [44]. Females, already decked out in camouflaging brown, are hard to spot anywhere.

Female Andean cock-of-the-rock (image from [43])

The name "Cock-of-the-rock" comes from the species' preference for building their mud cup nests on rocks and ledges [44].

D

RECIPES

D.0.1 *Crypto Crack (E. McNicholas)*

Ingredients:

1 sleeve of saltine crackers
1 cup butter
1 cup brown sugar
1 bag semi-sweet or dark chocolate chips
crushed nuts, candy canes, or sea salt (optional)

Instructions:

- Preheat oven to 400 degrees
- Line 1 large jelly-roll pan with aluminum foil and then butter or spray with nonstick spray.
- arrange saltines on pan so that the pan is covered and saltines are not overlapping (excellent square packing problem)
- In a medium saucepan, melt butter and brown sugar together, stirring constantly. Bring to boil and continue to whisk while mixture boils another 4 minutes (it should turn an opaque caramel color)
- Pour caramel mixture over crackers, covering them completely and as evenly as possible (you can smooth it out with a spatula after pouring)
- Pop into oven for 5 minutes
- Take out and sprinkle chocolate over top. Let sit for 5 minutes or until chocolate has melted. Carefully spread chocolate over top with spatula.
- Garnish with crushed nuts, peppermint candy, or a light sprinkling of sea salt if desired.
- Pop into freezer of fridge until completely cooled. Break into pieces, and enjoy with your most magnificent mathematical friends.

D.0.2 *Famous Oatmeal Cookies - Gluten free style (J. McConachie)*

Ingredients:
1 cup firmly packed brown sugar
3/4 cup vegetable shortening/margarine
1/2 cup granulated sugar
1 egg
1 tsp vanilla
3 1/2 cups gluten free oats
1 cup all purpose gluten free flour (rice or mix)
1 tsp salt (optional)
1/2 tsp baking soda
1 tsp xantham gum

Preparation:

1. Heat oven to 350 degrees. In a large bowl, beat brown sugar, margarine, and granulated sugar until creamy. Add egg, water, and vanilla. Beat well. Add combined oats, xantham gum, flour, salt, and baking soda. Mix well.
2. Drop dough by rounded teaspoonfuls onto ungreased cookie sheets. (If a few teaspoonfuls disappear between the bowl and the sheet, so be it).
3. Bake 11 to 13 minutes or until edges are golden brown. Remove to wire rack. Cool completely. Store tightly covered.

I'd like to thank Quaker Oats for providing such an easily adaptable recipe.

D.0.3 *Jello and Strawberries (Alicia Nichols)*

Ingredients:
8 Jello Cups
1 Package of Strawberries

Preparation:

1. Go to Safeway
2. Buy the Jello with sugar because sugarless Jello is a crime against humanity
3. Buy the strawberries that look the least rotten
4. Take home 7 uneaten Jello cups after class because your fellow students are Jello Haters

D.0.4 *Chocolate Chip Cookie Dough Balls (Daphne Jacobsen)*

Ingredients:

Cookie Dough Balls:

2/3 cup (1 1/3 sticks) butter, room temperature

3/4 cup granulated sugar

1/4 cup dark brown sugar, packed

1/4 cup applesauce

1 teaspoon vanilla extract

1 3/4 cups all-purpose flour

1/2 teaspoon baking soda

1/2 teaspoon salt

1 cup mini semisweet chocolate chips

Dipping Chocolate:

12 ounces dark chocolate wafers or semisweet chocolate chips

Preparation:

For the cookie dough balls: In an electric mixer, beat the butter, granulated sugar, brown sugar, applesauce and vanilla together until smooth. Sift the flour, baking soda and salt together, and with the mixer running, slowly add to the butter mixture. Mix in the chocolate chips. Scoop out the batter using a mini ice cream scoop and roll it into balls. Place the dough balls about an inch apart on a baking sheet lined with wax paper and transfer them to the freezer for 30 minutes to let the dough set up.

For the dipping chocolate: While the dough is chilling, melt the chocolate wafers and chocolate-almond bark together in a heavy-bottomed saucepan over low heat, stirring constantly. Remove the chilled dough balls from the freezer. Using two spoons, dip the balls one at a time in the melted chocolate, rolling them to coat fully. Returned the dough balls to the wax paper-lined baking sheet and refrigerate until set, about 30 minutes. Store in the fridge for up to 2 weeks.

D.0.5 *Dairy Free, Gluten Free, Coconut Rice Crispy Treats (Dusan Pekich)*

Ingredients:
4 cups rice crispies
3/4 cup vegetable shortening/margarine
1 cup marshmellows
1/2 cup coconut
2 tsp vanilla

Preparation:

1. In a pot on the stove top melt the vegetable shortening and the marsh mellows on low heat. Stir until a sticky but consistent mess.
2. Stir in vanilla and coconut.
3. Stir in rice crispies. Stir until all marshmellows and rice crispies are evenly mixed with no large areas that are too wet or too dry. Turn off the heat after you start stirring to prevent burning on the bottom.
4. Pour into a large greased glass baking pan and let cool. If you are impatient, you can put them in the freezer so they will firm up quicker.
5. Once firm, cut into the portions that you would like and serve!

D.0.6 *Jello and Strawberries (Alicia Nichols)*

Ingredients:
8 Jello Cups
1 Package of Strawberries

Preparation:

1. Go to Safeway
2. Buy the Jello with sugar because sugarless Jello is a crime against humanity
3. Buy the strawberries that look the least rotten
4. Take home 7 uneaten Jello cups after class because your fellow students are Jello Haters

D.0.7 *Rice Krispies Treats (Journey Penney)*

Ingredients:

4 cups butter
4 cups mini marshmallows
4 cups rice cereal

Instructions:

- Heat the butter in a large saucepan on low heat until almost melted.
- Add the marshmallows and heat slowly with little to no stirring, until the marshmallows are almost melded together.
- Stir gently until the marshmallows resemble marshmallow cream. The cream should be light and fluffy. If you heat too quickly, it will become tough.
- When it is creamy and the butter is blended in, turn off the heat and add the rice cereal. Gently blend the marshmallow cream and the cereal until evenly distributed.
- Pour the mixture into a greased pan.
- The treats can be cut while warm or cool.

D.0.8 *Classic Chocolate Chip Cookies (M. Altenhof-Long)*

Ingredients:

1 cup butter

1 cup brown sugar

1/2 cup white sugar

1-3 tsp vanilla, depending on strength

2 eggs

2 1/4 cups flour

1/2 tsp table salt

1 tsp baking soda

2 cups chocolate chips

Instructions:

1. Soften butter in large mixing bowl for several hours at room temperature. If impatient, melt butter in microwave.
2. Mix brown sugar and white sugar into softened butter.
3. Add vanilla and eggs. Mix well.
4. Stir in 2 cups of flour. Add salt and baking soda.
5. Mix in chocolate chips and remaining 1/4 cup of flour.
6. Chill dough in refrigerator for at least 2 hours.
7. Preheat oven to 350 degrees.
8. Form dough into balls (roughly 1/8 cup dough each) and place on greased cookie sheet.
9. Bake on center rack for about 12 minutes or until golden brown around edges.
10. Let cool, and enjoy!

D.0.9 *Nigerian Beans (Leo Goldstein)*

Serves 2, 1 hour active time, 3-4 hours total

Ingredients:

1 cup white beans (navy, great northern, etc)

3 tbsp olive/vegetable oil

1 medium onion, chopped

4 cloves garlic, peeled and diced

1 tbsp curry powder

1 12 oz can of chopped tomatoes (or 2 medium tomatoes, chopped)

2 1/2 tbsp peanut butter

Salt and pepper to taste

Instructions:

1. Soak beans overnight OR bring beans to boil, immediately turn off heat and soak covered for 1 hr
2. Simmer beans in 3 1/2 cups of water for 40-60 minutes, until tender
3. After 20 min, preheat oven to 325 and heat oil in a saucepan or dutch oven. Add onions and cook for 1-2 minutes
4. Add garlic, cook for an additional 1-2 minutes then add curry powder and stir until well-mixed
5. Add tomatoes to the garlic and onions and cook for 8 minutes. Salt and pepper to taste
6. Once beans are cooked, add 6 tbsp of the liquid to peanut butter in a small bowl and mix. Drain beans of remaining liquid
7. Combine beans, tomato onion mixture, and peanut butter mixture in the dutch oven (if using) or a casserole dish and bake uncovered for 2 hrs, serve while hot

D.0.10 *Wasted Chicken – Ian Gilbert*

Serves one, preferably lonely, person.

Ingredients:

- 1 tablespoon vegetable oil
- 1 tablespoon butter
- 1 tablespoon flour
- 1/2 teaspoon sugar
- 2 chicken breasts
- 1/2 cup chicken broth
- 1 pound mushrooms
- bottle of red wine

Instructions:

1. Start to assemble the ingredients. Since you will have some leftover wine, I recommend taking this time to pour yourself a glass and turn on some music.
2. Preheat the oven to 375 degrees, and heat a skillet over medium-high heat; add 1/2 tbsp of vegetable oil. Have you finished your glass of wine yet? If so, this is an opportune moment to pour yourself a second glass.
3. Toss the chicken into the skillet, and cook until one side gets all brown and stuff.
4. Throw the chicken in one of those baking tray thingies. You should really have another glass of wine about now. It's been a long week, and you deserve it.
5. Chuck in the mushrooms and some more vegetable oil.
6. Pick up the friggin mushrooms on the floor. It's fine, like just wash them and put them back. It's fine. Just like, have some more wine to relax. Put the oven in the chicken, and wait for a while. Or something, I don't know.
7. Put the butter in the ban until it melts. Add the flour, then mix in the chicken broth, sugar, and shalt.
8. Ok, look. You finished the wine, but you havend even used it for the cook yet. But like don't even worry about it, cause you got another bottle downshairs. But you should hurry, cause you're like cooking right now.
9. Pour some pine in the pan. Also in your glass. Also, the chicken's probably burned, so you should do somefink about that. It's ok though, lifesh shard. Bwahahaha, lifesh shard! Cause the chicken's charred! You know, you're, like, so funny, why don't other people see that about you? Whatever, those people like aren't even that cool anyways you know its just like they're all like "hey" and...
10. Pour the pan on the shicken, and yer done!. Add one table, to taste, and sherve on a plate or shumfink. Have shum more wine in bed, and go to shleep.

D.0.11 *Hummus (Miles Smith)*

Ingredients:

- 1/2 half cup of roasted tahini
- 1/4 cup of extra virgin olive oil
- 2 garlic cloves (mashed)
- 2 15oz cans of chickpeas
- 1/4 cup of freshly squeezed lemon juice
- 1/2 cup of the purest water you can find
- 1/2 teaspoon of salt

Instructions:

1. In a bowl combine the tahini and olive oil mix until smooth.
2. Add literally everything else on the list, and mix it together
3. Add more salt if you like that sort of thing, or more olive oil.

Remark D.0.1 (Eating Instructions). Often Hummus is best enjoyed as a dip. Therefore it is highly recommend that you buy food like carrots, pretzels, pita bread, or anything you can think of that would help you enjoy your hummus.

E

QUOTES

"I have a biological answer."
– Jaime
"Gross." – Leo

"What if my whole proof is based on a lie." – Jaime
"You're thinking too small. What if your whole life is based on a lie." – Leo

"Growthset mindset" – Daphne

"Hi, I'm Daphne. A fun fact for Leo is whenever I get a run in my stocking I want to destroy it." – Daphne

"What is the generic term for jello? Gelatinous flavored blob?" – Daphne

"My proof strat" – Miles

"I'm biased against prime numbers." – Jaime

"You think this is a game?" – Dan

Squart – (written by) Maggie

"Do not cut my banana in half, Daphne." – Alicia

"You stole my seat, Dan."
– Alicia

[snorts] – Alicia

"Jaime, are you ready to write?"
– Leo
"Are you ready to rock?"
– Alicia

"Dan, thank you for leading us in our journey of discovery."
– Alicia

[In respones to Miles' fun fact that his favorite color is blue] "We're already scraping at the end of the Miles barrel." – Leo

"In a lot of ways, all of our souls have been broken." – Leo

"Daniel stole it. He steals things." – Alicia

"One brain is broken, but one brain is fixed." – Miles

"This one's a nasty boy." – Leo

"If the marker is smaller and the writer is small then the writing has to be small. It's the transitive property of small." – Alicia

"the Alfie cipher" – Ian

"Sometimes I forget my mouth is airtight" – Dan

"This homework is due in like 24 hours!" – Maggie "Yeah and you're wasting 8 of those hours *sleeping*" – Dan

"It's an LGBT lobster." – Leo

"We know English real good."
– Daphne

"This place is like a fortress"
– Alicia
"Its a FORDtress" – Maggie

"Just be sure to specify where those variables live" – Miles
"In a pineapple under the sea?"
– Ian

"Look how sexy I am, I'm hopping aggressively." – Jaime

"You're better than the internet." – Daphne to Maggie

"I usually really like weird, boring, repetitive tasks... but this is ridiculous (in reference to Example 3.5.11)" - Maggie

"As a white, male in 2018, I can say with certainty that this [Example 3.5.11] is worse than any systematic oppression in all times."– Dan

"Mwahahahaha! Hufflepuff my ass!" – Erin

" I saw Miles today without Ian" – Jaime

"What?!?" – Maggie

"Shield your ears Jaime, we are doing math our way"– Maggie

"Jaime grew up in a very traditional household where they took their modulus at the end"– Dan

"You probably have not seen me eat, it happens when you are blinking." – Dan

"Not Today" – Daphne

"When I was young I had my own cipher" – Maggie

"Even though no one heard me, I really like my own quote about..." – Ian

"I didn't want to do anything as a kid" – Maggie

"If g is a penetrator..." –Daphne

" I'm a good addition to any thesis" – Dan

The roots are distinct – (written by) Anonymous

"You made an effort, you just didn't do a good job." – Leo

REFERENCES

[1] J. Hoffstein, J. Pipher, and J. H. Silverman, *An introduction to mathematical cryptography*, Second, ser. Undergraduate Texts in Mathematics. Springer, New York, 2014, pp. xviii+538. DOI: 10.1007/978-1-4939-1711-2. [Online]. Available: https://doi.org/10.1007/978-1-4939-1711-2.

[2] S. Singh, *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography*, 1st. New York, NY, USA: Doubleday, 1999, ISBN: 0385495315.

[3] D. Stinson, *Cryptography: Theory and Practice,Second Edition*, 2nd. CRC/C&H, 2002, ISBN: 1584882069.

[4] L. D. Broemeling, "An account of early statistical inference in arab cryptology.," *The American Statistician*, vol. 65, no. 4, p. 255, 2011, ISSN: 15372731. [Online]. Available: https://ezproxy.app.willamette.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=msn&AN=MR2867510.

[5] R. E. Lewand, *Cryptological mathematics*. Mathematical Association of America, Washington, DC, 2000, ISBN: 0-88385-719-7. [Online]. Available: https://ezproxy.app.willamette.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=msn&AN=MR1818075.

[6] L. S. Hill, "Cryptography in an algebraic alphabet.," *American Mathematical Monthly*, vol. 36, no. 6, p. 306, 1929, ISSN: 19300972. [Online]. Available: https://ezproxy.app.willamette.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=msn&AN=MR1521759.

[7] H. Anton and R. Busby, *Contemporary Linear Algebra, Textbook and Student Solutions Manual*. John Wiley & Sons Incorporated, 2002, ISBN: 9780471450016. [Online]. Available: https://books.google.com/books?id=5j3BAQAACAAJ.

[8] J. Levine, "Some elementary cryptanalysis of algebraic cryptography.," *The American Mathematical Monthly*, vol. 68, p. 411, 1961, ISSN: 00029890. [Online]. Available: https://ezproxy.app.willamette.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=msn&AN=MR0129074.

[9] S. Khazaei and S. Ahmadi, "Ciphertext-only attack on $d \times d$ hill in $O(d13^d)$.," *Information Processing Letters*, vol. 118, p. 25, 2017, ISSN: 00200190. [Online]. Available: https://ezproxy.app.willamette.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=msn&AN=MR3574187.

[10] L. S. Hill, "Concerning certain linear transformation apparatus of cryptography.," *American Mathematical Monthly*, vol. 38, no. 3, p. 135, 1931, ISSN: 19300972. [Online]. Available: https://ezproxy.app.willamette.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=msn&AN=MR1522201.

[11] M. Agrawal, N. Kayal, and N. Saxena, "PRIMES is in P," *Ann. of Math. (2)*, vol. 160, no. 2, pp. 781–793, 2004, ISSN: 0003-486X. DOI: 10.4007/annals.2004.160.781. [Online]. Available: https://doi.org/10.4007/annals.2004.160.781.

[12] Aftab, Cheung, Kim, Thakkar, and Yeddanapudi, *Information theory: Information theory and the digital age*. [Online]. Available: http://web.mit.edu/6.933/www/Fall2001/Shannon2.pdf.

[13] *Perfectly-secret ciphers and shannon's theory*, Dec. 2013. [Online]. Available: http://cryptowiki.net/index.php?title=Perfectly-secret_ciphers_and_Shannon's_theory.

[14] U. of Pennsylvania, *Probability, conditional probability & bayes rule*. [Online]. Available: http://www.seas.upenn.edu/~cis391/Lectures/probability-bayes-2015.pdf.

[15] H. C. A. v. Tilborg, *An Introduction to Cryptology*. Kluwer, 1998.

[16] J. B. Anderson and R. Johnnesson, *Understanding Information Transmission (IEEE Press Understanding Science & Technology Series)*. Wiley-IEEE Press, 2005.

[17] D. Welsh, *Codes and Cryptography*. New York, NY, USA: Clarendon Press, 1988, ISBN: 0-19-853287-3.

[18] D. R. Stinson, *Cryptography: Theory and Practice*, 3rd. Chapman & Hall, 2006.

[19] *Venona project*, Apr. 2018. [Online]. Available: https://en.wikipedia.org/wiki/Venona_project.

[20] *Random number generation*, Apr. 2018. [Online]. Available: https://en.wikipedia.org/wiki/Random_number_generation.

[21] *One-time pad*, Apr. 2018. [Online]. Available: https://en.wikipedia.org/wiki/One-time_pad.

[22] S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. System Sci.*, vol. 28, no. 2, pp. 270–299, 1984, ISSN: 0022-0000. DOI: 10.1016/0022-0000(84)90070-9. [Online]. Available: https://doi.org/10.1016/0022-0000(84)90070-9.

[23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptology—EUROCRYPT '99 (Prague)*, ser. Lecture Notes in Comput. Sci. Vol. 1592, Springer, Berlin, 1999, pp. 223–238.

[24] I. Damgård, M. Jurik, and J. Nielsen, "A generalization of paillier's public-key system with applications to electronic voting.," *International Journal of Information Security*, vol. 9, no. 6, pp. 371–385, 2010, ISSN: 16155262.

[25] S. W. Gebregiyorgis, "Algorithms for the elliptic curve discrete logarithm and the approximate common divisor problem," PhD thesis, 2016.

[26] P. Novotney, "Weak curves in elliptic curve cryptography," *Modular Math Washington*, 2010.

[27] N. Koblitz, A. Menezes, and S. Vanstone, "The state of elliptic curve cryptography," *Designs, Codes, Cryptography*, vol. 19, pp. 173–193, 2000.

[28] F. Leprévost, J. Monnerat, S. Varrette, and S. Vaudenay, "Generating anomalous elliptic curves," *Information processing letters*, vol. 93, no. 5, pp. 225–230, 2005.

[29] D. Husemoeller, *Elliptic Curves*, second, ser. Graduate Texts in Mathematics. Springer, 2000, ch. 3.

[30] L. Bauer, "Weierstrass equations: Seminar on elliptic curves and the weil conjectures," PhD thesis, University of Regensburg, 2016.

[31] N. Carella, "Topic in elliptic curves over finite fields: The groups of points," *arXiv preprint arXiv:1103.4560*, 2011.

[32] J. H. Silverman, *The arithmetic of elliptic curves*, Second, ser. Graduate Texts in Mathematics. Springer, Dordrecht, 2009, vol. 106, pp. xx+513, ISBN: 978-0-387-09493-9. DOI: 10.1007/978-0-387-09494-6. [Online]. Available: https://doi.org/10.1007/978-0-387-09494-6.

[33] P. Griffiths and J. Harris, *Principles of Algebraic Geometry*, ser. Wiley Classics Library. Wiley, 2011, ISBN: 9781118030776. [Online]. Available: https://books.google.com/books?id=Sny48qKdW40C.

[34] A. E. Aftuck. (). The weil pairing on elliptic curves and its cryptographic applications, [Online]. Available: https://digitalcommons.unf.edu/cgi/viewcontent.cgi?article=1138&context=etd.

[35] S. Wang. (). Efficient computation of miller's algorithm in pairing-based cryptography, [Online]. Available: https://scholar.uwindsor.ca/cgi/viewcontent.cgi?article=7026&context=etd.

[36] B.-Y. Y. Jintai Ding, *Post-quantum cryptography*, 1st. Springer International Publishing, 2006.

[37] D. S. Jintai Ding Jason E. Gower, *Multivariate Public Key Cryptosystems*, 1st. Springer International Publishing, 2006.

[38] H. I. Tsutomu Matsumoto, "Public quadratic polynomial-tuples for efficeint signature-verication and message encryption," *Advances in Cryptology*, 1988.

[39] J. A. Gallian, *Contemporary Abstract Algebra*, 8th. Brooks and Cole Cengage Learning, 2006.

[40] J. Patarin, "Cryptanalysis of the matsumtoto and imai public key scheme of eurocrypt'88," *Advances in Cryptology*, 1995.

[41] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, Dec. 1982, ISSN: 1432-1807. DOI: 10.1007/BF01457454. [Online]. Available: https://doi.org/10.1007/BF01457454.

[42] M. Tischer, Z. Durumeric, S. Foster, S. Duan, A. Mori, E. Bursztein, and M. Bailey, "Users really do plug in usb drives they find," in *Security and Privacy (SP), 2016 IEEE Symposium on*, IEEE, 2016, pp. 306–319.

[43] *Andean cock-of-the-rock*, May 2018. [Online]. Available: https://en.wikipedia.org/wiki/Andean_cock-of-the-rock.

[44] *Andean cock-of-the-rock (Rupicola peruvianus)*, May 2018. [Online]. Available: https://neotropical.birds.cornell.edu/Species-Account/nb/species/andcot1/overview.

[45] A. Fowler. (). Kangaroo methods for solving the interval discrete logarithm problem, [Online]. Available: https://arxiv.org/pdf/1501.07019.pdf.

[46] C. Studholme. (). The discrete logarithm problem, [Online]. Available: http://www.cs.toronto.edu/~cvs/dlog/research_paper.pdf.

[47] C. Ritzenthaler. (). Discrete logarithms, [Online]. Available: https://perso.univ-rennes1.fr/christophe.ritzenthaler/cours/DLP.pdf.

[48] J. M. Pollard, "Monte carlo methods for index computation (mod p)," *Mathematics of Computation*, vol. 32, no. 143, pp. 918–924, 1978. DOI: http://pages.cs.wisc.edu/~cs812-1/kangaroo.pdf.

[49] J. S. Coron, D. Lefranc, and G. Poupard. (). A new baby-step giant-step algorithm and some applications to cryptanalysis, [Online]. Available: https://iacr.org/archive/ches2005/004.pdf.

[50] S. Galbraith, *Mathematics of Public Key Cryptography*. Reading, Massachusetts: Cambridge University Press, 2012, ch. Chapter 14 Factoring and Discrete Logarithms using Pseudorandom Walks.

[51] G. Fuchsbauer, "An introduction to probabilistic encryption," *Osječki Matematički List*, vol. 6, no. 1, pp. 37–44, 2006.

[52] L. Hoffmann, "Cracking the code.," *Communications of the ACM*, vol. 56, no. 6, pp. 120–122, 2013, ISSN: 00010782.

[53] F. Zhang, S. Liu, and K. Kim. (2002). Id-based one round authenticated tripartite key agreement protocol with pairings.

[54] M. O. Courseware. (). Elliptic curve cryptography, [Online]. Available: https://ocw.mit.edu/courses/mathematics/18-704-seminar-in-algebra-and-number-theory-rational-points-on-elliptic-curves-fall-2004/projects/asarina.pdf.

[55] A. Sinkov, *Elementary cryptanalysis.* Ser. Anneli Lax New Mathematical Library, 22. Mathematical Association of America, Washington, DC, 2009, ISBN: 978-0-88385-647-5. [Online]. Available: https://ezproxy.app.willamette.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=msn&AN=MR2530836.

[56] J. Levine, "Variable matrix substitution in algebraic cryptography.," *The American Mathematical Monthly*, vol. 65, p. 170, 1958, ISSN: 00029890. [Online]. Available: https://ezproxy.app.willamette.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=msn&AN=MR0106793.

[57] A. M. Odlyzko, "The rise and fall of knapsack cryptosystems," *Cryptology and computational number theory*, vol. 42, pp. 75–88, 1990. [Online]. Available: http://www.dtc.umn.edu/~odlyzko/doc/arch/knapsack.survey.pdf.

[58] J. Bakker. (2004). The knapsack problem and the LLL algorithm, [Online]. Available: http://www.math.ucsd.edu/~crypto/Projects/JenniferBakker/Math187/#Anchor-Th-1435.

[59] D. P. Chi, J. W. Choi, J. San Kim, and T. Kim, "Lattice based cryptography for beginners.," *IACR Cryptology ePrint Archive*, vol. 2015, p. 938, 2015. [Online]. Available: https://eprint.iacr.org/2015/938.pdf.

[60] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *International Workshop on Public Key Cryptography*, Springer, 2010, pp. 420–443. [Online]. Available: https://eprint.iacr.org/2009/571.pdf.

[61] Z. Brakerski and V. Vaikuntanathan, "Lattice-based FHE as secure as PKE," in *Proceedings of the 5th conference on Innovations in theoretical computer science*, ACM, 2014, pp. 1–12. [Online]. Available: https://eprint.iacr.org/2013/541.pdf.

[62] J. Alin and C. Starr, *Undergraduate matrix theory and linear algebra*, 2018. [Online]. Available: http://willamette.edu/~cstarr/math253/0LinAlgMain-18.pdf.

[63] R. Merkle and M. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE transactions on Information Theory*, vol. 24, no. 5, pp. 525–530, 1978. [Online]. Available: http://www.academia.edu/13845771/Hiding_information_and_signatures_in_trapdoor_knapsacks.