2012

An introduction to modern cryptology within an algebraic framework

John Szwast
Eastern Washington University

AN INTRODUCTION TO MODERN CRYPTOLOGY WITHIN AN ALGEBRAIC FRAMEWORK

A Thesis

Presented To

Eastern Washington University

Cheney, Washington

In Partial Fulfillment of the Requirements

for the Degree

Master of Science

By

John Szwast

Fall 2012

THESIS OF JOHN SZWAST APPROVED BY

_____ DATE:_____

DR. RON GENTLE, GRADUATE STUDY COMMITTEE

_____ DATE:_____

DR. DALE GARRAWAY, GRADUATE STUDY COMMITTEE

_____ DATE:_____

DR. PAUL SCHIMPF, GRADUATE STUDY COMMITTEE

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Scope

This work is an introduction to modern cryptology built within an algebraic framework, rather than a number theoretic one. Working within algebra, historic shift cyphers (such as the Caesar cypher) shall be studied and expanded. Recent exponential cyphers including public-key cyphers will be studied. Finally, a survey of methods of solving the arithmetic problems that form the basis of public-key cryptosystems will be offered.

In addition to the cyphers, methods of breaking them will be studied along side the cyphers and statistical analyses of the different attacks will be provided.

1.2 Background

The material presented herein is done so with the expectation of a solid grounding in basic finite algebra. A basic understanding of finite group theory, finite ring theory, and finite fields is expected; A more detailed understanding will be helpful. An undergraduate level understanding of number theory is also expected of the reader.

The rings of integers mod l will be very frequently used, as will their groups of units. This includes the fields of integers mod p, for prime p. Since binary data is under study, finite fields with a characteristic of 2 will also be seen.

1.3 Conventions and Notations

For the purposes of this work, zero shall be considered a natural number ($0 \in \mathbb{N}$). When necessary, the set of natural numbers excluding zero shall be denoted as $\mathbb{N}^*$.

The base-10 logarithm, $\log_{10}$, shall be denoted as log. The natural logarithm, $\log_e$, shall be denoted as ln. The base-2 logarithm, $\log_2$, shall be denoted as lg.

Numbers may be represented in bases other than 10 at various times. Bases of 2 and 10 are standard and when the context provides clarity numbers written in those bases will be done so with no special notation, otherwise base-2 numbers shall be followed with a subscript of 2: $101 = 1100101_2$. When a number is written in a non-standard base, each digit will be written in base 10 unless otherwise noted, with a colon ':' separating each digit. For example, $1383 = 7 \cdot 14^2 + 0 \cdot 14 + 11$, it would be expressed in base 14 as 7:0:11. This notation mimics the common notation for time in the United States, which uses a base of 60 for the minutes and

seconds digits.

The ring of integers mod n shall be represented as $\mathbb{Z}_n$. Finite fields shall be denoted by $\mathbb{F}_q$ or $\mathbb{Z}_p$ for a prime p. The multiplicative group of units of a ring shall be denoted with the postfix, unary, * operator: R^*.

The R-module of n-dimensional vectors over R shall be denoted by R^n. The R-module of $m \times n$ matrices over R shall be denoted by $R^{m \times n}$.

The ring of $n \times n$ matrices over a ring, R, shall be denoted as $M_n(R)$. The multiplicative group of invertible $n \times n$ matrices over a ring may be denoted with the * operator or by $GL_n(R)$.

1.4 Overview

Data encryption through cyphers is an ancient form of security that has been relied on to protect information for thousands of years. Julius Caesar used them to communicate securely with his officers. One of the greatest victories of the Allied forces over Nazi Germany in World War II is the breaking of The Enigma, Germany's secret military code.[1] Today, many governments, corporations, and individuals rely on data encryption to secure data.

Initially, cyphers worked by operating on letters of the alphabet. Each letter in an original message was replaced with another letter of the alphabet. Decyphering the secret message merely required knowing the original substitutions and applying the inverse substitution. The Caesar Cypher was one of these where every letter was replaced by the one three positions later

[1] For a short description of Germany's Enigma machine, which used permutation cyphers, see Hardy, pp 81-86. [2]

in the alphabet. Decyphering messages with the Caesar Cypher worked by substituting each letter in the secret message with the one three positions earlier in the alphabet.

It quickly becomes advantageous to associate each letter or other character with a unique numerical value so encyphering and decyphering can be described with analytic functions. With this, cyphers realized by addition, like the Caesar Cypher, can be expanded to cyphers using multiplication on the initial characters, like the linear cyphers. Exponentiation can also be added to the list of possible operations to perform on the characters of the message to be encyphered. Since these operations are valid in any ring, the characters of the messages need not necessarily be associated with numbers, *per se*, but may be associated with unique elements of any ring of an appropriate size.

Each style of encryption will have aspects that can be analyzed: speed, or computational complexity, of encryption and decryption, as well as ease, or not, of a third party breaking the cypher.

1.5 General Definitions

Encyphering is always performed by the application of an invertible transformation $f : \mathcal{P} \to \mathcal{C}$. Decyphering, therefore, is achieved by applying the inverse transformation f^{-1}. The original, legible message is referred to as **plaintext**, P, and the result is referred to as **cyphertext**, C.

Definition 1.1 (plaintext, P)**.** The standard, userland data. This may be a text message (i.e. email) or a binary or text computer file (JPEG, ZIP, INI, etc...) or object.

Definition 1.2 ($\mathcal{P}$)**.** The set of all possible plaintexts for a given transformation f, its domain.

Definition 1.3 (cyphertext, C)**.** The result of applying a transformation f to a plaintext $P \in \mathcal{P}$, $f(P)$.

Definition 1.4 ($\mathcal{C}$)**.** The set of all possible cyphertexts, $f(\mathcal{P})$.

Definition 1.5 (encypher)**.** The act of applying a transformation f to a plaintext $P \in \mathcal{P}$ resulting in a cyphertext $C \in \mathcal{C}$.

Definition 1.6 (decypher)**.** The act of applying the inverse f^{-1} of an encyphering transformation to cyphertext resulting in plaintext.

In practice, encyphering a plaintext is expected to conceal its contents, but these definitions do not exclude the identity function $f(x) = x$ as the transformation. Different **encryption schemes** will provide different degrees of obfuscation of the original plaintext.

Definition 1.7 (encryption scheme)**.** A family of related invertible transformations, each differing only in the value of the parameters used.

Definition 1.8 (key)**.** The values used for the parameters of an encryption scheme in a particular instance.

1.6 Bit Operations and Big-O Notation

1.6.1 Overview

Computers work in binary. Each binary digit is called a bit. The time a computer would spend running a certain algorithm is measured in bit operations, and this time is usually expressed as

a function of the input, or of the size of the input. Since all (classical) computers are built on the same engineering fundamentals, the actual number of bit operations required on a specific processor will not vary significantly from properly calculated theoretical values. What can vary significantly from one computer to another, is the number of bit operations it can perform in a given amount of real time. This significant variance in the powers of computers is the primary motivation for using bit operations as the unit of measurement for the time of an algorithm, rather than a unit of clock time like seconds.

Rather than being concerned with the exact number of bit operations required for a specific input, it is usually more important to know how the time required changes with regard to the input When the size of an input to a procedure doubles, it is most desirable to know whether the time required for the procedure will double with it, or increase 4-fold, or square, or increase by only a small percentage or not change at all. This information is the focus of the **Big-O notation**.

Definition 1.9 (Big-O notation)**.** If $f, g : \mathbb{N}^* \to \mathbb{R}^>$, it is said that $f = \mathcal{O}(g)$, "f is on the order of g," if there exists a non-zero constant c such that $f(n) \leq cg(n)$ for all n.

Example 1.10. The following are all true.

1. $3n + \log n = \mathcal{O}(n)$

$$3n + \log n = \left(3 + \frac{\log n}{n}\right) n \leq 4n$$

2. $2n^2 + 4n = \mathcal{O}(n^2)$

$$2n^2 + 4n = \left(2 + \frac{4}{n}\right) n^2 \leq 6n^2$$

Big-O notation is small and simple, and shows nicely the time complexity of a process as a function of the size of its input.

The following simplification and extension will make Big-O notation even easier and more helpful. First, there may be multiple inputs to a given procedure, so that will be taken into account. Second, on very small inputs smaller order aspects of a procedure may dominate the time requirement, but since that time requirement will be so small it will not be of any concern. The time required to complete large tasks is of primary concern. For example, the time required for a computer to add two 2-digit numbers might be dedicated primarily to system overhead, such as memory access. It may take essentially the same amount of time to add two 2-digit numbers as two 15-digit numbers. But perhaps the overhead involved starts requiring proportionally shorter time as the size of addends increases above 15 digits so that it is negligible for the addition of two 200-digit numbers. It won't be important about how the time required increased going from 1-digit to 15-digit addition; what is important is how the time required increases going from 100-digit addition to 1000-digit addition.

While in practice any addition takes a minuscule amount of real time, algorithms other than addition will be analyzed, for which only the time required for larger situations will be important. The following redefinition of Big-O notation will make it more extensible and more forgiving of quirky behavior on small problems.

Definition 1.11 (Big-O notation)**.** If $f, g : \mathbb{N}^{*m} \to \mathbb{R}^{>}$, it is said that $f = \mathcal{O}(g)$, "f is on the order of g," if there exists a non-zero constant c and a vector $b \in \mathbb{N}^{*m}$ such that $f(n) \leq cg(n)$ for all $n \in \mathbb{N}^{*m}$ where $n_i \geq b_i$ for all $i \leq m$.

In this way, Big-O notation will describe the time behavior of an algorithm when the inputs

are "big enough."

1.6.2 Time Estimates for Integer Arithmetic

Addition and Subtraction

When counting the time required to add two integers, m and n, it can be assumed without loss of generality than $m \geq n$. Integer addition is performed digit-by-digit starting with the least significant digit, the ones digit. In each digit operation, the two corresponding digits are summed along with any carry digit from the previous digit. The least significant digit of the result is the corresponding digit of the sum, and the other digits are carried to the next digit's summation.

The number of digit operations is equal to the larger of the number of digits of m and the number of digits of n. Since it was presumed that $m \geq n$, then m will have the greater (or same) number of digits, $\log m + 1$. Since $\log m + 1 = \mathcal{O}(\log m)$, then it can be said that the time required to add a smaller integer to m is $\mathcal{O}(\log m)$. It is also said that addition takes logarithmic time with respect to the summands, or linear time with respect to the number of digits, or length, of the summands.

Computers work in binary (base 2), so it could be said that a computer takes $\mathcal{O}(\lg m)$ bit operations. However, $\mathcal{O}(\log m)$ and $\mathcal{O}(\lg m)$ are equivalent since the two functions only differ by a constant multiple, as specified by the logarithm change of base formula, so either one is correct no matter who or what is doing the addition. Subtraction, which is performed internally by computers in a very similar fashion to addition, similarly requires $\mathcal{O}(\lg m)$ time. Specifically, subtraction is performed by first converting the subtrahend, n, into its opposite in $\mathcal{O}(\lg n)$ time,

then adding $-n$ to m in $\mathcal{O}(\lg m)$ time.

Multiplication and Division

Multiplication is performed by computers in a manner that could be called **repeated doubling**. A running total is initialized to a value of zero; At the end this running total will hold the result of mn. For each bit (binary digit) of n, if 2^i is the place value of that bit, and that bit of n is a one, then the running total is incremented by $2^i m$.

Example 1.12. The method of multiplying two integers shall be illustrated with $n = 11 = 1011_2$ and $m = 26 = 11010_2$

i	n	$2^i m$	Running total
	1011	0	0
0	101**1**	11010	11010
1	10**1**1	110100	1001110
2	1**0**11	1101000	1001110
3	**1**011	11010000	100011110

$100011110_2 = 256 + 16 + 8 + 4 + 2 = 286$, which is 11×26.

Theorem 1.13. *Multiplication of two positive integers, m and n, requires $\mathcal{O}(\lg m \lg n)$ time.*

Proof. It may be assumed without loss of generality that $m \geq n$. For each bit of n, of which there are $\mathcal{O}(\lg n)$, an $\mathcal{O}(\lg(mn))$-digit addition may be performed. So the total time required is

$$\mathcal{O}(\lg n)\mathcal{O}(\lg(mn)) = \mathcal{O}(\lg n)\mathcal{O}(\lg m + \lg n) = \mathcal{O}(\lg n)\mathcal{O}(\lg m) = \mathcal{O}(\lg m \lg n). \qquad \square$$

In this sense, multiplication can be said to take **quadratic time** with respect to the lenth of the factors. More generally, an algorithm that runs in a time as a power of the logarithm of the size of the inputs, $\mathcal{O}((\lg n)^\alpha)$, is said to run in **polynomial time** and one that runs in a time as a power of the size of the inputs, $\mathcal{O}(n^\alpha)$, is said to run in **exponential time**.

Integer division is performed in a parallel, though opposite, fashion to multiplication, utilizing repeated doubling and halving to speed up the process of repeated subtraction. Two running values are maintained: one will end up as the quotient and the second will be the remainder. Thus if both the quotient and remainder are desired, only one operation is required.

Suppose the results of $m \div n$ (integer quotient) and/or $m \mod n$ (remainder) are desired for some given pair of integers m and n. First, the running values are initialized: the one to end as $m \div n$, q, to 0 and the one to end as $m \mod n$, r, as m. Second, the largest integer i such that $2^i n \leq m$ is identified. If $i < 0$ then nothing further need be done; $m = qn + r$ with $r < n$ already. Assuming $i \geq 0$, 2^i is added to the quotient running total, q, while $2^i n$ is subtracted from the remainder running total, r, then the next largest i is found such that $2^i n$ is less than or equal to the remainder running total. This process continues until $i < 0$, at which point the running values will contain the proper quotient and remainder. In practice, the first i is found by doubling n until $n > m$ then dropping back one step. Each following i is then found by halving n until n is less than or equal to the remainder counter.

Example 1.14. The method of dividing two integers shall be illustrated with $n = 11 = 1011_2$ and $m = 300 = 100101100_2$.

i	n	Remainder	Quotient
0	1011	100101100	0
1	10110	100101100	0
2	101100	100101100	0
3	1011000	100101100	0
4	10110000	100101100	0
5	101100000	100101100	0
4	10110000	1111100	10000
3	1011000	100100	11000
2	101100	100100	11000
1	10110	1110	11010
0	1011	11	11011

$11_2 = 3$, and $11011_2 = 27$, and $300 = 27 * 11 + 3$.

Theorem 1.15. *$m \div n$ and $m \mod n$ require $\mathcal{O}(\lg m \lg n)$ time (assuming $m \geq n > 0$).*

Proof. First, the initial i is found in $\mathcal{O}(\lg m - \lg n) = \mathcal{O}(\lg m)$ steps. Second, for each of $\mathcal{O}(\lg m)$ iterations, a subtraction with $\mathcal{O}(\lg n)$ non-trivial bits is performed. The total time consumed is $\mathcal{O}(\lg m) + \mathcal{O}(\lg m)\mathcal{O}(\lg n) = \mathcal{O}(\lg m \lg n)$. □

Exponentiation

Exponentiation is performed in a manner parallel to the repeated doubling of multiplication, by a procedure referred to as **repeated squaring**. To evaluate a^m, a running product is initialized

to 1, then for each bit of m, if 2^i is the place value of that bit, and that bit of m is a one, then the running product is multiplied by a^{2^i}.

Example 1.16. The repeated squaring method of exponentiation shall be illustrated with $a = 15$ and $m = 13 = 1101_2$.

i	m	a^{2^i}	Running product
	1101		1
0	110**1**	15	15
1	11**0**1	225	15
2	1**1**01	50625	759375
3	**1**101	2562890625	1946195068359375

Theorem 1.17. *Exponentiation by repeated squaring takes $\mathcal{O}(m^2(\lg m)(\lg a)^2)$ time.*

Proof. The final result will have $\mathcal{O}((\lg a^m) - 1) = \mathcal{O}(m \lg a)$ binary digits. For simplicity, all intermediary results will be said to have $\mathcal{O}(m \lg a)$ bits (which is still true).

$\mathcal{O}(\lg m)$ iterations will be performed. In each iteration, a squaring (multiplication) and maybe another multiplication will be performed, each in $\mathcal{O}((m \lg a)^2)$ time. So the total time required will be $\mathcal{O}(\lg m)\mathcal{O}((m \lg a)^2) = \mathcal{O}(m^2(\lg m)(\lg a)^2)$. □

1.6.3 Time estimates for modular arithmetic

For any arithmetic done in a $\mathbb{Z}_l$ ring, it will be assumed that most operands are almost as big as l, after all, less than 1% of the elements of $\mathbb{Z}_l$ will have fewer than $\log l - 2$ digits. Therefore each addition and subtraction will be said to occur in $\mathcal{O}(\lg l)$ time and each multiplication, division

and mod will be said to occur in $\mathcal{O}((\lg l)^2)$ time. However, these time estimates do not consider the reduction of the result $\pmod{l}$.

Addition, Subtraction and Multiplication

Theorem 1.18. *In the ring $\mathbb{Z}_l$, addition and subtraction each take $\mathcal{O}(\lg l)$ time and multiplication takes $\mathcal{O}((\lg l)^2)$ time.*

Proof. The time estimates for addition and subtraction in $\mathbb{Z}_l$ do not change because of the special nature of the modular reduction that is able to be employed.

After adding two numbers each less than l, the result must be less than $2l$. So after the $\mathcal{O}(\lg l)$ addition, if the sum is greater than l, subtract l; no division is required, just two $\mathcal{O}(\lg l)$ operations. So addition requires $2\mathcal{O}(\lg l) = \mathcal{O}(\lg l)$ time.

Similarly, the result of a subtraction must be in the range $[-l+1, l-1]$. If it is less than 0, add l. Subtraction also takes $\mathcal{O}(\lg l)$ time.

Multiplication is similar in that after an $\mathcal{O}((\lg l)^2)$ multiplication operation, an $\mathcal{O}((\lg l)^2)$ mod operation follows and $2\mathcal{O}((\lg l)^2) = \mathcal{O}((\lg l)^2)$. □

Exponentiation

Theorem 1.19. *In the ring $\mathbb{Z}_l$, exponentiation, the evaluation of a^m, requires $\mathcal{O}((\lg m)(\lg l)^2)$ time.*

Proof. All results will have $\mathcal{O}(\lg l)$ bits. There will be $\mathcal{O}(\lg m)$ iterations each with up to two modular multiplications in $\mathcal{O}((\lg l)^2)$ time. Therefore the exponentiation will require $\mathcal{O}((\lg m)(\lg l)^2)$ time. □

Division

Division is not defined in $\mathbb{Z}_l$ but multiplication by an inverse (if it exists) can be performed after it is found. If $\phi(l)$ is known, then $a^{-1} = a^{\phi(l)-1}$. $\phi(l) < l$ and thus $\phi(l) = \mathcal{O}(l)$ as a quantity, however knowing $\phi(l)$ is equivalent to knowing the prime factorization of a number (very hard for very large l), and thus it may be practically unknowable.

Finding the inverse of a number, a, in $\mathbb{Z}_l$ when $\phi(l)$ is not known can be done with the Extended Euclidean Algorithm.[2] After finding the greatest common divisor of a and l, which will be 1 if an inverse exists, by the Euclidean Algorithm, run back up the sequence of steps from the Euclidean Algorithm to express 1 as a linear combination of a and l, $1 = ra + sl$. Then $ar = (-s)l + 1$ or $ar \equiv 1 \pmod{l}$ and $r = a^{-1}$ in $\mathbb{Z}_l$.

Theorem 1.20. *The (Extended) Euclidean Algorithm takes $\mathcal{O}((\lg l)^3)$ time.*[3]

Proof. This hinges on the speed of the decrease in the successive remainders from step to step. Specifically, that if r_j is the remainder from the j^{th} step, then $r_{j+2} < \frac{1}{2} r_j$.

If $r_{j+1} \leq \frac{1}{2} r_j$, then $r_{j+2} < r_{j+1} \leq \frac{1}{2} r_j$. If $r_{j+1} > \frac{1}{2} r_j$, then $r_j = 1 \cdot r_{j+1} + r_{j+2}$ and $r_{j+2} = r_j - r_{j+1} < \frac{1}{2} r_j$ because $r_{j+1} > \frac{1}{2} r_j$.

Since every two steps, the remainders are reduced by at least half, there will be no more than $2 \lg l$ steps. Each step down requires one $\mathcal{O}((\lg l)^2)$ integer division, so the Euclidean Algorithm requires $\mathcal{O}((\lg l)^3)$ time. Running back up the steps in the extension requires two $\mathcal{O}((\lg l)^2)$

[2] For a complete description of the (Extended) Euclidean Algorithm see any elementary text on Number Theory, including [7].

[3] The proof of this is essentially copied from Koblitz, 13. [4]

multiplications at each of the $\mathcal{O}(\lg l)$ steps. The extension also occurs in $\mathcal{O}((\lg l)^3)$ time, so the entire Extended Euclidean Algorithm, and thus finding a^{-1} in $\mathbb{Z}_l$ requires $\mathcal{O}((\lg l)^3)$ time. □

Theorem 1.21. *Multiplication in $\mathbb{Z}_l$ by a^{-1} (assuming it exists) requires $\mathcal{O}((\lg l)^3)$ time (regardless of the knowledge of $\phi(l)$.*

Proof. The procedure entails first finding a^{-1}, then multiplying by it in $\mathcal{O}((\lg l)^2)$ time.

If $\phi(l)$ is not known, finding a^{-1} by the Extended Euclidean Algorithm will take $\mathcal{O}((\lg l)^3)$ time. If $\phi(l)$ is known, then finding $a^{-1} = a^{\phi(l)-1}$ will take $\mathcal{O}(\lg(\phi(l)-1)(\lg l)^2) = \mathcal{O}((\lg l)(\lg l)^2) = \mathcal{O}((\lg l)^3)$ time

$$\mathcal{O}((\lg l)^3) + \mathcal{O}((\lg l)^2) = \mathcal{O}((\lg l)^3)$$ □

1.7 Probability

1.7.1 Discrete distributions

Probability distributions are usually classified into two sets: discrete and continuous. Continuous distributions describe situations where a continuum of outcomes are possible, such as an interval of the Real numbers. Discrete distributions describe situations where the set of outcomes is discrete, including all finite distributions.

Let S be the discrete set of all possible outcomes of an experiment. Let X represent the outcome of a specific running of the experiment. X is called a **random variable** or a **discrete random variable**. S is called the **sample space** of X. For each $s \in S$, X has a probability of being s. Designate that probability $p(s)$. $p : S \to \mathbb{R}$ is called a **probability distribution function**, or **pdf**.

Definition 1.22 (random variable)**.** A variable that will randomly assume values from a given set.

Definition 1.23 (sample space)**.** The set of all possible values of a random variable.

Definition 1.24 (probability distribution function)**.** A function $p : S \to \mathbb{R}$ from the sample space of a random variable to the Real numbers such that for each $s \in S$, $p(s)$ is the probability that $X = s$. Probability distribution functions have the following properties:

- $0 \leq p(s) \leq 1$ for each $s \in S$.
- $\sum_{s \in S} p(s) = 1$.
- For any subset $A \subseteq S$, $p(A) = \sum_{s \in A} p(s)$.
- $p(\emptyset) = 0$.
- For $A, B \subseteq S$, $p(A \cup B) = p(A) + p(B) - p(A \cap B)$.

Verifying that a given $p(x)$ is a valid probability distribution function requires $p(s) \geq 0$ for each $s \in S$ and $\sum_{s \in S} p(s) = 1$.

For many discrete distributions, and all the ones used herein, the sample space S will be a (not necessarily finite) set of consecutive integers. Whenever the sample space is a set of numbers certain measurements of the probability distribution may be made. In fact, being numbers is stricter than is necessary. In general, these measurements may be made whenever the sample space S is a module of a ring containing the image of p.

First will be the average, or arithmetic mean, value of X. This is referred to as the **expected value** of X. It is a weighted average of the sample space of X, weighted by each element's probability, commonly referred to as the **mean**.

Definition 1.25 (expected value)**.** The weighted arithmetic mean of all the possible values of X, denoted $\mathrm{E}[X]$.

$$\mathrm{E}[X] = \sum_{x \in S} xp(x)$$

Example 1.26. Find the expected value of X for the pdf given in Table 1.1.

Table 1.1: Example probability distribution function.

x	$p(x)$
0	0.5
1	0.25
2	0.125
3	0.125

First, verify that p is a valid pdf. By inspection, no $p(x)$ is less than 0, and a quick addition verifies that $\sum_{x=0}^{3} p(x) = 1$.

Finally, calculate the expected value of X using the definition of expected value.

$$\mathrm{E}[X] = \sum_{x=0}^{3} xp(x) = 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4} + 2 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} = 0 + \frac{1}{4} + \frac{2}{8} + \frac{3}{8} = \frac{7}{8}$$

The expected value of X is $\frac{7}{8} = 0.875$.

Theorem 1.27. *$E[aX + b] = aE[X] + b$.*

Proof. This follows straight from properties of summation.

$$\begin{aligned}
\mathrm{E}[aX+b] &= \sum_{x\in S}(ax+b)p(x) \\
&= \sum_{x\in S}(axp(x)+bp(x)) \\
&= \sum_{x\in S} axp(x) + \sum_{x\in S} bp(x) \\
&= a\sum_{x\in S} xp(x) + b\sum_{x\in S} p(x) \\
&= a\mathrm{E}[X] + b(1) \\
&= a\mathrm{E}[X] + b
\end{aligned}$$

□

The expected value of a random variable is a measure of its central tendency, where X will stay around. What it does not measure is how close X will stay. **Variance** is a measure of the spread of X, how far it usually is from it's expected value. It is almost a weighted average of X's distance from its mean.

Definition 1.28 (variance)**.** A weighted average of the squares of the distances of X from the mean, denoted $\mathrm{Var}(X)$.

$$\mathrm{Var}(X) = \sum_{x\in S}(x-\mathrm{E}[X])^2 p(x) = \mathrm{E}[(X-\mathrm{E}[X])^2]$$

It is usually more convenient to calculate variance by a different formula.

Theorem 1.29. *$Var(X) = E[X^2] - E[X]^2$.*

Proof.

$$\begin{aligned}
\mathrm{Var}(X) =& \mathrm{E}[(X - \mathrm{E}[X])^2] \\
=& \mathrm{E}[X^2 - 2X\mathrm{E}[X] + \mathrm{E}[X]^2] \\
=& \mathrm{E}[X^2] - 2\mathrm{E}[X\mathrm{E}[X]] + \mathrm{E}[\mathrm{E}[X]^2] && \text{Theorem 1.27} \\
=& \mathrm{E}[X^2] - 2\mathrm{E}[X]\mathrm{E}[X] + \mathrm{E}[X]^2 && \mathrm{E}[X] \text{ is a constant, Theorem 1.27} \\
=& \mathrm{E}[X^2] - \mathrm{E}[X]^2 && \square
\end{aligned}$$

Example 1.30. Calculate the variance of the pdf in Table 1.1 on page 17 by the definition of variance, and by the formula from Theorem 1.29.

Recall that $\mathrm{E}[X] = \frac{7}{8}$. Using the definition of variance,

$$\begin{aligned}
\mathrm{Var}(X) =& \mathrm{E}[(X - \mathrm{E}[X])^2] \\
=& \sum_{x=0}^{3} \left(x - \frac{7}{8}\right)^2 p(x) \\
=& \left(0 - \frac{7}{8}\right)^2 \left(\frac{1}{2}\right) + \left(1 - \frac{7}{8}\right)^2 \left(\frac{1}{4}\right) + \left(2 - \frac{7}{8}\right)^2 \left(\frac{1}{8}\right) + \left(3 - \frac{7}{8}\right)^2 \left(\frac{1}{8}\right) \\
=& \left(-\frac{7}{8}\right)^2 \left(\frac{1}{2}\right) + \left(\frac{1}{8}\right)^2 \left(\frac{1}{4}\right) + \left(\frac{9}{8}\right)^2 \left(\frac{1}{8}\right) + \left(\frac{17}{8}\right)^2 \left(\frac{1}{8}\right) \\
=& \left(\frac{49}{64}\right) \left(\frac{1}{2}\right) + \left(\frac{1}{64}\right) \left(\frac{1}{4}\right) + \left(\frac{81}{64}\right) \left(\frac{1}{8}\right) + \left(\frac{289}{64}\right) \left(\frac{1}{8}\right) \\
=& \frac{49}{128} + \frac{1}{256} + \frac{81}{512} + \frac{289}{512} \\
=& \frac{568}{512} = \frac{71}{64} = 1.109375.
\end{aligned}$$

And using the formula from Theorem 1.29,

$$\begin{aligned}\mathrm{Var}(X) =& \mathrm{E}[X^2] - \mathrm{E}[X]^2 \\ =& \sum_{x=0}^{3} x^2 p(x) - \left(\frac{7}{8}\right)^2 \\ =& 0^2 \cdot \frac{1}{2} + 1^2 \cdot \frac{1}{4} + 2^2 \cdot \frac{1}{8} + 3^2 \cdot \frac{1}{8} - \frac{49}{64} \\ =& \frac{1}{4} + \frac{4}{8} + \frac{9}{8} - \frac{49}{64} \\ =& \frac{15}{8} - \frac{49}{64} = \frac{71}{64} = 1.109375.\end{aligned}$$

Using the formula from Theorem 1.29 for the variance took fewer steps utilizing easier arithmetic than using the definition of variance directly.

While expected value and variance give meaningful measurements of a probability distribution that provide one with an understanding of the location and spread of a probability distribution, they do not necessarily give a precise understanding of the shape of the distribution neither do they define the distribution. The reader may verify that the two distributions of Table 1.2 each have a mean of 0 and a variance of 2.

1.7.2 The uniform distribution

If S is a finite sample space, then a **uniform distribution** is one where each s has the same likelihood.

Definition 1.31 (uniform distribution)**.** A distribution whose pdf $p(s) = c$, some fixed constant, for every $s \in S$.

Table 1.2: Two distributions with equal means and variances.

x	$p_1(x)$	$p_2(x)$
-2	0.2	0.25
-1	0.2	0
0	0.2	0.5
1	0.2	0
2	0.2	0.25

Since the sum of all probabilities in a distribution must be 1, it is clear that in the uniform distribution the constant probability, c, must be $\frac{1}{|S|}$. The standard primitive uniform distribution covers the integers 0 to n, where $p(x) = \frac{1}{n+1}$ for each $x \in \{0, 1, \ldots, n\}$. Any other discrete uniform distribution on a finite set of consecutive integers is just a shift of this one.

Theorem 1.32. *If X is a discrete random variable with a uniform distribution on the integers from 0 to n, then $E[X] = \frac{n}{2}$ and $Var(X) = \frac{n(n+2)}{12}$.*

Proof.

$$\begin{aligned} \mathrm{E}[X] =& \sum_{x=0}^{n} xp(x) = \sum_{x=0}^{n} x\left(\frac{1}{n+1}\right) \\ =& \frac{1}{n+1}\sum_{x=0}^{n} x = \frac{1}{n+1}\frac{n(n+1)}{2} = \frac{n}{2} \end{aligned}$$

$$\begin{aligned}\mathrm{Var}(X) =& \mathrm{E}[X^2] - \mathrm{E}[X]^2 = \sum_{x=0}^{n} x^2 p(x) - \left(\frac{n}{2}\right)^2 = \sum_{x=0}^{n} x^2 \left(\frac{1}{n+1}\right) - \frac{n^2}{4} \\ =& \frac{1}{n+1}\sum_{x=0}^{n} x^2 - \frac{n^2}{4} = \frac{1}{n+1}\frac{n(n+1)(2n+1)}{6} - \frac{n^2}{4} \\ =& \frac{2n^2+n}{6} - \frac{n^2}{4} = \frac{n^2+2n}{12} = \frac{n(n+2)}{12}\end{aligned}$$

□

1.7.3 The geometric distribution

The **geometric distribution** models the situation where an experiment is tried repeatedly until success is achieved. Each trial is considered independent of the others. The probability of success each time is denoted by p and the probability of failure, $1 - p$, is denoted by q. The random variable, X, represented by the geometric distribution tells on which attempt the first success occurs. X can take on any positive integer value. The geometric distribution, while discrete, is not finite. The probability distribution function of a geometric distribution is $p(x) = pq^{x-1}$ for all positive integers, x, and 0 for any other value. The experiment must fail with probability q for the first $x - 1$ trials then succeed on the x^{th} trial with probability p.

Theorem 1.33. *$p(x) = pq^{x-1}$ is a valid probability distribution function when $0 < p < 1$, $q = 1 - p$, and x takes only positive integer values.*

Proof. Clearly $p(x)$ is never negative, as both p and q are positive and x is a positive integer.

It remains to verify that $\sum_{x=1}^{\infty} p(x) = 1$. Remembering that $0 < q < 1$,

$$\begin{aligned}\sum_{x=1}^{\infty} p(x) =& \sum_{x=1}^{\infty} pq^{x-1} = p\sum_{x=1}^{\infty} q^{x-1} = p\sum_{i=0}^{\infty} q^i \\ =& p\left(\frac{1}{1-q}\right) = \frac{p}{1-(1-p)} = 1.\end{aligned}$$

□

If one is rolling a fair die for a 6, one might expect the first 6 to have an average appearance on the third roll (half of the six possibilities). It would average on the third or fourth roll if numbers couldn't be re-rolled. Because trials can fail with the same outcome multiple times without affecting future trials, the average first appearance of the 6 (or any other desired side of the die) will be on the sixth roll. Likewise the average first head (or tail) on a fair coin will happen on the second flip. In general, if the probability of success is p, then the first success will average on the $\left(\frac{1}{p}\right)^{\text{th}}$ trial, as demonstrated below.

Theorem 1.34. *If X is a random variable with a geometric distribution, then $E[X] = \frac{1}{p}$ and $Var(X) = \frac{q}{p^2}$.*

Proof.

$$\begin{aligned}\mathrm{E}[X] =& \sum_{x=1}^{\infty} xp(x) = \sum_{x=1}^{\infty} xpq^{x-1} = p\sum_{x=1}^{\infty} xq^{x-1} = p\sum_{i=0}^{\infty}(i+1)q^i \\ =& p\sum_{i=0}^{\infty}(q^i + iq^i) = p\left(\sum_{i=0}^{\infty} q^i + \sum_{i=0}^{\infty} iq^i\right) = p\left(\frac{1}{1-q} + \frac{q}{(1-q)^2}\right) \\ =& p\left(\frac{1}{p} + \frac{q}{p^2}\right) = \frac{p}{p} + \frac{q}{p} = \frac{p+q}{p} = \frac{1}{p}\end{aligned}$$

$$\begin{aligned}
\mathrm{E}[X^2] =& \sum_{x=1}^{\infty} x^2 p(x) = \sum_{x=1}^{\infty} x^2 p q^{x-1} = p \sum_{i=0}^{\infty} (i+1)^2 q^i = p \sum_{i=0}^{\infty} (1 + 2i + i^2) q^i \\
=& p \left(\sum_{i=0}^{\infty} q^i + 2 \sum_{i=0}^{\infty} i q^i + \sum_{i=0}^{\infty} i^2 q^i \right) = p \left(\frac{1}{1-q} + \frac{2q}{(1-q)^2} + \frac{q(q+1)}{(1-q)^3} \right) \\
=& \frac{p^2}{p^2} + \frac{2pq}{p^2} + \frac{q^2+q}{p^2} = \frac{p^2 + 2pq + q^2 + q}{p^2} = \frac{(p+q)^2 + q}{p^2} = \frac{q+1}{p^2}
\end{aligned}$$

$$\mathrm{Var}(X) = \mathrm{E}[X^2] - \mathrm{E}[X]^2 = \frac{q+1}{p^2} - \left(\frac{1}{p}\right)^2 = \frac{q+1}{p^2} - \frac{1}{p^2} = \frac{q}{p^2}$$

□

Example 1.35. An experiment consists of rolling a fair, six-sided die until a one is rolled. What are the expected value and variance of the number of rolls it takes?

With six sides on the die and each side being equally likely, $p = \frac{1}{6}$ and $q = \frac{5}{6}$. So the expected number of rolls is $\mathrm{E}[X] = \frac{1}{p} = 6$, and the variance is $\mathrm{Var}(X) = \frac{q}{p^2} = \frac{5}{6} \cdot 36 = 30$.

Chapter 2

Affine Character Cyphers

Affine character cyphers are the oldest and most simple. These act on one character at a time with the elementary operations of addition and multiplication.

Permutation cyphers, where each encyphering transformation is merely an element of the symmetric group on a set X (to be defined later) could be considered. Using a cryptosystem based on a general permutation cypher would require the storage and use of a complete cypher table, whereas affine cyphers can be defined with only a few parameter values. The tradeoff for this is that only a fraction of the permutations are able to be modeled by these simple operations.

2.1 Shift Cyphers

One of the earliest recorded users of encyphered messages is Julius Caesar.[1] The Caesar Cypher is a simple one, where the transformation from plaintext to cyphertext merely involves replacing each character with the one three positions further in the alphabet. See Table 2.1 for the transformation.

Table 2.1: Caesar Cypher transformation.

p_i	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
$f(p_i)$	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Before encyphering, plaintext is stripped of spaces and punctuation, and the characters are arranged in blocks of five. Each character is then transformed according to Table 2.1 to generate the cyphertext.

Example 2.1. Suppose, now, it is desired to encypher the message

```
I will cross the Rubicon next week,
```

with the Caesar Cypher. Arranging the letters in blocks of five and ignoring capitalization results in

```
IWILL CROSS THERU BICON NEXTW EEK.
```

[1] Rosen, 189. [7]

Looking up each letter in Table 2.1, each plaintext letter can be converted into it's corresponding cyphertext letter. For example, $f(\mathrm{A}) = \mathrm{D}$, $f(\mathrm{J}) = \mathrm{M}$ and $f(\mathrm{Q}) = \mathrm{T}$. The corresponding cyphertext is

LZLOO FURVV WKHUX ELFRQ QHAWZ HHN.

Decyphering is accomplished by applying the inverse transformation first, and then reinterpreting the spacing and punctuation.

LZLOO FURVV WKHUX ELFRQ QHAWZ HHN

$\downarrow f^{-1}$

IWILL CROSS THERU BICON NEXTW EEK

$\downarrow$ interpret

I will cross the Rubicon next week.

The blocking of the letters into clusters of a fixed length has twofold purpose. First, an adversary attempting to break the cypher could use the information from the original spacing to his advantage. Second, for ease of reading, since a long, unbroken string of characters can be difficult to read. The blocking into clusters isn't used when the space is in the alphabet, but then its associated cyphertext character will not likely be itself. The reasons for using all capital letters is also twofold. First, ancient Latin didn't have lowercase letters (though it didn't have the letter 'J' either). Secondly, a 26-letter **alphabet** was chosen for this example.

Definition 2.2 (alphabet)**.** A set, X, of characters used to build the plaintext and/or cyphertext.

While ‘A’ and ‘a’ are usually considered as the same letter, they are two distinct characters. The alphabet used in this past example had 26 characters, ‘A’ through ‘Z’ in the usual order, and no others. This required the plaintext and cyphertext to be spelled with only capital letters. It shall become evident why the use of a 52-character alphabet with uppercase and lowercase letters could give an adversary extra information to use to break this cypher, especially on a message with multiple sentences. The 26 characters occurring most often would likely be the lowercase letters, and the other 26 the uppercase letters.

The Caesar Cypher is an example of a **shift cypher** whose shift parameter is 3. Any shift parameter could be used (though a shift of 0 would be neither interesting nor effective). Any message using this 26-character alphabet has only 26 possible shift cyphers.

Definition 2.3 (shift cypher)**.** A cypher scheme where each character of the plaintext is replaced be the character a specified distance away in the alphabet to generate the cyphertext.

This definition of a shift cypher can be brought into an algebraic context. First, we use a bijection to associate each character of the l-character alphabet with an element of $\mathbb{Z}_l$. Table 2.2 shows a standard assignment for the 26-character alphabet used in the previous example.

Table 2.2: 26-character alphabet.

X	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
$\mathbb{Z}_{26}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

With this bijection in mind, each character of the alphabet, X, can be thought of as equivalent to its associated element of $\mathbb{Z}_{26}$. In other words, the ring structure of $\mathbb{Z}_{26}$ can be copied onto

X. It can now be said, for example, that D = A + 3, and so the encyphering transformation, f, can be defined with an algebraic expression, namely $f(p) = p + 3$ for the Caesar Cypher. A more algebraic definition of shift cypher may now be given.

Definition 2.4 (shift cypher)**.** A **shift cypher** on an alphabet represented by an additive group G is one whose encyphering transformation, $f : G \to G$, is of the form $f(p) = p + b$, where $b \in G$.

Example 2.5. Encypher the message "Meet me at the drop spot at noon," using the standard 26-character alphabet and a shift cypher with $b = 10$.

Write the characters in the chosen alphabet and in blocks of a fixed length (five).

```
MEETM EATTH EDROP SPOTA TNOON
```

Apply the encyphering transformation $f(p) = p + 10$ to each character.

```
WOODW OKDDR ONBYZ CZYDK DXYYX
```

Decyphering cyphertext into plaintext requires the use of f^{-1}. For a shift cypher, clearly $f^{-1}(c) = c - b$. Anyone knowing what b was used to encypher a message will be able to decypher it. This b is the **key** of the shift cypher.

Definition 2.6 (key)**.** The value(s) of the parameter(s) of a given encryption scheme used for a specific encyphering.

Any cryptosystem where inverse keys are easy to compute from each other is called a **classical** or **symmetric**. Cryptosystems where inverse keys are not easy to compute from each other are discussed in Chapter 4.

Definition 2.7 (Symmetric cryptosystem)**.** A cryptographic system where knowledge of the encyphering key and the decyphering key are equivalent or computationally "easy."

Example 2.8. Knowing that "WOODW OKDDR ONBYZ CZYDK DXYYX" was encyphered in the standard 26-character alphabet with a shift cypher and a key of 10, decypher it to recover the original message.

Apply the decyphering transformation $f^{-1}(c) = c - 10$ to each character.

```
MEETM EATTH EDROP SPOTA TNOON
```

Spacing and punctuation may now be interpreted.

```
Meet me at the drop spot at noon.
```

2.2 Linear Cyphers

Rings, such as $\mathbb{Z}_l$, have two operations. In the shift cypher, addition is used. In the **linear cypher**, multiplication is used. The encyphering transformation of a linear cypher is of the form $f(p) = ap$. Remembering that f must be an invertible transformation, so that any cyphertext generated by it may be uniquely decyphered, a must have a multiplicative inverse, so that $f^{-1}(c) = a^{-1}c$ can decypher cyphertext.

Definition 2.9 (linear cypher)**.** A linear cypher on an alphabet represented by the ring R is one whose encyphering transformation, $f : R \to R$, is of the form $f(p) = ap$, where $a \in R^*$.

Example 2.10. Encypher the following plaintext with a linear cypher. Use a 30-character alphabet where A-Z are represented by 0-25 as before, (space)=26, ‘,’=27, ‘.’=28, and ‘?’=29. Use the encyphering key $a = 7$.

```
The eagle lands at dawn.  Will you be ready?
```

Since the space is a member of the alphabet, the letters should not be rearranged. Additionally, note that two spaces follow the period between the sentences. Write the plaintext in the alphabet to be used.

```
THE EAGLE LANDS AT DAWN. WILL YOU BE READY?
```

Convert the plaintext to elements of $\mathbb{Z}_{30}$.

$$(19, 7, 4, 26, 4, 0, 6, 11, 4, 26, 11, 0, 13, 3, 18, 26, 0, 19, 26, 3, 0, 22, 13,$$

$$28, 26, 26, 22, 8, 11, 11, 26, 24, 14, 20, 26, 1, 4, 26, 17, 4, 0, 3, 24, 29)$$

Apply the encyphering transformation $f(p) = 7p$.

$$(13, 19, 28, 2, 28, 0, 12, 17, 28, 2, 17, 0, 1, 21, 6, 2, 0, 13, 2, 21, 0, 4,$$

$$1, 16, 2, 2, 4, 26, 17, 17, 2, 18, 8, 20, 2, 7, 28, 2, 29, 28, 0, 21, 18, 23)$$

Convert the elements of $\mathbb{Z}_{30}$ into the cyphertext.

```
NT.C.AMR.CRABVGCANCVADBQCCD RRCSIUCH.C?.AVSX
```

2.3 Affine Cyphers

The affine cypher combines the linear and shift cyphers, into a transformation of the form $f(p) = ap + b$. Or, rather, the shift and linear cyphers are special cases of the affine cypher where $a = 1$ and $b = 0$ respectively. The key of an affine cypher is an ordered pair, (a, b).

Theorem 2.11. *The composition of two affine cyphers is an affine cypher.*[2]

Proof.

$$\begin{aligned} f_2(f_1(p)) =& f_2(a_1p + b_1) \\ =& a_2(a_1p + b_1) + b_2 \\ =& (a_2a_1)p + (a_2b_1 + b_2) \end{aligned}$$

□

Just as shift and linear cyphers are special cases of the affine cypher, as special cases of Theorem 2.11 the composition of two shift cyphers is a shift cypher (where $a_2 = a_1 = 1$) and the composition of two linear cyphers is a linear cypher (where $b_2 = b_1 = 0$).

2.4 Abstractions

2.4.1 Ring Choices

All of the examples so far, and most of the considerations, have been using a cyclic ring of integers (mod l), $\mathbb{Z}_l$, as the algebraic representation of the alphabet set, X. However, any ring

[2]Further analysis of the algebraic structure of affine cyphers will be developed in section 2.4.

of the same size as X will do. In addition to modular rings of integers, finite polynomial rings may be used. Table 2.3 illustrates a finite polynomial ring representing a 32-character alphabet. This ring has 15 units, which are indicated in the third column. A finite polynomial field would have a larger group of units.

Being a unit in a ring is equivalent to not being a multiple of any irreducible zero divisors. The irreducible zero divisors of the ring in Table 2.3 are $t-1$ and $t^4+t^3+t^2+t+1$. Since the ring only has five dimensions, being a multiple of $t^4+t^3+t^2+t+1$ would mean being $t^4+t^3+t^2+t+1$. Being a multiple of $t-1$ means 1 is a zero of the polynomial, which would mean having an even number of terms. So the units in Table 2.3 are all the polynomials with an odd number of terms other than $t^4+t^3+t^2+t+1$.

2.4.2 Group of keys

Once an appropriate ring, R, is chosen to represent the alphabet being used, X, the set of all possible affine character cyphers, $f(p) = ap + b$, is determined. b can be any element of R, and a can be any element of R^*. Each unique cypher can be represented by its key, (a, b). Let K represent the set of all affine character cypher keys for a particular ring, R, thus $K = \{(a, b) \colon a \in R^*, b \in R\}$. Theorem 2.11 established that the composition of affine character cyphers in a ring is closed. This structure may be imposed onto K as a binary operation.

Theorem 2.12. *K is a group under this imposed binary operation and the product of two keys is the key of the composition cypher.*

Proof. Using the result from Theorem 2.11, if $k_1 = (a_1, b_1)$ and $k_2 = (a_2, b_2)$, then the result of encyphering with k_1 first, followed by k_2, is equivalent to a single encyphering with the key

Table 2.3: 32-character alphabet represented by the polynomial ring $\frac{\mathbb{F}_2[t]}{\langle t^5-1\rangle}$

Character	Ring element	Unit
A	0	No
B	1	Yes
C	t	Yes
D	$t+1$	No
E	t^2	Yes
F	t^2+1	No
G	t^2+t	No
H	t^2+t+1	Yes
I	t^3	Yes
J	t^3+1	No
K	t^3+t	No
L	t^3+t+1	Yes
M	t^3+t^2	No
N	t^3+t^2+1	Yes
O	t^3+t^2+t	Yes
P	t^3+t^2+t+1	No
Q	t^4	Yes
R	t^4+1	No
S	t^4+t	No
T	t^4+t+1	Yes
U	t^4+t^2	No
V	$^{34}\,t^4+t^2+1$	Yes
W	t^4+t^2+t	Yes
X	t^4+t^2+t+1	No
Y	t^4+t^3	No

$(a_2a_1, a_2b_1 + b_2)$. Let this result be the definition of k_2k_1. Since $a_2a_1 \in R^*$ and $a_2b_1 + b_2 \in R$, then $k = k_2k_1 \in K$. The identity is $(1, 0)$:

$$(1,0)(a,b) = (1a, 1b + 0) = (a,b) = (a \cdot 1, a \cdot 0 + b) = (a,b)(1,0).$$

The inverse of (a, b) is $(a^{-1}, -a^{-1}b)$:

$$(a,b)(a^{-1}, -a^{-1}b) = (aa^{-1}, a(-a^{-1}b) + b) = (1,0) = (a^{-1}a, a^{-1}b - a^{-1}b) = (a^{-1}, -a^{-1}b)(a,b).$$

Thus K has a group structure mimicking the group of encyphering and decyphering transformations themselves. □

This group can be represented in matrix form as

$$K \cong \begin{bmatrix} R^* & R \\ 0 & 1 \end{bmatrix}$$

with standard matrix multiplication as the operation.

$$\begin{bmatrix} a_2 & b_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} a_2a_1 & a_2b_1 + b_2 \\ 0 & 1 \end{bmatrix}$$

With the formula for the determinant of a 2×2 matrix, it is clear that valid keys require an invertible a, and the value of b is irrelevant to invertibility.

When considering the character cypher transformations, shift and linear cyphers were considered special cases of the affine cypher. Therefore, the sets of keys yielding shift and linear cyphers are subsets of this group K. Let the set of all shift cypher keys in K be denoted by $S = \{(1, b) \in K\}$, and the linear cypher keys in K by $L = \{(a, 0) \in K\}$. Clearly $S, L \subseteq K$, but it can also be easily verified that $S, L \leq K$. Furthermore $S \trianglelefteq K$

Theorem 2.13. *The group of shift cyphers is a normal subgroup of the group of affine cyphers.*

Proof. Clearly the identity is in S, and the inverse of a shift is another shift. Closure may easily verified by the reader. For normality, using the inverse and composition formulas stated in Theorem 2.12,

$$(a, b)^{-1}(1, b_0)(a, b) = (a^{-1}, -a^{-1}b)(a, b + b_0) = (1, a^{-1}b_0) \in S.$$

Therefore $S \trianglelefteq K$. □

S is an Abelian group by the commutativity of addition in R, and L will also be Abelian if multiplication in R is commutative, but K is not Abelian. As sets, $K \cong L \times S$, but not as groups. Each $(a, 0) \in L$ can represent an element of $\mathrm{Aut}(S)$, specifically, left multiplication of b by a in R. As groups, $K \cong S \ltimes L$. Although, in this sense, one might want to list the components of an element of K in the reverse order: (b, a) instead of (a, b). Written that way, the definition of multiplication in K looks exactly like a semi-direct product.

$$(b_2, a_2)(b_1, a_1) = (b_2 + a_2b_1, a_2a_1)$$

At this point, encyphering and decyphering can be phrased in terms of group actions of keys, K, on an alphabet, X. Encyphering and decyphering are actions of inverse keys. The only distinction, at this point, is one of order. Given a key, k, and its inverse, k^{-1}. If one applies k first, then k^{-1} to a character of plaintext, x: $k^{-1} \cdot (k \cdot x)$, then k is called the encryption key and k^{-1} is called the decryption key. If they are used in the other order, then k^{-1} is called the encryption key and k the decryption key.

2.4.3 Text Vectors And Sequences

For a given alphabet, X, a **message** (whether plaintext or cyphertext) is a finite sequence of elements of X; $(x_i)_{i=1}^n$ for some $n \in \mathbb{N}$. The message could also be considered a finite-length vector over X, $[x_i]_{i=1}^n \in X^n$. Vector notation is more compact, and so will be preferred, though sequence nomenclature and verbiage is more natural to the situation.

Definition 2.14 (message)**.** Given an alphabet set, X, a message written in X is a finite length vector element of X^n. The message may also be equated with the same length sequence of the same alphabet characters in the same order:

$$[x_i]_{i=1}^n \cong (x_i)_{i=1}^n.$$

Each message is an element of X^n. X^n is the set of all messages of length n. With the structure of a ring, R, imposed through a bijection onto X, then X^n is also a set of n-dimensional vectors. The set of all messages, $\mathcal{P}$ and/or $\mathcal{C}$, is the infinite union $\cup_{n=1}^{\infty} X^n$.

Earlier, encyphering was described as a group action on the alphabet, X. With that group action already defined, then the action of encyphering on a vector of characters, $x \in X^n$, can be described as distributing the group action to each entry in the vector, just like multiplication by a scalar in R. Encyphering is now a group action of the keyspace, K, on the message space, $\mathcal{P}$. However, because the only difference between the encyphering group action on a vector and on a character is a repetition of the encyphering function n times, the base action on the character is all that needs to be analyzed.

2.5 Breaking and Time Analyses

The number of shift cyphers is equal to the number of possible shifts (b), the number of elements of R. A b of zero is neither interesting nor effective, but $|R| - 1$ is still $\mathcal{O}(|R|)$. So the size of the keyspace of a shift cypher is $\mathcal{O}(|R|)$.

The number of linear cyphers is equal to the number of elements in R^*. When $R = \mathbb{Z}_l$, then this is equal to $\phi(l)$, the Euler Totient (or Phi) Function.[3] The actual value of $\phi(l)$ depends greatly on the prime factorization of l. The more distinct prime factors l has, the smaller $\phi(l)$ will be. However, if l is prime then $\phi(l) = l-1$ and if l is the product of two distinct primes, then $\phi(l)$ is still respectably big. Since time estimates should cover "worst case" scenarios, and it is desirable in cryptographic settings to use rings with large groups of units, and the ring to work in may be freely chosen, it can be said that $l = \mathcal{O}(\phi(l))$ and $\phi(l) = \mathcal{O}(l)$. As a generalization, it shall be considered that $|R| = \mathcal{O}(|R^*|)$. So the size of the keyspace of a linear cypher is $\mathcal{O}(|R|)$.

The keyspace of an affine cypher is the semi-direct product of the keyspaces of the shift and linear cyphers. Its size is equal to the product of the two constituents' sizes. So the size of the keyspace of an affine cypher is $\mathcal{O}(|R|^2)$.

To use an affine cypher (or either of its special cases), an element of the keyspace must be chosen at random then up to one multiplication and one addition are performed on each character. Choosing the key is a constant-time activity: some fixed number of random numbers in a fixed range are chosen. Performing the multiplication and addition require times that are dependent upon the ring being used. In the case of $\mathbb{Z}_l$, then the time estimates from Section 1.6.3

[3] The Euler Phi Function and it's properties should be adequately described in any introductory number theory text.

may be used. Thus, for any particular message with a fixed size, the running time to encypher it with an affine cypher is $\mathcal{O}((\lg l)^2 + \lg l) = \mathcal{O}((\lg l)^2)$. The running time naturally grows linearly with respect to the message length, but since all operations discussed here will be linear with respect to message length, that aspect may be ignored from all analyses.

2.5.1 Cypher Attacks

There are many different types of sophisticated analyses that can be performed on a cypher. This section will give a brief synopsis of three simple methods of attacking a cypher. The **brute force attack** is the simplest, just keep guessing keys and trying them until the correct one is found. It neither requires nor uses any information beyond knowledge of what cryptosystem was used. When analyzing a cryptosystem, it is always presumed that an attacker knows everything about the cryptosystem and its keyspace, just not which key was used. Academically, a **break** of a cryptosystem is deemed whenever a vulnerability is found that allows a key to be identified faster than brute force, even if employing such a break is practically infeasible.

Brute Force Attack

The brute force attack is the least sophisticated and most time consuming. Armed with no additional knowledge than the cryptosystem used and its affiliated keyspace, an attacker tries all keys until the correct one is found. This will eventually work, presuming that intelligible plaintext can be easily identified (such as actual linguistic text or a known binary file's header format).

Theorem 2.15. *The expected time to execute a brute force attack on a cypher will be linear with respect to the size of the keyspace.*

Proof. If the key used to generate a given cyphertext was chosen randomly and fairly, then each key had an equal chance of being chosen. Thus the probability of being the correct key is a uniform distribution over the keyspace, K, where each key has a probability of $p = \frac{1}{|K|}$ of being correct.

If an attacker is able to order the entire keyspace in any arrangement and systematically iterate across the list of keys, then each guess also has an equal likelihood: the first key tried, the third key in the sequence and the last key all have a probability of p of being correct. The probability of the attacker finding the key in n tries is exactly the probability that the n^{th} key is correct. Thus the number of guesses required will also be a uniform distribution on the set $\{1, 2, \ldots |K|\}$. The average number of tries will be the mean of a uniform distribution, $\frac{|K|+1}{2} = \mathcal{O}|K|$.

If an attacker is not able to order the entire keyspace (if, perhaps, it is too large to fit in his computer's memory and/or storage and too complex to define a systematic method of iteration) then the attacker must guess randomly. Without being able to remember which keys had already been guessed, then it becomes possible for the same key to be guessed multiple times. On each guess, the attacker will have a probability of p of guessing correctly. The number of guesses required to find the key now follows a geometric distribution. The average number of guesses required will be the mean of a geometric distribution, $\frac{1}{p} = |K| = \mathcal{O}(|K|)$.

And, therefore, the number of guesses required is $\mathcal{O}(|K|)$. Each guess will require applying a decyphering. The time complexity of the decyphering is dependent upon the cryptosystem and

representative ring used. For an affine cypher, each guess will require $\mathcal{O}((\text{one multiplication}) + (\text{one addition}))$ time. While the choice of ring can affect both the size of the keyspace and the complexity of the cypher transformations (*i.e.* $\mathbb{Z}_{29}$ will cause a larger keyspace and slightly longer operations than $\mathbb{Z}_{27}$), if the complexity of the (en-/de-)cyphering could be held constant while changing the size of the keyspace, the time for a brute force attack would grow linearly. □

Example 2.16. The time required for a brute force attack on an affine cypher using $\mathbb{Z}_l$ is equal to the product of the number of guesses, $\mathcal{O}(l^2)$, and the time to decypher with each guess, $\mathcal{O}((\lg l)^2)$, thus $\mathcal{O}(l^2(\lg l)^2)$.

Known Plaintext Attack

If the plaintext for a specific piece of cyphertext is known, then the key could be solved for algebraically. Set up the equation $c = f(p)$ and solve for the parameters of f. This is called a **known plaintext attack**. In practice, the knowledge of the plaintext could come from knowledge of a standard signature used by the sender of a text message or by the known header format of a binary file (ZIP, JPEG, MP3, DOC, ...) or other covert means.

For a shift cypher ($f(p) = p + b$) or a linear cypher ($f(p) = ap$) only one plaintext/cyphertext pair of characters is required to find the key. For an affine cypher ($f(p) = ap + b$) two plaintext/cyphertext pairs of characters are required to solve for the key, but having just one will still cut down the number of possible keys in a brute force attack to $\mathcal{O}(\sqrt{|K|})$.

Example 2.17. Suppose that in the cyphertext

```
KIYYO TKOQG FUBGF WITVI YVWTS
```

generated with an affine cypher in the standard 26-letter alphabet from Table 2.2 it is known that O represents E and that W represents I.

This gives the following system of equations.

$$14 = 4a + b$$

$$22 = 8a + b$$

Linear systems of equations in rings can be solved the same way as elementary systems in $\mathbb{R}$: with either substitution or elimination/addition, however, if the ring being worked in is not a field, there may be no solution or multiple ones even if the system is otherwise linearly independent. For a known plaintext attack, it is guaranteed that a key exists to map the plaintext letters to the cyphertext letters, but the linear system in the ring may still not have one unique solution.

Subtracting the two equations in $\mathbb{Z}_{26}$ gives $4a = 8$, which is equivalent to $2a = 4$ or 17. 17 is not a double in $\mathbb{Z}_{26}$, but 4 is, so $a = 2$ or 15. Substituting either value for a, or more simply substituting 8 for $4a$ gives $b = 6$. This system has two solutions: $(2, 6)$ and $(15, 6)$. To be a proper key, the a must be invertible. 2 is not invertible in $\mathbb{Z}_{26}$, so the key must have been $(15, 6)$. The inverse key is $(7, 10)$ and when it is applied to the cyphertext plain English results,

`COMMENCE SATURATION BOMBING.`

If only one pair of letters were known, say that O was the image of E, then for each possible a (only the units), only one b was possible and could be solved for. That would reduce the brute force attack from $13 * 25 = 325$ possibilities to only 13, but with the extra effort required in each try of solving for b (one modular subtraction).

Frequency Analysis

If no piece of plaintext is actually known, it could still be possible to guess some plaintext. An analysis of the cyphertext could imply some good guesses.

For example, the cyphertext of Example 2.17 has a double Y near the beginning. Few English letters are doubled often, so it would make sense to guess that the doubled letter was one of those, such as E, S or T.

An analysis of the distribution of characters in the cyphertext, and a comparison of that distribution with the already known distribution of characters in plaintext is a very common technique. This type of attack is called a **frequency analysis**. Spoken languages are not written with a uniform distribution of characters from their alphabets. In English, for example, E and T are the most common characters and Z, X and Q are the least common. A comparison of the cyphertext character distribution with a known character distribution of the plaintext language will also imply some good guesses.

The character distribution of the cyphertext from Example 2.8 is given in Table 2.4. The most commonly occurring character in the cyphertext is D. The most commonly occurring letter in English is E. It could be guessed that the encyphering key was -1. Decyphering with a key of -1 gives the following:

```
XPPEX PLEES POCZA DAZEL EYZZY.
```

Apparently -1 was not the key. Looking at the distribution again, if the key were -1 then the plaintext would have had to have a surprisingly large number of X's, Y's, and Z's. The second most popular letter in English is T. It could be guessed that the encyphering key was 10, and

this second guess is correct.

Table 2.4: Character distribution of cyphertext in Example 2.8.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	1	5	0	0	0	0	0	0	2	0	0	0	4	0	0	1	0	0	0	0	2	2	4	2

Example 2.18. Suppose it is known that the cyphertext

```
JKXMK ZFKJJ KXMJB SJGEJ BMUOM EJGKF
```

was generated by an affine cypher on the standard 26-letter alphabet. A statistical analysis on a larger portion of cyphertext revealed that 'M' and 'J' are the two most common letters. Knowing that the most popular letters in English plaintext are 'E' and 'T', find the encryption key and decrypt the previous snippet.

Assuming that $f(\mathrm{E} = 4) = \mathrm{M} = 12$ and $f(\mathrm{T} = 19) = \mathrm{J} = 9$ the following system of equations (in $\mathbb{Z}_{26}$) comes out

$$\begin{cases} 12 = a \cdot 4 + b \\ 9 = a \cdot 19 + b \end{cases}.$$

Which is easily solved:

$$\begin{bmatrix} 12 \\ 9 \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ 19 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ 19 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 12 \\ 9 \end{bmatrix}$$

$$= \begin{bmatrix} 19 & 7 \\ 3 & 24 \end{bmatrix} \begin{bmatrix} 12 \\ 9 \end{bmatrix} = \begin{bmatrix} 5 \\ 18 \end{bmatrix}$$

So the encryption key was $(5, 18)$. The decyphering transformation is $f^{-1}(c) = a^{-1}(c - b) = a^{-1}c - a^{-1}b$. So the decryption key is $(21, 14)$. Applying the decyphering transformation of $f^{-1}(c) = 21c + 14$ to the above cyphertext results in

```
TOBEO RNOTT OBETH ATIST HEQUE STION,
```

or rather

```
To be, or not to be, that is the question.
```

2.5.2 Composition of encypherings

Theorem 2.11 established that the composition of encypherings is closed. This is considered a weakness in a cryptosystem. One might think that encyphering a piece of plaintext twice would square, or at least double, the difficulty in breaking it, but when the set of keys form a group, no matter how many times a plaintext is encyphered with how many different keys, it is still no more difficult to break than if it was encyphered once with the single key of the encyphering composition.

The closure of affine encypherings under composition is with the assumption that both encypherings were performed in the same ring with the same bijection between the alphabet set, X, and the ring, R. If $|X_1| = |R_1| \leq |R_2| = |X_2|$, then one can generate injective maps $\iota_1 : X_1 \to R_1$, $\iota_2 : R_1 \to R_2$, and $\iota_3 : R_2 \to X_2$, and the composition $\iota_3 \circ f_2 \circ \iota_2 \circ f_1 \circ \iota_1$ will be an encyphering from the alphabet X_1 to the alphabet X_2. This will be fully invertible as it is the composition of injective functions.

Example 2.19. Encypher the following plaintext using the 32-letter alphabet described in Table 2.5 with two affine character transformations. Do the first encyphering in $\mathbb{Z}_{32}$ with a key of $(5, 12)$, and the second in $\frac{\mathbb{F}_2[t]}{\langle t^5-1\rangle}$ with a key of $(t^3 + t^2 + 1, t^2 + 1)$.

```
To be, or not to be:  that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
And by opposing end them?
```
[4]

The plaintext should first be written in the chosen alphabet (all capital letters and ignore line breaks).

```
TO BE, OR NOT TO BE: THAT IS THE QUESTION: WHETHER 'TIS NOBLER IN THE MIND TO
SUFFER THE SLINGS AND ARROWS OF OUTRAGEOUS FORTUNE, OR TO TAKE ARMS AGAINST A
SEA OF TROUBLES, AND BY OPPOSING END THEM?
```

Next, the intermediary cyphertext is computed with the first key.

[4]Shakespeare, William. *Hamlet, Prince of Denmark*, III.1

LSORATOSBONSLOLSORAHOLPMLOUGOLPAO.QAGLUSNHO

PALPABOCLUGONSRDABOUNOLPAOIUN,OLSOGQFFABOLPAOGDUNKGOMN,OMBBS

GOSFOSQLBMKASQGOFSBLQNATOSBOLSOLMKAOMBIGOMKMUN[5]

GLOMOGAMOSFOLBSQRDAGTOMN,OREOSXXSGUNKOAN,OLPAI?

Then, use the second key on the intermediary cyphertext to compute the final cyphertext.

ZJB'FEBJIBWJZBZJB'FHBZM,ZBGKBZMFBNTFKZGJWHBCMFZMFIB:ZGKBWJ'SFIBGWB[6]

ZMFBOGWPBZJBKT??FIBZMFBKSGWGKB,WPB,IIJCKBJ?BJTZI,UFJTKB?JIZTJFEBJI

BZJBZ,XFB,IOKB,U,GWKZB,BKF,BJ?BZIJT'SFKEB,WPB'QBJRRJKGJUBFJPBZMFOA

If the result of a first affine encyphering is injected into a different ring of equal or greater size, then put through a second affine encyphering, this composition will be harder to break. Assume that the pre-images of two cyphertext characters is already known, whether by knowledge of the form of the plaintext or a statistical analysis. What is not known is the intermediary cyphertext characters (the image of the two known plaintext letters through only the first encyphering transformation). For each of the possible $\mathcal{O}({l_1}^2)$ encryption keys for the first encyphering, there is up to one valid second encryption key of the $\mathcal{O}({l_2}^2)$ total that will map the two intermediary cyphertext characters to the correct two final cyphertext characters. And this will require a brute force check: for each possible decryption key for the second transformation, calculate the images of the two known cyphertext characters, solve the 2x2 system to find the decryption key for the first transformation (if it exists), apply the two decyphering transformations and see if

[5]This line break was inserted for legibility, while the previous two occurred naturally at single spaces in the cyphertext.

[6]Both of these line breaks were inserted for legibility, as the final cyphertext has no spaces in it.

Table 2.5: 32-character alphabet represented by $\mathbb{Z}_{32}$ and by the polynomial ring $\frac{\mathbb{F}_2[t]}{\langle t^5-1\rangle}$

Character	$\mathbb{Z}_{32}$	$\frac{\mathbb{F}_2[t]}{\langle t^5-1\rangle}$
A	0	0
B	1	1
C	2	t
D	3	$t+1$
E	4	t^2
F	5	t^2+1
G	6	t^2+t
H	7	t^2+t+1
I	8	t^3
J	9	t^3+1
K	10	t^3+t
L	11	t^3+t+1
M	12	t^3+t^2
N	13	t^3+t^2+1
O	14	t^3+t^2+t
P	15	t^3+t^2+t+1
Q	16	t^4
R	17	t^4+1
S	18	t^4+t
T	19	t^4+t+1
U	20	t^4+t^2
V	21	t^4+t^2+1
W	22	t^4+t^2+t
X	23	t^4+t^2+t+1
Y	24	t^4+t^3

legible plaintext results. This requires the same $\mathcal{O}(l^2(\lg l)^2)$ as the dumb (knowledgeless) brute force attack described in Section 2.5.1.

Example 2.20. The two most common alphabet characters in the plaintext in Example 2.19 are (space) occurring 38 times and T occurring 19 times. O occurs 18 times and E occurs 17 times. Assume that an attacker already knows two plaintext characters, say that (space) is represented by B and that T is represented by Z in the cyphertext. What the attacker does not know is the intermediary cyphertext characters B and L for (space) and T respectively. If those were known, then all that would need to be done is solve two 2×2 systems, one in each ring. However, without knowing those two characters, he must consider every possible pair of intermediary characters. For example, there are 32*15=480 possible keys in the polynomial ring and 512 possible keys in the cyclic ring. For each of the 480 keys, an attacker may calculate the pre-image, intermediary, cyphertext characters of B and Z, then solve the system in the cyclic ring to find the key (if it exists) that would send (space) and T to them. On average, this will take 240 tries, based on Theorem 2.15.

2.5.3 Distribution Count Variance

The United States' Declaration of Independence is reproduced in Appendix A. This is a decently sized document free of copyright. It will be a running example throughout this work. Each time it will be considered to be in a 128-letter alphabet, encoded by the standard 7-bit ASCII standard for the computer representation of text characters. Each uppercase letter, lowercase letter, decimal digit, punctuation mark, and other control character (such as tabs and new lines) have a standard numerical assignment. There are actually a number of standard character assignments

for alphabets in different languages (ANSI, ASCII, the ISO family, the Unicode family, etc.. . .), most of the encodings of Latin alphabets (especially the English alphabet) agree with the ASCII standard for most of the first 128 characters. This choice of alphabet is usually automatically handled by the text software (word processor, web browser, email program, etc. . . .) so in application, all that really needs to be considered is bits or bytes. The 26- to 30-character alphabets will continue to be used in small examples throughout this work for ease of human comprehension.

The character distribution of the plaintext Declaration of Independence is shown in Figure 2.1. The Declaration of Independence was encyphered with an affine character cypher, the result of which is also presented in Appendix A. Note the jumbling of the paragraph breaks, as the Carriage Return and Line Feed (new paragraph) characters are considered alphabetic elements. The character distribution of the affine cyphertext Declaration of Independence is also shown in Figure 2.1 for comparison. Note that the net effect between the two distributions is just a shuffling of the bars in the histogram. The characters in the cyphertext appear no more random than in the plaintext, they are just different characters.

The statistical measures of expectation and variance can be used to measure how evenly spread out the distribution of characters is in a plaintext or cyphertext. Actually, expectation (average) will not indicate any of that, but it is used in the formula for variance. An average of the counts of each character should come out as $\frac{total\ characters}{size\ of\ alphabet}$ regardless of whether it is the plaintext or the cyphertext, or of the encryption scheme used. The variance will give an indication of how non-uniform the character distribution is. The variance is the average of the squares of the distribution count minus the square of the average of the distribution count, per

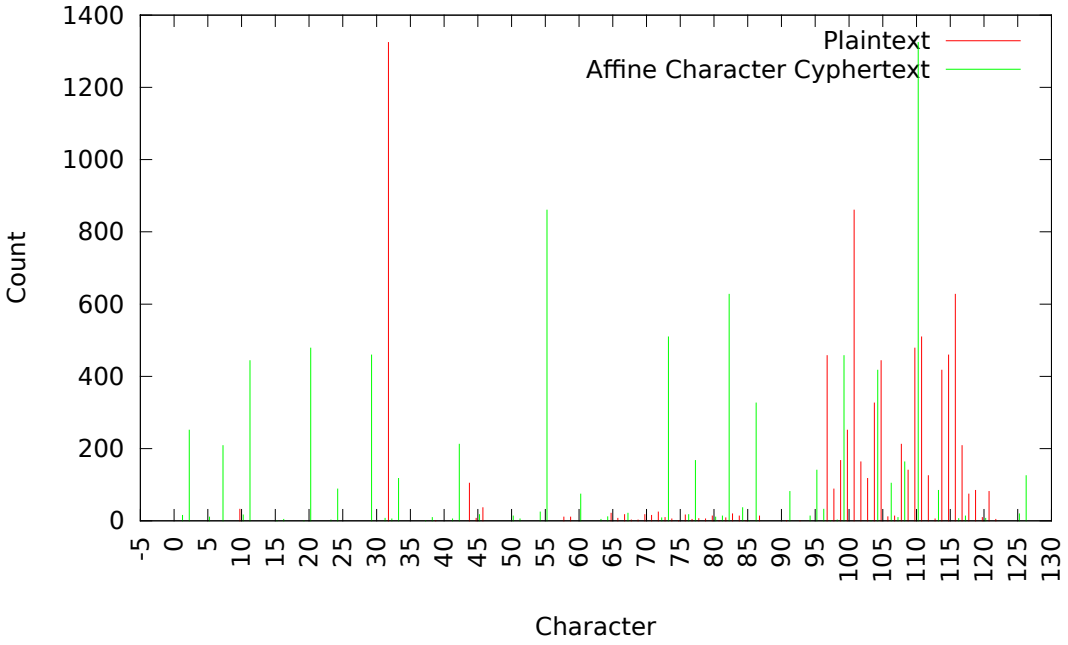

Figure 2.1: Character distributions of plaintext and an example affine cyphertext Declaration of Independence.

Theorem 1.29.

$$Variance = \frac{\sum_{a \in X} c(a)^2}{|X|} - \left(\frac{\sum_{a \in X} c(a)}{|X|} \right)^2,$$

where $c(a)$ is the count of the number of a's in the plaintext or cyphertext.

A higher variance means that some characters are occurring more often than other characters, a less uniform distribution of characters. A lower variance means that more characters occur approximately the same number of times, a more uniform distribution of characters. The more uniform the character distribution of a cyphertext is, the closer it appears to a random sequence, and the harder it is to analyze for information.

Example 2.21. Find the variance of the cyphertext from Example 2.8. The character distribution table is reproduced here.

First find the average of the character distribution.

$$\frac{1+1+5+2+4+1+2+2+4+2}{26} = \frac{24}{26} = \frac{12}{13} \approx 0.923$$

Find the average of the squares of the distribution.

$$\frac{1+1+25+4+16+1+4+4+16+4}{26} = \frac{76}{26} = \frac{38}{13} \approx 2.92$$

Subtract the square of the average from the average of the squares.

$$\frac{38}{13} - \left(\frac{12}{13} \right)^2 = \frac{494}{169} - \frac{144}{169} = \frac{350}{169} \approx 2.07$$

The variance of the character distributions of both the plaintext and the cyphertext of affine character transformations are the same. For the text of the Declaration of Independence, the variance of both distributions shown in Figure 2.1 is approximately 31707.

Table 2.6: Character distribution of cyphertext in Example 2.8 (reproduced).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	1	5	0	0	0	0	0	0	2	0	0	0	4	0	0	1	0	0	0	0	2	2	4	2

Chapter 3

Affine, Block Cyphers

There are a number of mutually related disadvantages with the character cyphers described in Chapter 2. First is the small number of possible encryption keys. Second is the ease of a statistical analysis on the cyphertext to break it. This is because of the small alphabet, and that every 'A' in the plaintext gets mapped to the exact same cyphertext character. Both of these weaknesses may be overcome by a simple alteration. Rather than encyphering one character at a time, in a ring the size of the alphabet, blocks of m characters are associated together and encyphering is done in a ring magnitudes larger than the size of the alphabet. Specifically, if the alphabet has l characters, then the ring will have l^m elements.

This modification drastically increases the number of possible encryption keys; the larger rings will have a larger number of invertible elements for the multiplicative a component and a larger number of total elements for the additive b component. This modification also diversifies the image of each character, as encyphering happens block-by-block rather than character-by-

character, thus increasing the statistical analysis required for breakage. What character any specific 'A' in the plaintext gets mapped to by the encyphering, will now depend on where in the block the 'A' occurs and the other characters in the block.

3.1 Multiple Digit m-graphs

Definition 3.1 (m-graph)**.** A block of m letters in a plaintext or cyphertext. 2-graphs and 3-graphs are also called digraphs and trigraphs.

One simple way to inject a block of m characters of a size l alphabet into a cyclic ring, is by considering that block as an m-digit number in base l. Each block of m letters will be mapped to an element of the ring $\mathbb{Z}_{l^m}$. Given a bijection from the alphabet to the ring, $j : X \to \mathbb{Z}_l$, one can define $j_m : X^m \to \mathbb{Z}_{l^m}$ by $j_m(a_1, a_2, \ldots, a_m) = \sum_{i=1}^{m} a_i l^{m-i}$.

Affine transformations in $\mathbb{Z}_{l^m}$ happen just the same as ones in $\mathbb{Z}_l$, except there are now more choices for the parameters a and b. There are l^m possibilities for b and $\phi(l^m) = l^{m-1}\phi(l)$ possibilities for a. So the total number of possible keys is $\mathcal{O}(l^{2m})$. The number of possible encryption keys grows exponentially with the size of the block.

Example 3.2. Encypher the message

`RUBBER BABY BUGGY BUMPERS`

using the standard 26-letter alphabet from above with a multidigit affine transformation on digraphs using the encryption key $(93, 521)$. Note that $26^2 = 676$ and that our message has an above average number of B's. Character distribution tables for both the plaintext and cyphertext appear at the end of the example.

First write the code in blocks of two.

```
RU BB ER BA BY BU GG YB UM PE RS
```

Convert each digraph to an element of $\mathbb{Z}_{676}$ with the aid of Table 2.2.

```
17:20 1:1 4:17 1:0 1:24 1:20 6:6 24:1 20:12 15:4 17:18

462 27 121 26 50 46 162 625 532 394 460
```

Apply the encyphering transformation $f(p) = 93p + 512$.

```
223 328 282 235 439 67 39 510 649 659 37
```

Convert back to digraphs in the 26-letter alphabet.

```
8:15 12:16 10:22 9:1 16:23 2:15 1:13 19:16 24:25 25:9 1:11

IP MQ KW JB QX CP BV TQ YZ ZJ BL
```

Arrange the cyphertext in blocks of five for easy human reading.

```
IPMQK WJBQX CPBNT QYZZJ BL
```

Table 3.1: Character distributions of Example 3.2.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
plaintext	1	6	0	0	2	0	2	0	0	0	0	0	1	0	0	1	0	3	1	0	3	0	0	0	2	0
cyphertext	0	3	1	0	0	0	0	0	1	2	1	1	1	1	0	2	3	0	0	1	0	0	1	1	1	2

The character distributions for Example 3.2 are in Table 3.1. Example 3.2 started with a plaintext containing 6 B's. Those 6 B's were mapped by the encyphering transformation to 4 different letters: M, Q, J, and C, and the three B's that are in the cyphertext came from three different plaintext letters: A, G, and R. The letters in the cyphertext are more spread out than in the plaintext. This is visible in the histogram of Figure 3.1. It can also be demonstrated by finding the variance of the counts of the letters in each. Using the formula from Theorem 1.29 it is found that the variance in the character counts in the plaintext is $\frac{1+36+4+4+1+1+9+1+9+4}{26} - (\frac{22}{26})^2 \approx 1.98$, but the variance in the character counts in the cyphertext is about 0.822.

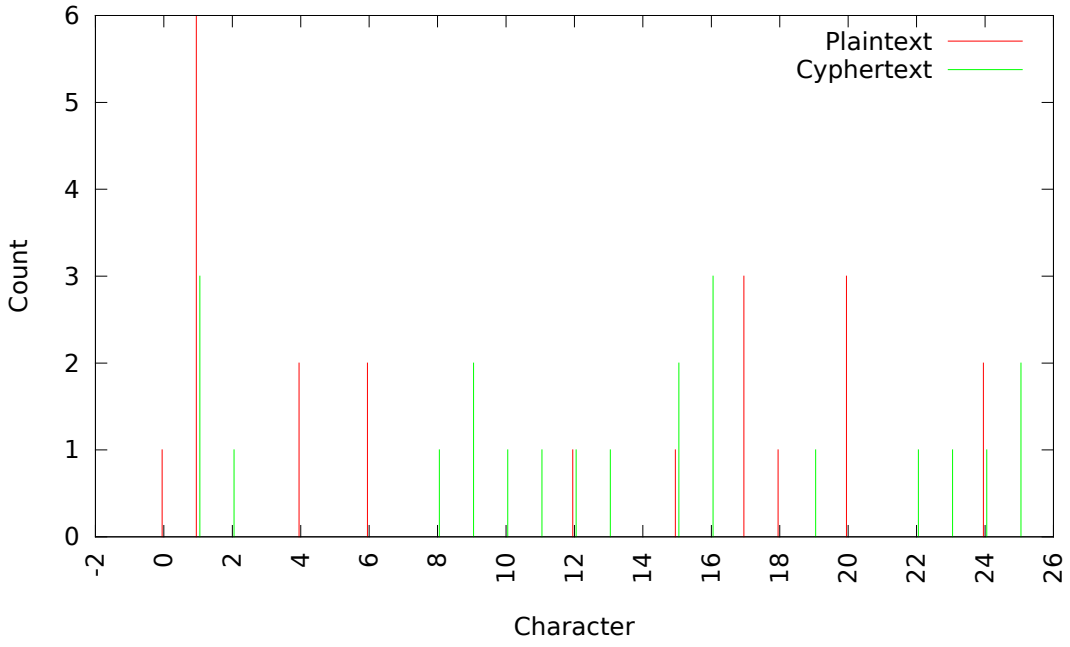

Figure 3.1: Character distributions of Example 3.2.

That the third B in the plaintext went to a Q is a coincidence, but it is not a coincidence that the second and sixth B's went to the same letter. The least significant digit in an affine transformation is not affected by the more significant digits, this is true even in base 10. The ones digit of any multiplications and additions is only determined by the ones digits of the components. Notice also that the first two U's, each in the least significant digit of their respective digraphs, each went to P.

The fact that the image of each letter in the least significant digit of each block is constant with respect to the more significant digits is a weakness of this block cypher scheme. If the alphabet and block size are known, which is standard, an attacker could gain information by a statistical analysis of only the least significant digits of each block. With enough statistical data, an attacker could determine the residue of the encryption key mod l.

For known plaintext attack, an attacker would need to know the plaintext equivalents of 2 blocks of cyphertext, just as he needed to know the plaintext equivalent of two characters of cyphertext for an affine, character transformation. This could be more than m times harder, because it requires two properly aligned, length-m blocks of contiguous characters to be known, not just m times as many characters.

A statistical analysis can still be performed if enough cyphertext is available. Rather than counting the number of each character, the number of each possible length-m block must be counted and compared to a known m-graph distribution in the plaintext alphabet. Because the data being counted are m times as big as with the character cyphers, and there are l^{m-1} times as many possible data elements, it will now take at least ml^{m-1} times as much cyphertext to be able to make as accurate of a statistical analysis as it did against single character cyphers.

Affine transformations for a multidigit m-graph now require time on the order of $\mathcal{O}((\lg l^m)^2)$ because each arithmetic operation now happens on numbers the size of l^m rather than l. The number of arithmetic operations required for a block cypher, though, is now only $\frac{1}{m}$ as many as for a character cypher on a message of the same length. The time required to use a multi-digit m-graph cypher is $\mathcal{O}(\frac{(\lg l^m)^2}{m}) = \mathcal{O}(m(\lg l)^2)$. With $\mathcal{O}(l^{2m})$ possible keys, a brute force attack against a multi-digit, affine, m-graph cypher will take $\mathcal{O}(ml^{2m}(\lg l)^2)$ time. This encryption scheme has a much better break-to-utilize time ratio than affine character cyphers.

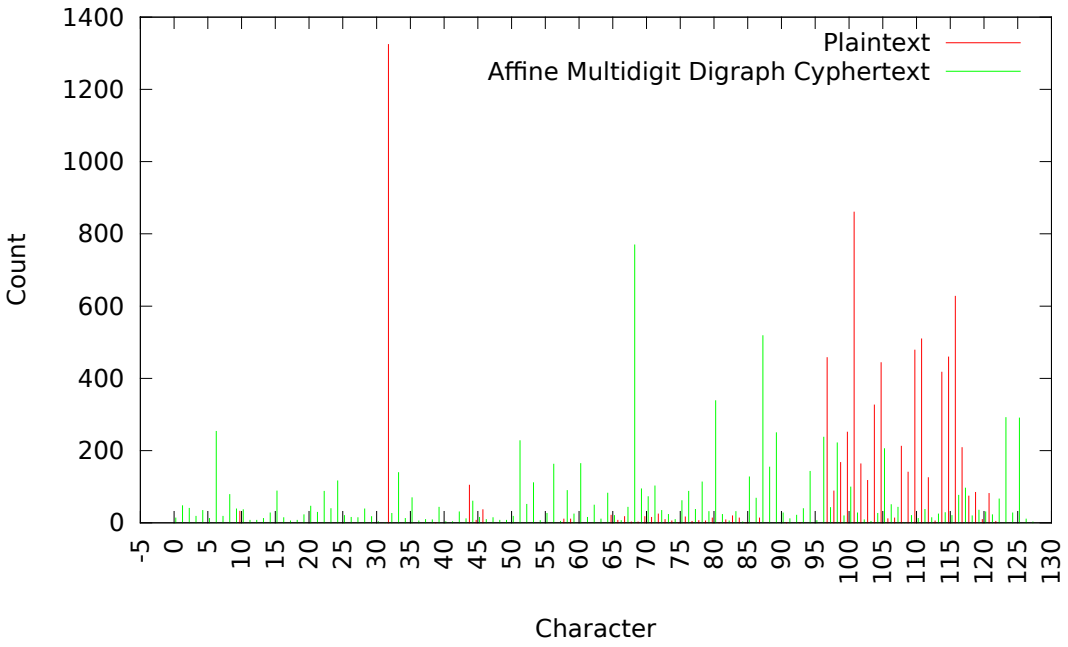

Figure 3.2: Character distributions of plaintext and an example multidigit, digraph, affine cyphertext Declaration of Independence

Once again, the Declaration of Independence was encyphered. This time with a multi-

digit, digraph, affine transformation. The key used was $(8567, 612)$. The text of the resulting cyphertext is included in Appendix A. The character distribution histogram of the cyphertext is graphed in Figure 3.2. The distribution of the original plaintext is included in the figure for reference. Clearly, the cyphertext distribution shows a more even spread in the character counts. This cyphertext has a character count variance of approximately 10100, as opposed to the plaintext's of approximately 31700. This smoother spread in the character distribution indicates that a statistical analysis will require more than merely counting the characters to be meaningful.

Notice that plaintext character 32 (space) has over 1300 occurrences, but no cyphertext character has so many. This is because not every space in the plaintext was mapped to the same character in the cyphertext. However, as previously mentioned, since the least significant digit of the result of an arithmetic computation is only dependent on the least significant digits of the operands, any space that occurred as the second character in a digraph was mapped to the same character. Only character 68 (D) of the cyphertext has over 750 occurrences. Clearly, any D in the cyphertext that occurs as the second character of a digraph is the image of a space in the plaintext.

Rather than counting characters and guessing a pair of mappings based on known characters in the plaintext alphabet, a statistical analysis on the cyphertext requires a count of the digraphs. This count can then be compared to a list of known common digraphs in the plaintext alphabet (such at 'th', 'he', and 'ng' in English) to generate probable guesses for pair mappings to solve for the key algebraically. The process is the same as for the single character cyphers, but requires more data handling and computations. For this digraph example, a histogram of digraph

frequencies could be generated for the two texts; this histogram would resemble Figure 2.1 in the sense that the bars of the cyphertext histogram would look merely like rearrangements of the bars of the plaintext histogram, except that there would now be $128^2 = 16384$ bars instead of 128. A presentation of this histogram here would, therefore, be impractical.

3.2 Vector m-graphs

Rather than considering each component character of an m-graph as a digit, it could also be considered as a vector component. Rather than a mapping into $\mathbb{Z}_{l^m}$, a mapping into ${\mathbb{Z}_l}^m$, or any R^m, is used. However, the vectors of R^m do not form a ring, for they may only be added, not multiplied. An affine transformation will require a matrix multiplication by an invertible element of $M_m(R)$. It will have the form

$$f(p) = Ap + b,$$

where $A \in M_m(R)^*$ and $b \in R^m$. The benefit thereof, is that the image of every component of each vector, p, will be determined by every other component of the vector (for a reasonably well chosen value of A).

The multidigit m-graph method only lends itself well to rings of the form $\mathbb{Z}_l$, but finite-length vectors and the matrices by which they may be multiplied can be naturally formed from any ring. And the natural (left) multiplication of elements of R^m by elements of $M_m(R)$ show R^m to be a (left) $M_m(R)$-module.[1]

Example 3.3. Encypher the message

[1] See Grove, p125 [1], for a discussion of modules.

RUBBER BABY BUGGY BUMPERS

using the standard 26-letter alphabet from above with a vector affine transformation on digraphs using the encryption key

$$\left(\begin{bmatrix} 3 & 12 \\ 15 & 7 \end{bmatrix}, \begin{bmatrix} 22 \\ 2 \end{bmatrix} \right).$$

First write the code in blocks of two.

RU BB ER BA BY BU GG YB UM PE RS

Convert each digraph to an element of ${\mathbb{Z}_{26}}^2$ with the aid of Table 2.2.

$$\begin{bmatrix} 17 \\ 20 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 4 \\ 17 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 24 \end{bmatrix} \begin{bmatrix} 1 \\ 20 \end{bmatrix} \begin{bmatrix} 6 \\ 6 \end{bmatrix} \begin{bmatrix} 24 \\ 1 \end{bmatrix} \begin{bmatrix} 20 \\ 12 \end{bmatrix} \begin{bmatrix} 15 \\ 4 \end{bmatrix} \begin{bmatrix} 17 \\ 18 \end{bmatrix}$$

Apply the encyphering transformation $f(p) = \begin{bmatrix} 3 & 12 \\ 15 & 7 \end{bmatrix} p + \begin{bmatrix} 22 \\ 2 \end{bmatrix}$.

$$\begin{bmatrix} 1 \\ 7 \end{bmatrix} \begin{bmatrix} 11 \\ 24 \end{bmatrix} \begin{bmatrix} 14 \\ 7 \end{bmatrix} \begin{bmatrix} 25 \\ 17 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \end{bmatrix} \begin{bmatrix} 8 \\ 4 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \end{bmatrix} \begin{bmatrix} 18 \\ 22 \end{bmatrix} \begin{bmatrix} 11 \\ 21 \end{bmatrix} \begin{bmatrix} 3 \\ 19 \end{bmatrix}$$

Convert back to digraphs in the 26-letter alphabet.

BH LY OH ZR BD FB IE CF SW LV DT

Arrange the cyphertext in blocks of five for easy human reading.

BHLYO HZRBD FBIEC FSWLV DT

After Example 3.2, a comparison was made between the variances of the character distributions of the plaintext and the cyphertext to demonstrate that block cyphers yield a cyphertext

with a more spread out character distribution than single character cyphers. A character distribution count table for Example 3.3 is shown in Table 3.2. It was further demonstrated that the multidigit block cyphers are not able to spread out the distribution of the characters in the least significant positions of their respective blocks. Vector cyphers are able to spread out the character distributions of all positions in the block. Therefore, the vector cyphers should tend to produce cyphertext whose character distributions have an even lesser variance. To illustrate this, a calculation of the variance in the character counts of the cyphertext from Example 3.3 results in a variance of about 0.669, less than that for either the plaintext or the multidigit cyphertext.

Table 3.2: Character distributions of Example 3.3.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
plaintext	1	6	0	0	2	0	2	0	0	0	0	0	1	0	0	1	0	3	1	0	3	0	0	0	2	0
cyphertext	0	3	1	2	1	2	0	2	1	0	0	2	0	0	1	0	0	1	1	1	0	1	1	0	1	1

With $|M_m(R)^*| = \mathcal{O}(l^{m^2})$ (for an l-letter alphabet) choices for A and $|R^m| = \mathcal{O}(l^m)$ choices for b, there are $\mathcal{O}(l^{m^2+m})$ possible encryption keys. Each affine transformation requires m^2 multiplications in R. Each affine transformation will require $\mathcal{O}(m^2(\lg l)^2)$ time and again only $\frac{1}{m}$ as many are required, so the total time to use a vector affine cypher is $\mathcal{O}(m(\lg l)^2)$, the same as for the multidigit cypher. A brute-force attack will, therefore, require $\mathcal{O}(ml^{m^2+m}(\lg l)^2)$ time.

The Declaration of Independence was encyphered twice with vector block cyphers, once with digraphs and once with 4-graphs. Both cyphertexts are included in Appendix A. The vector

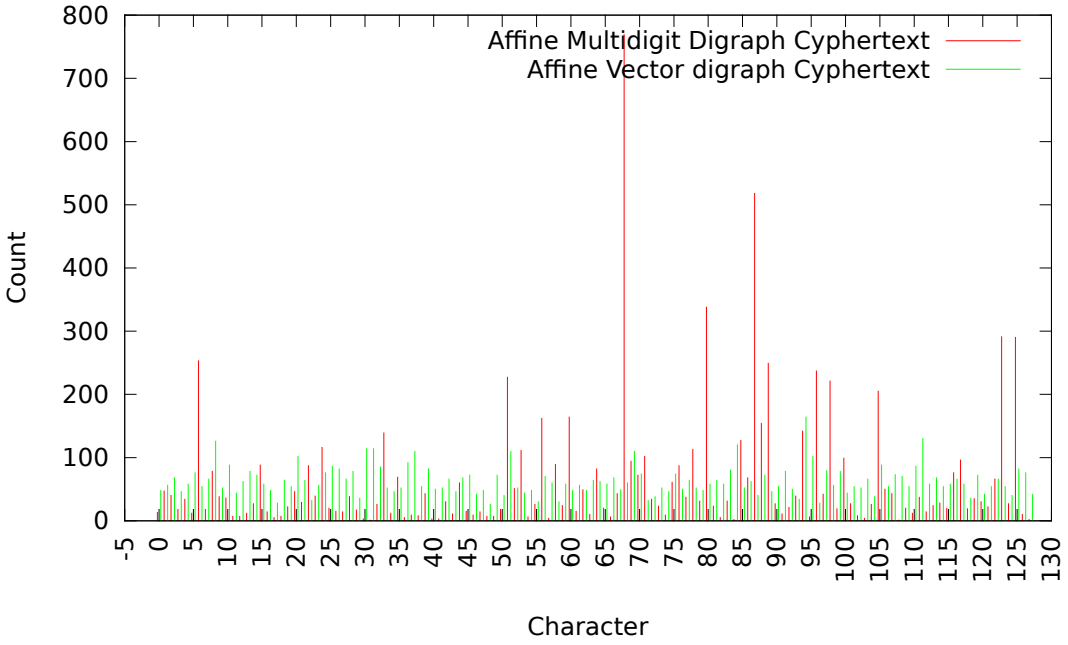

Figure 3.3: Character distribution of example vector digraph cyphertext of the Declaration of Independence

digraph encyphering used the key

$$\left(\begin{bmatrix} 95 & 5 \\ 97 & 58 \end{bmatrix}, \begin{bmatrix} 43 \\ 99 \end{bmatrix} \right),$$

and the vector 4-graph encyphering used the key

$$\left(\begin{bmatrix} 37 & 68 & 26 & 95 \\ 16 & 103 & 100 & 89 \\ 122 & 33 & 17 & 51 \\ 55 & 42 & 82 & 24 \end{bmatrix}, \begin{bmatrix} 89 \\ 92 \\ 59 \\ 92 \end{bmatrix} \right).$$

Figure 3.3 shows the resulting character distribution of the vector, digraph encyphering with the distribution for the multidigit, digraph encyphering included for reference. Notice that the vector encyphering produced cyphertext with a much smoother character distributions. Figure 3.4 shows the distribution of the 4-graph cyphertext with the distribution of the digraph cyphertext for reference. The character count variances for the digraph and 4-graph cyphertexts are approximately 515 and 261 respectively, significantly less than the 10100 of the multidigit digraph cyphertext example. A visual inspection of the cyphertexts show a smoother look to the paragraph breaks with the vector cyphers than the previous ones, agreeing with the numerical measurement.

3.3 Vigenère Cyphers - An Historical Note

The Vigenère cypher was the first historical attempt at a block cypher, and was used successfully for several centuries until cryptanalysis caught up with it.[2] In practice, it was an extension of a

[2] Koblitz, 66. [4]

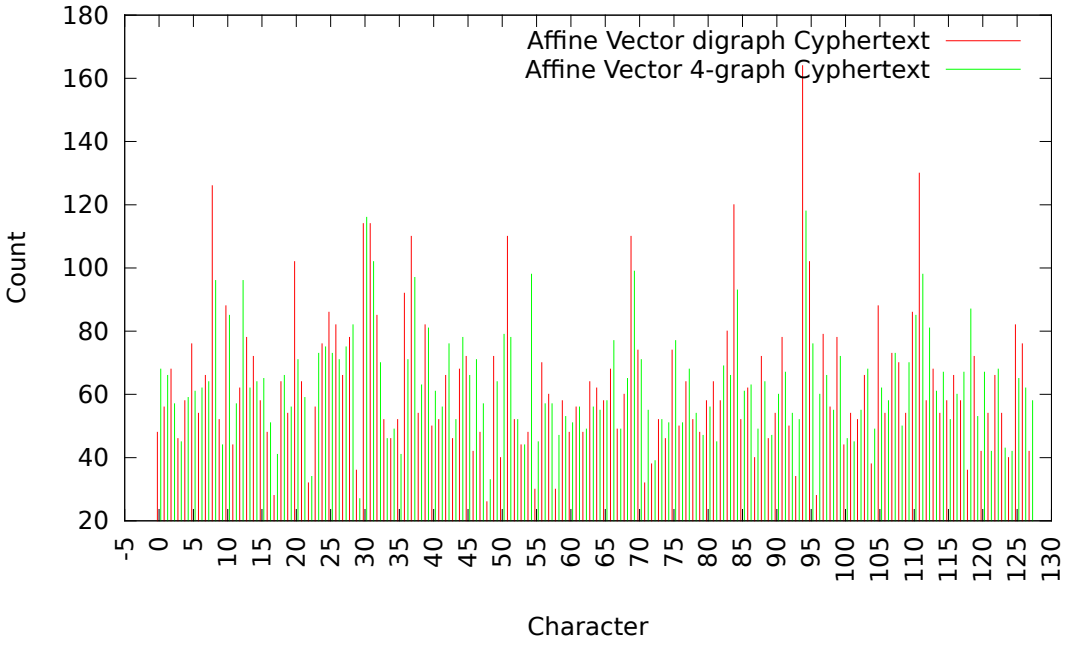

Figure 3.4: Character distribution of example vector 4-graph cyphertext of the Declaration of Independence

single character shift cypher to a rotating key sequence. The key sequence was usually associated with an easily remembered key word. The first letter of the word was the shift key for the first letter. The second letter of the word was the shift key of the second letter. When all the letters of the key word were used, the first would be used again in a repeating cycle.

Example 3.4. If the key word being used in a Vigenère cypher were "Petunia," then the first letter's key would be P, the second letter's key would be E, ... the seventh letter's key would be A, the eighth letter's key would be P again. Suppose we wanted to encypher the message "The king has a mistress." with a Vigenère cypher using the keyword "Petunia." We could write our plaintext on one line, repeat the word 'petunia' under it, and cypher one letter at a time as so:

THEKI NGHAS AMIST RESS

PETUN IAPET UNIAP ETUN

ELXGV VGWEL UZQSE VXMF

The Vigenère cypher could be considered a block cypher. However, rather than the block being encyphered together, each character of the block is encyphered separately, with a shift cypher. This still gives the effect of smoothing the character distribution of the cyphertext, thus prohibiting a direct statistical analysis of the entire character distribution. But once the block size is known, the cyphertext can be fractured according to position in the block, and a simple statistical analysis on the character distribution of each block position will lead to the discovery of each respective key.

The Vigenère cypher may be classified as a special case of the vector m-graph transformation where the multiplicative portion of the key, A, is the identity matrix. The keyword determines

the entries of the shift portion, b. For example, a keyword of "Petunia" would represent a cypher key of $b = [15\ 4\ 19\ 20\ 13\ 8\ 0]^T$.

3.4 Matrix m-graphs

3.4.1 Single-sided Affine Transformations

Rather than mapping blocks of m characters into vectors of length m, they could also be mapped into matrices of size $j \times k$ where $m = jk$. If the encyphering transformation is of the form $f(P) = AP + B$ then A should be an element of $M_j(R)$ and b an element of $R^{j\times k}$. This type of scheme, however, will be little more secure than a j-graph vector cypher. When the block of size n is broken into k sub-blocks of size j and those arranged as the k columns of the $j \times k$ matrix P, what gets created is a vector parallel to the Vigenère cypher: Each j-graph vector sub-block gets encyphered with the key (A, b_i) where b_i cycles through the columns of B.

Example 3.5. Given a block of 9 characters mapped to the 3×3 matrix

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} \\ p_{2,1} & p_{2,2} & p_{2,3} \\ p_{3,1} & p_{3,2} & p_{3,3} \end{bmatrix},$$

encypher the block with the key

$$\left(A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}, B = \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix} \right).$$

The resulting cyphertext block will be

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix} = AP + B$$

$$= \begin{bmatrix} a_{1,1}p_{1,1}+a_{1,2}p_{2,1}+a_{1,3}p_{3,1}+b_{1,1} & a_{1,1}p_{1,2}+a_{1,2}p_{2,2}+a_{1,3}p_{3,2}+b_{1,2} & a_{1,1}p_{1,3}+a_{1,2}p_{2,3}+a_{1,3}p_{3,3}+b_{1,3} \\ a_{2,1}p_{1,1}+a_{2,2}p_{2,1}+a_{2,3}p_{3,1}+b_{2,1} & a_{2,1}p_{1,2}+a_{2,2}p_{2,2}+a_{2,3}p_{3,2}+b_{2,2} & a_{2,1}p_{1,3}+a_{2,2}p_{2,3}+a_{2,3}p_{3,3}+b_{2,3} \\ a_{3,1}p_{1,1}+a_{3,2}p_{2,1}+a_{3,3}p_{3,1}+b_{3,1} & a_{3,1}p_{1,2}+a_{3,2}p_{2,2}+a_{3,3}p_{3,2}+b_{3,2} & a_{3,1}p_{1,3}+a_{3,2}p_{2,3}+a_{3,3}p_{3,3}+b_{3,3} \end{bmatrix}.$$

The extra flexibility of choosing an arrangement of the characters into the $j \times k$ matrix merely adds a permutation layer. This layer does not affect the statistical complexity.

If the arrangement of the m characters were not known, then the transformation could be modeled as an m-graph vector transformation. The m-graph vector model of the same transformation would line all m entries of the block in a vertical vector. If the length-j sub-blocks of the m-graph vector do not correspond exactly with columns of the original $j \times k$ matrix P, then that difference is a simple $m \times m$ permutation matrix. This same permutation matrix would also be the difference between the original $j \times k$ matrix B and the models length-m vector B'.

The vector transformation model's A' will be an $m \times m$ matrix. Since each element of C was originally determined by exactly one row of A and j elements of P, then each row of A' will have $m - j$ zero elements and the other j will be an arrangement of the elements from one row of A. Since each row of A originally contributed to exactly k elements of C, then each row of A will be represented by exactly k rows of A'. Each of A's j rows will populate k rows of A', accounting for all $m = jk$ rows of A'.

Only kj^2 entries of A are (likely) non-zero, and those contain only j^2 (permittedly) unique values, each occurring k times. Furthermore, the kj^2 non-zero entries will be distributed equally over the jk rows, j in each row, with none of the j^2 distinct entries occurring twice on the same row. Note that if the original $j \times j$ A has multiple entries of the same value, then that value will occur the same number of times as often in the $m \times m$ A'.

Example 3.6. Transform the 3×3 matrix transformation into a 9-graph vector transformation and verify that the new 9×9 matrix in the vector transformation will also be invertible if the original 3×3 A is.

The length-9 vectors P' and B' will be

$$P' = \begin{bmatrix} p_{1,1} & p_{2,1} & p_{3,1} & p_{1,2} & p_{2,2} & p_{3,2} & p_{1,3} & p_{2,3} & p_{3,3} \end{bmatrix}^T,$$

and

$$B' = \begin{bmatrix} b_{1,1} & b_{2,1} & b_{3,1} & b_{1,2} & b_{2,2} & b_{3,2} & b_{1,3} & b_{2,3} & b_{3,3} \end{bmatrix}^T.$$

To compute a properly corresponding C' of

$$C' = \begin{bmatrix} c_{1,1} & c_{2,1} & c_{3,1} & c_{1,2} & c_{2,2} & c_{3,2} & c_{1,3} & c_{2,3} & c_{3,3} \end{bmatrix}^T$$

$$= \begin{bmatrix} a_{1,1}p_{1,1} + a_{1,2}p_{2,1} + a_{1,3}p_{3,1} + b_{1,1} \\ a_{2,1}p_{1,1} + a_{2,2}p_{2,1} + a_{2,3}p_{3,1} + b_{2,1} \\ a_{3,1}p_{1,1} + a_{3,2}p_{2,1} + a_{3,3}p_{3,1} + b_{3,1} \\ a_{1,1}p_{1,2} + a_{1,2}p_{2,2} + a_{1,3}p_{3,2} + b_{1,2} \\ a_{2,1}p_{1,2} + a_{2,2}p_{2,2} + a_{2,3}p_{3,2} + b_{2,2} \\ a_{3,1}p_{1,2} + a_{3,2}p_{2,2} + a_{3,3}p_{3,2} + b_{3,2} \\ a_{1,1}p_{1,3} + a_{1,2}p_{2,3} + a_{1,3}p_{3,3} + b_{1,3} \\ a_{2,1}p_{1,3} + a_{2,2}p_{2,3} + a_{2,3}p_{3,3} + b_{2,3} \\ a_{3,1}p_{1,3} + a_{3,2}p_{2,3} + a_{3,3}p_{3,3} + b_{3,3} \end{bmatrix},$$

requires the use of

$$A' = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{1,1} & a_{1,2} & a_{1,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{2,1} & a_{2,2} & a_{2,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{3,1} & a_{3,2} & a_{3,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{2,1} & a_{2,2} & a_{2,3} \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}.$$

Or, more concisely, starting with

$$P = \left[\begin{array}{c|c|c} p_1 & p_2 & p_3 \end{array}\right]$$

and similarly

$$B = \left[\begin{array}{c|c|c} b_1 & b_2 & b_3 \end{array}\right]$$

and a

$$C = \left[\begin{array}{c|c|c} c_1 & c_2 & c_3 \end{array}\right]$$

such that

$$C = AP + B = \left[\begin{array}{c|c|c} Ap_1 + b_1 & Ap_2 + b_2 & Ap_3 + b_3 \end{array}\right],$$

then if one wants to restructure P as

$$P' = \left[\begin{array}{c} p_1 \\ \hline p_2 \\ \hline p_3 \end{array}\right],$$

and similarly B as

$$B' = \left[\begin{array}{c} b_1 \\ \hline b_2 \\ \hline b_3 \end{array}\right],$$

and C as

$$C' = \left[\begin{array}{c} c_1 \\ \hline c_2 \\ \hline c_3 \end{array}\right],$$

then A must be restructured as

$$A' = \begin{bmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & A \end{bmatrix}.$$

The reader may easily verify that if $C = AP + B$, then $C' = A'P' + B'$.

Lastly, $|A'| = |A|^3$, so A' will be invertible if, and only if, A is.

Example 3.6 demonstrates what could be called a standard mapping of a character block into a matrix, where each column, from left to right, is filled from top to bottom. Any other arrangement can be converted into a vector block, like in Example 3.6, and if the coefficient matrix A' and constant shift vector B are multiplied by appropriate permutation matrices, then the message vectors P' and C' can be equated with message blocks in the standard, same-order sequence manner.

3.4.2 Double-sided Affine Transformations

Rather than a transformation of the form $f(P) = AP + B$, a matrix transformation could also use a transformation of the form $f(P) = APB + C$, where if $P \in R^{m\times n}$, then $A \in GL_m(R)$, $B \in GL_n(R)$, and $C \in R^{m\times n}$. The key space is now $GL_m(R) \times GL_n(R) \times R^{m\times n}$, the size of which is $\mathcal{O}(|R|^{m^2+n^2+mn})$.

This type of transformation is, also, able to be converted to one of a simpler form like the single-sided transformation above was. While, in general, it is not possible to convert a right-hand multiplication like PB into a left-hand of the form $B'P$ with a fixed B' and variable P, the

double-sided $m \times n$ matrix affine transformation can be converted into a single-sided length-mn vector transformation just like the single-sided matrix transformation above.

For this, we'll introduce some extra notation. First, the act of vectorizing a matrix, M, by stacking its columns on top of each other will be notated with the vec operator. Thus if M is an $n \times m$ matrix, then $\operatorname{vec} M$ is an mn-length column vector. Additionally, the column of a matrix shall be denoted with a single subscript in the following.

A new matrix product, known as the **Kronecker Product**, shall be denoted with the $\otimes$ operator.

Definition 3.7 (Kronecker Product)**.** If A is a $k \times l$ matrix and B is an $m \times n$ matrix, then the Kronecker product $A \otimes B$ is a $km \times ln$ matrix formed by left distributing B to each element of A.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,l} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k,1} & a_{k,2} & \cdots & a_{k,l} \end{bmatrix} \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \cdots & a_{1,l}B \\ a_{2,1}B & a_{2,2}B & \cdots & a_{2,l}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{k,1}B & a_{k,2}B & \cdots & a_{k,l}B \end{bmatrix}$$

Theorem 3.8. *A matrix transformation of the form $f(P) = APB + C$ where $A \in GL_m(R)$, $B \in GL_n(R)$, and $P, C \in R^{m \times n}$ is convertible to one of the form $f(P) = A'P' + C'$ where $A' \in GL_{mn}(R)$ and $P', C' \in R^{mn}$.*[3]

[3]The proof of this is essentially copied from Horn and Johnson, 254-255. [3]

Proof. We'll first examine the k^{th} column of the product APB.

$$
\begin{aligned}
(APB)_k = A(PB)_k = & APB_k \\
= & A \begin{bmatrix} \sum_{i=1}^{n} p_{1,i} b_{i,k} \\ \vdots \\ \sum_{i=1}^{n} p_{m,i} b_{i,k} \end{bmatrix} \\
= & A \begin{bmatrix} \sum_{i=1}^{n} b_{i,k} p_{1,i} \\ \vdots \\ \sum_{i=1}^{n} b_{i,k} p_{m,i} \end{bmatrix} \\
= & A \sum_{i=1}^{n} \begin{bmatrix} b_{i,k} p_{1,i} \\ \vdots \\ b_{i,k} p_{m,i} \end{bmatrix} \\
= & A \sum_{i=1}^{n} b_{i,k} \begin{bmatrix} p_{1,i} \\ \vdots \\ p_{m,i} \end{bmatrix} \\
= & A \left(\sum_{i=1}^{n} b_{i,k} P_i \right) \\
= & \sum_{i=1}^{n} A b_{i,k} P_i
\end{aligned}
$$

This distribution of A into the summation will eventually become a Kronecker product of a matrix with A. At this point we may reinterpret the summation as a vector inner product and

continue like so:

$$\begin{aligned}\sum_{i=1}^{n} Ab_{i,k}P_i &= \begin{bmatrix} Ab_{1,k} & Ab_{2,k} & \cdots & Ab_{n,k} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_k \end{bmatrix} \\ &= \begin{bmatrix} b_{1,k}A & b_{2,k}A & \cdots & b_{n,k}A \end{bmatrix} \operatorname{vec} P \\ &= ({B_k}^T \otimes A) \operatorname{vec} P\end{aligned}$$

Remember that this is just the k^{th} column of APB. The vectorization of APB will result in stacking these columns on top of each other.

$$\begin{aligned}\operatorname{vec} APB &= \begin{bmatrix} ({B_1}^T \otimes A) \operatorname{vec} P \\ ({B_2}^T \otimes A) \operatorname{vec} P \\ \vdots \\ ({B_n}^T \otimes A) \operatorname{vec} P \end{bmatrix} \\ &= \begin{bmatrix} ({B_1}^T \otimes A) \\ ({B_2}^T \otimes A) \\ \vdots \\ ({B_n}^T \otimes A) \end{bmatrix} [\operatorname{vec} P]\end{aligned}$$

But since $\operatorname{vec} P$ is already a vector, $[\operatorname{vec} P] = \operatorname{vec} P$.

$$\begin{aligned}
\operatorname{vec} APB &= \begin{bmatrix} (B_1{}^T \otimes A) \\ (B_2{}^T \otimes A) \\ \vdots \\ (B_n{}^T \otimes A) \end{bmatrix} \operatorname{vec} P \\
&= \left(\begin{bmatrix} B_1{}^T \\ B_2{}^T \\ \vdots \\ B_n{}^T \end{bmatrix} \otimes A \right) \operatorname{vec} P \\
&= (B^T \otimes A) \operatorname{vec} P
\end{aligned}$$

And so, not only do we know that a transformation exists to vectorize P and C, but we also know the form of the transformation. The important part is that the same vector-based analysis may be used by an attacker whether the actual transformation is a vector, single-sided matrix or double-sided matrix transformation. Note also, that the single-sided matrix vectorization of Example 3.6 is a special case of the double-sided one where the right-hand matrix is just an identity, I. □

The time involved in using a double-sided matrix transformation on a plaintext message is dominated by the matrix multiplications. There are m^2n multiplications in R when multiplying by A and mn^2 multiplications in R when multiplying by B. If $R = \mathbb{Z}_l$, then each multiplication requires $\mathcal{O}((\lg l)^2)$ time, but there are only $\frac{1}{mn}$ of them. The full transformation will therefore

require $\mathcal{O}((m+n)(\lg l)^2)$ time. With a keyspace of $\mathcal{O}(l^{m^2+n^2+mn})$, a brute-force attack would require $\mathcal{O}(l^{m^2+n^2+mn}(m+n)(\lg l)^2)$ time.

Compare this with the time requirements for an mn vector transformation: $\mathcal{O}(mn(\lg l)^2)$ transformation time and $\mathcal{O}(l^{(mn)^2+mn})$-size keyspace. The vector transformation has a usage requirement $\frac{mn}{m+n}$ times harder, and a keyspace $\frac{l^{m^2n^2+mn}}{l^{m^2+n^2+mn}} = l^{m^2n^2-m^2-n^2}$ times larger, the breakage requirement is $\frac{mn}{m+n} \cdot l^{m^2n^2-m^2-n^2}$ times harder. A vector transformation with the same block size seems favorable to a matrix transformation.

3.5 Combining m-graph Methods

Different m-graph methods may also be used together. The most simple combination would be to use multiple digit m_1-graphs as the base ring elements in a vector or matrix m_2-graph. This is especially true with computers.

Computers use the byte (8-digit binary numbers) as their base unit of storage. This produces a natural 256-letter alphabet with a natural mapping to $\mathbb{Z}_{256}$. However, the base unit (or perhaps more properly called the optimal unit) for mathematical operations may be 2, 4, or 8 bytes, depending on architecture, and is called a word. On a computer that can natively (in hardware) handle 32-bit operands (4 bytes), it would be very convenient to use multiple digit 4-graphs mapped to $\mathbb{Z}_{2^{32}}$. Even more convenient, is the fact that arithmetic operations will naturally wrap around from $2^{32}-1$ to 0, easing the requirements for a mod operation to always follow.[4] A 32-bit arithmetic processor will naturally work with integers mod 2^{32}.

[4] Computer programmers refer to this "feature" as **overflow** when modular computations are not desired.

Once the characters of a plaintext message are grouped into words of length 4 (for example), $\mathbb{Z}_{2^{32}}$ may then be used as the base ring to form m-graphs (of words now, not characters) into vectors or matrices.

Section 2.5.2 discussed the composition of affine encypherings and demonstrated that it was only beneficial when a different ring was used for each encyphering. With all these different m-graph techniques, it is not hard to find two or more different rings in which to work. Even using the same m-graph technique with different sizes of blocks will suffice. And while the net result will be a block cypher with a block size of the lcm of each of the component block cyphers, the resultant cypher may not be representable as an affine cypher, and the associated keyspace would then not form a group structure.

Example 3.9. Take the notes of *Twinkle, Twinkle, Little Star* (in the key of C-Major), written in the seven-character alphabet $X = \{A, \ldots G\}$ (neglecting note lengths):

C C G G A A G F F E E D D C G G F F E E D G G D D E E D C C G G A A G F F E E D D C.

Let the notes be represented by the ring $\mathbb{Z}_7$ in alphabetical order with A = 0 and G = 6. First encypher the sequence as multidigit 2-graphs (in the ring $\mathbb{Z}_{7^2}$ with the key $(45, 8)$. First collect the message in groups of two and then convert the groups of two to elements of $\mathbb{Z}_{49}$ considering the first element of each pair as the 7's digit.

CC GG AA GF FE ED DC GG FF EE DG GF FE ED CC GG AA GF FE ED DC

16 48 0 47 39 31 23 48 40 32 27 47 39 31 16 47 0 47 39 31 23

Next apply the affine transformation $c = 45p + 8$ then convert the resulting elements of $\mathbb{Z}_{49}$ back to character pairs.

42 12 8 16 48 31 14 12 44 27 47 16 48 31 42 12 8 16 48 31 14

GA BF BB CC GG ED CA BF GC DG GF CC GG ED GA BF BB CC GG ED CA

Next collect the characters into groups of three and convert them to elements of $\mathbb{Z}_{7^3}$.

GAB FBB CCG GED CAB FGC DGG FCC GGE DGA BFB BCC GGE DCA

295 253 118 325 99 289 495 261 340 189 85 65 340 161

Then encypher them with the key $(177, 140)$ in $\mathbb{Z}_{7^3}$ and convert back to tri-graphs.

219 331 103 41 170 186 12 32 295 322 93 326 295 168

EDC GFC CAF AFG DDC DFE ABF AEE GAB GEA BGC GEE GAB DDA

These two encypherings in series have the effect of a 6-graph encyphering. However it is certainly not a multidigit, 6-graph affine cypher. As 6-graphs, the first two blocks of the final cyphertext would correspond to the values 75,448 and 35,370 in $\mathbb{Z}_{7^6}$ respectively. The first two blocks of the original plaintext correspond to the values 40,768 and 114,789 in $\mathbb{Z}_{7^6}$ respectively. Using these two blocks, a known plaintext attack yields the system of equations

$$75,448 = 40,768a + b$$

$$35,370 = 114,789a + b$$

Using a computerized algebra software such as Sage[5] or Maxima[6], because working by hand in a ring this big is tedious, yields a solution of the key $(17912, 86375)$. The next two blocks of cyphertext correspond to the values 58,496 and 4148 respectively and the next two blocks of plaintext correspond to the values 57,615 and 78,202 respectively. Setting up a system of equations and solving for the key again gives 7 possible keys that work with this pair: $\{(6495, 90540)$, $(23302, 6505)$, $(40109, 40119)$, $(56916, 73733)$, $(73723, 107347)$, $(90530, 23312)$, $(107337, 56926)\}$ (like the system in Example 2.17 this one need not have a single unique solution because it is only over a ring and not a field), none of which match the key that works for the first pair of 6-graphs.

[5] `http://www.sagemath.org/`

[6] `http://maxima.sourceforge.net/`

Chapter 4

Exponential Cyphers

4.1 Introduction

The encyphering systems of Chapters 2 and 3 belong to the class of symmetric cryptosystems. The cryptosystems of this chapter are **asymmetric**.

Definition 4.1 (Asymmetric cryptosystem)**.** A cryptographic system where knowledge of one of the pair of keys is "easily" computable from knowledge of the second, but knowledge of the second is computationally "difficult" from knowledge of the first.

The affine transformations of Chapters 2 and 3 all have relatively easily computed inverses, and the encyphering transformations themselves are just as easily computed from their inverses. With asymmetric cryptosystems, the key from which the other can be "easily" computed is considered the decyphering key and the key from which it would be "difficult" to compute the other is considered the encyphering key. In practice, a participant in an asymmetric cryptosystem

makes his encyphering key publicly available, and keeps his decyphering key private. The two keys, therefore, are commonly referred to as the public key and the private key. And by virtue of this public key, asymmetric cryptosystems are commonly called **public-key** cryptosystems.

The security of an interaction between two parties using a symmetric cryptosystem is dependent upon the pre-existing knowledge of the encryption keys by both parties and the secrecy of those same keys from all other (or at least all untrusted) parties. For both interested parties to already know the secret key, it would have to have been agreed upon at an earlier meeting. For no third parties to know the key, the meeting where it was agreed upon must have been absolutely secure and the key must not have been cracked by an eavesdropper. For example, if Alice and Bob want to pass notes to each other in English class, but because they sit in opposite corners of the room and don't want anyone else to be able to read their notes decide to encrypt their messages with a symmetric encryption, they must agree on a key sometime before class, somehow so that no one else in class knows the key.

A third party would only need to crack the key once, and discover either the encyphering or the decyphering key. Once one is known the other may be easily computed and the third party will have covertly gained full privileges to the secret conversation. With this knowledge, the third party could continue eavesdropping (now in knowledge of the correspondence), decrypt saved messages that had been encyphered with the key before he cracked it, and/or impersonate one or both of the members of the secret conversation. The third party is now capable of sending encrypted messages to the legitimate participants (alleging to be the other legitimate member), and could fully hijack the conversation if he is capable of intercepting the legitimate messages and blocking their delivery.

With a symmetric cryptosystem, a group of parties may all share one key for secret messages amongst themselves, but if two parties of the group wish to converse without the others, they must share a separate key for just themselves. If every distinct pair of parties sets up a symmetric key, then there will be as many keys as handshakes ($\frac{n(n-1)}{2}$). Furthermore, if every possible set of 3 parties, and every possible set of 4 parties, and every possible set of 5 parties, etc... want to set up a symmetric key to be able to communicate to the exclusion of the rest of the group, then the number of required keys will be close to 2^n.

In asymmetric cryptography, when one party desires to send a message to a second, he transforms the message with his private key, which only he knows, then transforms that result with the intended recipients public key, which had been made public knowledge. Upon receipt, the recipient transforms the cyphertext with his private key, which only he knows, inverting the senders second transformation, then transforms that result with the senders public key, which had been made public knowledge, inverting the senders first transformation. Suppose party A's public key performs the transformation f_A and private key performs the transformation ${f_A}^{-1}$, and party B's public key performs the transformation f_B and private key performs the transformation ${f_B}^{-1}$. When party A wants to send a message to party B, party A encyphers the plaintext message with

$$C = f_B({f_A}^{-1}(P)).$$

Upon receipt, party B decyphers the cyphertext with

$$f_A({f_B}^{-1}(C)) = f_A({f_B}^{-1}(f_B({f_A}^{-1}(P)))) = f_A({f_A}^{-1}(P)) = P.$$

For example, if Alice and Bob decide to use an asymmetric cyptosystem for passing notes

in English class, they may agree in front of everyone (except perhaps the teacher) to do so, each go to his/her seat and generate a key pair, then share their public keys publicly. When Alice uses f_B in encyphering a message to Bob she knows that only Bob will be able to invert (decypher) it, and when Bob uses f_A in decyphering the message he knows that only Alice could have encyphered it. The messages are both secure (only the recipient will get them) and authenticated (the sender is verified to be as claimed).

With an asymmetric cryptosystem, only one key (public-private pair) is needed for each individual party, and any pair of parties may then communicate to the exclusion of everyone else. However, sharing messages among more than two parties requires a separate encyphering for each recipient. In practice, nearly all encrypted conversations involve only two parties anyway.

In general, an encyphering transformation is considered practical if encyphering with f and decyphering with f^{-1} take time no longer than on the order of a polynomial in $\log n$ where n is the size of the ring used. The transformation is considered asymmetric if decyphering a message knowing only f (cracking the key) requires time at least on the order or a polynomial in n.[1] Such a transformation, f, is called a **trap-door function**, because a message could be easily encyphered (fall through the trap door) by anyone with Alice's public key, but it could not be easily decyphered (escape back out through the trap door) except by Alice who has the private key.

Definition 4.2 (Trap-door function)**.** A trap-door function is an invertible function $f : X \to Y$ such that

- $f(x)$ is "easy" to evaluate for any $x \in X$ and

[1] Koblitz, 88. [4]

- $y = f(x)$ is "difficult" to solve for x given a $y \in Y$.

Such trap-door functions are best found in historically difficult problems. The two historically difficult problems whose derived cryptosystems will be described here are that of factoring an integer into prime factors and evaluating the discrete logarithm.

4.2 Prime Factorization

Given two or more prime integers, it is fairly simple to compute their product. However, given a composite integer, it can be very difficult to compute its prime factorization. The difficulty of factoring an integer into it's prime factors is the basis for the security of the RSA public-key cryptosystem.

The process of generating a key pair for RSA encryption proceeds as follows: First, two prime numbers are chosen at random, p and q. Their product is calculated, $n = pq$. The size of the group of units of $\mathbb{Z}_n$ is calculated, $\phi(n) = (p-1)(q-1)$. An invertible element of $\mathbb{Z}_{\phi(n)}$ is chosen at random, e. The inverse of e in $\mathbb{Z}_{\phi(n)}$ is calculated, d. The public key is $K = (e, n)$ and the private key is $K^{-1} = (d, n)$. Encyphering involves mapping m-graphs into $\mathbb{Z}_n$ and transforming them with the function

$$C = f(P) = P^e \mod n.$$

Decyphering is performed with the inverse function

$$P = f^{-1}(C) = C^d \mod n = P^{ed} \mod n = P^1 \mod n.$$

These functions are inverses because e and d were chosen so that they would be. For any $x \in \mathbb{Z}_n$, unit or not, the multiplicative, cyclic group or semi-group, respectively, $\langle x \rangle$ will have an order dividing $\phi(n)$. Therefore, because $\phi(n)$ divides $(ed-1)$, $|\langle x \rangle|$ will also divide $(ed-1)$ and $x^{ed} = x$. As a side note, knowing the structure of $\mathbb{Z}_n = \mathbb{F}_p \times \mathbb{F}_q$, ed need only be congruent to 1 $(\text{mod } \gcd(p-1, q-1))$. It would, therefore, be advisable that when choosing one's p and q, one chose a pair whose $p-1$ and $q-1$ have the fewest common multiples possible. (As they are both odd, a common factor of 2 will be unavoidable.)

The RSA cryptosystem is dependent of the difficulty of factoring integers, because if the factorization of n were known, then the inverse of any number in $\mathbb{Z}_{\phi(n)}$ could be easily calculated with the Extended Euclidean Algorithm.

Example 4.3. Encypher and then decypher the message

```
Call me Ishmael.
```

in a 33-character alphabet by appending '"'=32 to the 32-character alphabet in Table 2.5 on page 48 with the encryption key $(3, 33)$.

Note that $33 = 3 \cdot 11$, $\phi(33) = 2 \cdot 10 = 20$ and $3^{-1} \equiv 7 \pmod{20}$.

First, convert the characters to their representatives in $\mathbb{Z}_{33}$.

```
(2, 0, 11, 11, 26, 12, 4, 26, 8, 18, 7, 12, 0, 4, 11, 28)
```

Then, raise each element to the third power in $\mathbb{Z}_{33}$.

```
(8, 0, 11, 11, 20, 12, 31, 20, 17, 24, 13, 12, 0, 31, 11, 7)
```

And convert each element of $\mathbb{Z}_{33}$ back to a character of the chosen alphabet.

```
IALLUM:RUYNMA:LH
```

Decyphering will follow the reverse steps backwards. Convert the cyphertext to elements of $\mathbb{Z}_{33}$.

```
(8, 0, 11, 11, 20, 12, 31, 20, 17, 24, 13, 12, 0, 31, 11, 7)
```

Raise each element to the seventh power.

```
(2, 0, 11, 11, 26, 12, 4, 26, 8, 18, 7, 12, 0, 4, 11, 28)
```

And convert each element back to a character to reveal the plaintext.

```
CALL ME ISHMAEL.
```

Example 4.3 is unrealistic for multiple reasons. First, the chosen modulus is too small to be secure, though this was done intentionally so that the procedure could be the focus of the example. Second, the encyphering was performed on single characters, rather than blocks. And finally, the size of the alphabet was the same as the key's modulus.

The reason it is unrealistic for the modulus and alphabet size to be the same, is that every participant should have a different modulus. Each participants modulus is the product of their secret pair of prime numbers. If any two also had the same modulus, they would know the factorization of each others modulus and be able to decypher messages intended for the other.

For participants to send each other messages, they need an agreed-upon alphabet and block size. But if the moduli of the participants are all different, then how could the alphabets and block sizes be the same? This is handled by selecting a larger alphabet and/or block size for the cyphertext messages than the plaintext messages. Each participant chooses their p and q

such that the product is between the moduli for the two text bases. If $l_{\mathcal{P}}$ and $l_{\mathcal{C}}$ are the sizes of the plaintext and cyphertext alphabets respectively and $m_{\mathcal{P}}$ and $m_{\mathcal{C}}$ are the block sizes of the plaintext and cyphertext messages respectively, then each participant chooses his p and q such that ${l_{\mathcal{P}}}^{m_{\mathcal{P}}} \leq n = pq < {l_{\mathcal{C}}}^{m_c}$. The blocks of text each get interpreted as multidigit m-graphs. The order in which transformations are applied is determined by modulus size.

Suppose Alice has a modulus of n_A and Bob has a modulus of n_B. When Alice sends Bob a message she applies the transformations in the order of increasing modulus. If $n_A < n_B$, then Alice first applies her private key (mod n_A), then applies Bob's public key (mod n_B), and sends Bob

$$C = f_B({f_A}^{-1}(P)).$$

If $n_B < n_A$, then Alice first applies Bob's public key (mod n_B), then applies her private key (mod n_A), and sends Bob

$$C = {f_A}^{-1}(f_B(P)).$$

When Bob receives the message he applies the transformations in the order of decreasing modulus. If $n_A < n_B$, then Bob first applies his private key (mod n_B), then applies Alice's public key (mod n_A) and reduces that result (mod ${l_{\mathcal{P}}}^{m_{\mathcal{P}}}$) to read

$$f_A({f_B}^{-1}(C)) = f_A({f_B}^{-1}(f_B({f_A}^{-1}(P)))) = f_A({f_A}^{-1}(P)) = P.$$

If $n_B < n_A$, then Bob first applies Alice's public key (mod n_A), then applied his private key (mod n_B) and reduces that result (mod ${l_{\mathcal{P}}}^{m_{\mathcal{P}}}$) to read

$${f_B}^{-1}(f_A(C)) = {f_B}^{-1}(f_A({f_A}^{-1}(f_B(P)))) = {f_B}^{-1}(f_B(P)) = P.$$

The flow of events will look either like

$$\mathcal{P} \to m_{\mathcal{P}}\text{-graphs} \to \mathbb{Z}_{{l_{\mathcal{P}}}^{m_{\mathcal{P}}}} \xrightarrow{{f_A}^{-1}} \mathbb{Z}_{n_A} \xrightarrow{f_B} \mathbb{Z}_{n_B} \to \mathbb{Z}_{{l_{\mathcal{C}}}^{m_{\mathcal{C}}}} \to m_{\mathcal{C}}\text{-graphs} \to \mathcal{C}$$

$$C \to m_{\mathcal{C}}\text{-graphs} \to \mathbb{Z}_{{l_{\mathcal{C}}}^{m_{\mathcal{C}}}} \to \mathbb{Z}_{n_B} \xrightarrow{{f_B}^{-1}} \mathbb{Z}_{n_A} \xrightarrow{f_A} \mathbb{Z}_{{l_{\mathcal{P}}}^{m_{\mathcal{P}}}} \to m_{\mathcal{P}}\text{-graphs} \to \mathcal{P},$$

or like

$$\mathcal{P} \to m_{\mathcal{P}}\text{-graphs} \to \mathbb{Z}_{{l_{\mathcal{P}}}^{m_{\mathcal{P}}}} \xrightarrow{f_B} \mathbb{Z}_{n_B} \xrightarrow{{f_A}^{-1}} \mathbb{Z}_{n_A} \to \mathbb{Z}_{{l_{\mathcal{C}}}^{m_{\mathcal{C}}}} \to m_{\mathcal{C}}\text{-graphs} \to \mathcal{C}$$

$$C \to m_{\mathcal{C}}\text{-graphs} \to \mathbb{Z}_{{l_{\mathcal{C}}}^{m_{\mathcal{C}}}} \to \mathbb{Z}_{n_A} \xrightarrow{f_A} \mathbb{Z}_{n_B} \xrightarrow{{f_B}^{-1}} \mathbb{Z}_{{l_{\mathcal{P}}}^{m_{\mathcal{P}}}} \to m_{\mathcal{P}}\text{-graphs} \to \mathcal{P}.$$

Example 4.4. Use the standard 26-letter alphabet from Table 2.2 for both the plaintext and cyphertext. Use 3 character blocks for plaintext and 4 character blocks for cyphertext. Use a private key of ($78701 = 101^{-1}$,$124931 = 271 \cdot 461$) and encypher the following message for a recipient with a public key of (101,122431).

```
O ROMEO O ROMEO WHEREFORE ART THOU ROMEO
```

The plaintext should first be grouped into 3-graphs and converted into elements of $\mathbb{Z}_{26^3}$.

```
ORO MEO ORO MEO WHE REF ORE ART THO URO MEO
```

```
(14:17:14, 12:4:14, 14:17:14, 12:4:14, 22:7:4, 17:4:5, 14:17:4, 0:17:19,
19:7:14, 20:17:4, 12:4:14)
```

```
(9920, 8230, 9920, 8230, 15053, 11601, 9910, 461, 13040, 13966, 8230)
```

Since the recipient's key has the smaller modulus, his will be used first. Each entry in the latest vector shall be raised to an exponent of 101 (mod 122431). Exponentiation by repeated

squaring will be demonstrated here in a $\mathbb{Z}_l$ ring on the first entry only. See Section 1.6.3 for the description of the procedure.

i	101	9920^{2^i} (mod 122431)	Running product (mod 122431)
			1
0	1100101	9920	9920
1	1100101	94307	9920
2	1100101	55116	96305
3	1100101	15484	96305
4	1100101	34358	96305
5	1100101	114893	68740
6	1100101	13460	29333

`(29333, 43110, 29333, 43110, 9902, 30040, 110028, 43397, 75323, 46988, 43110)`

Now each entry gets raised to an exponent of 78701 (mod 124931).

`(3138, 55108, 3138, 55108, 124825, 64615, 10857, 33987, 17985, 16826, 55108)`

Then convert to 4-graphs in the 26-letter alphabet.

`(0:4:16:18, 3:3:13:14, 0:4:16:18, 3:3:13:14, 7:2:16:25, 3:17:15:5, 0:16:1:15, 1:24:7:5, 1:0:15:19, 0:24:23:6, 3:3:13:14)`

`AEQS DDNO AEQS DDNO HBQZ DRPF AQBP BYHF BAPT AYXG DDNO`

According to Theorem 1.19 in Section 1.6.3, each exponentiation will require $\mathcal{O}((\lg e)(\lg n)^2)$ or $\mathcal{O}((\lg d)(\lg n)^2)$ time. Each of d and e are elements of $\mathbb{Z}_{\phi(n)}$ and thus have magnitudes of

$\mathcal{O}(\phi(n)) = \mathcal{O}(n)$. And $n = \mathcal{O}({l_\mathcal{C}}^{m_\mathcal{C}})$. If the plaintext and cyphertext use the same alphabet (it is standard that each computer byte is an element of $\mathbb{Z}_{256}$) then the subscript on the l can be dropped. And once l is fixed then $m_\mathcal{P}$ and $m_\mathcal{C}$ differ by a fixed multiple, so the subscript on the m may also be dropped. Now, each exponentiation will require $\mathcal{O}((\lg l^m)^3) = \mathcal{O}(m^3(\lg l)^3)$ time. For any specific plaintext, only $\mathcal{O}(\frac{1}{m})$ as many exponentiations will be required, thus using this exponential cypher requires $\mathcal{O}(m^2(\lg l)^3)$ time. This is one order of magnitude larger than a vector transformation, which requires $\mathcal{O}(m(\lg l)^2)$ time, of the same block size, which makes perfect sense as exponentiation is repeated multiplication.

4.3 The Discrete Logarithm

Another arithmetic operation that historically has been computationally difficult is the discrete logarithm: the evaluation of the logarithm of an element in a discrete field.

In the Real Numbers and in discrete fields, exponential functions are (relatively) easy to evaluate. The repeated squaring method keeps evaluation time down. The exponential function's inverse, however, is not so simple.

In the case of the Real Numbers, exponential functions are smooth and continuous. Real logarithms tend to be irrational numbers but can be approximated with several calculus-based methods. This ease comes from the niceness of an exponential curve, as depicted in Figure 4.1.

Exponential functions in discrete fields have a much more random appearance to them. This makes estimation of a logarithm in a discrete field "difficult." The proximity of two inputs to an exponential function appears to have little relation to the proximity of the outputs. Examine

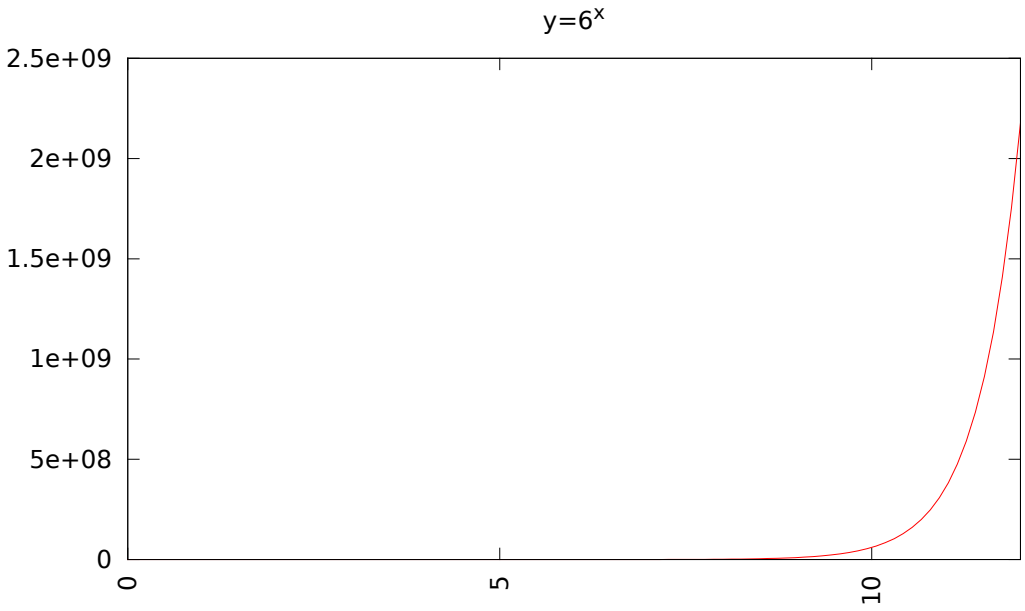

Figure 4.1: Exponential function in $\mathbb{R}$.

the function $y = 6^x$ in Figure 4.2. Notice that while 5 and 6 differ by only one, 6^5 and 6^6 differ by 10 (or 3). On the other hand, 6^3 and 6^4 differ by only one, as do 6^1 and 6^7.

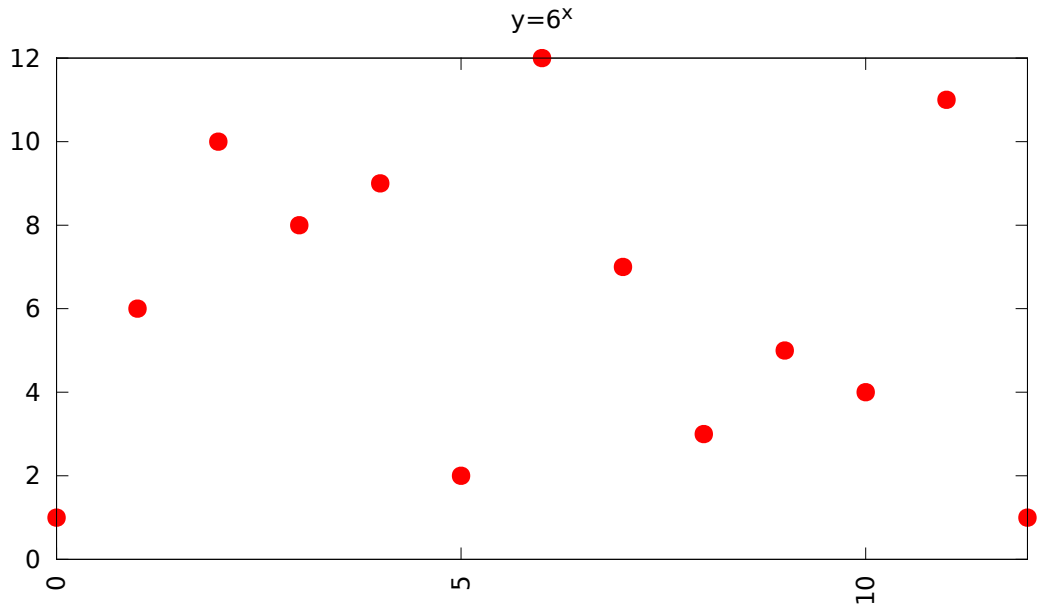

Figure 4.2: Exponential function in $\mathbb{Z}_{13}$.

This random appearance can be seen more clearly in larger fields. Figure 4.3 shows $y = 12^x$ in $\mathbb{Z}_{139}$. This exponential function clearly illustrates that estimating discrete logarithms is probably impossible, and calculating them is rather "difficult." This apparent randomness of the discrete exponential function is utilized in the generation of pseudo-random numbers.[2]

[2]Menezes, §5.5, p. 185-187. [5]

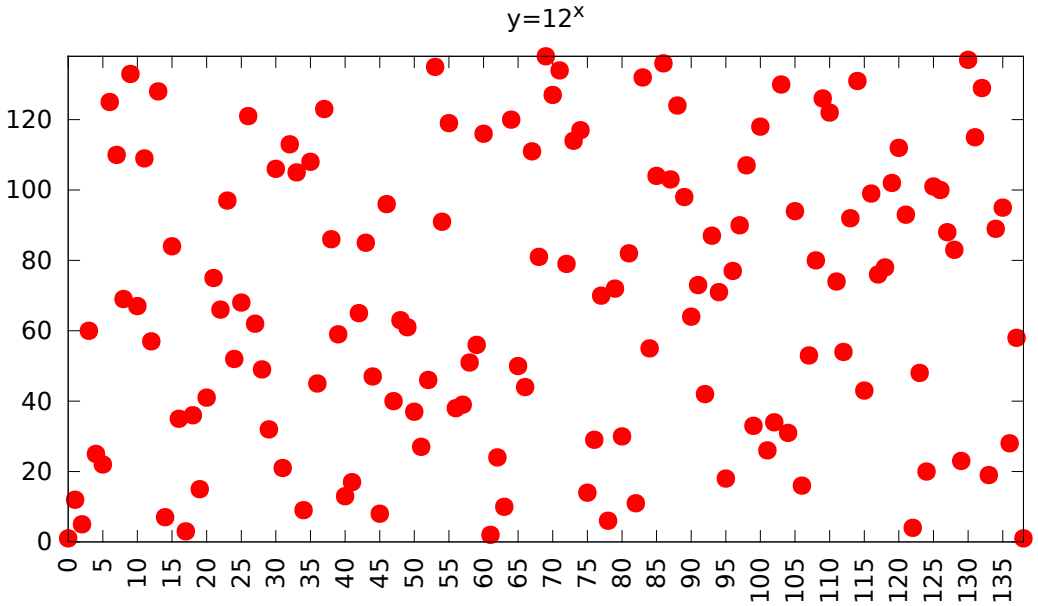

Figure 4.3: Exponential function in $\mathbb{Z}_{139}$.

4.3.1 Diffie-Hellman Key Exchange

As exponentiation is an order of magnitude more (computationally) expensive than multiplication, public-key cryptosystems using it are also significantly more expensive to use than classical, symmetric-key cryptosystems. As such, public key systems require significantly more time to use than classical systems. For this reason, public key systems are usually employed as an augmentation to a classical system, rather than as a replacement for one.

The Diffie-Hellman key exchange is a method by which two previously unacquainted parties may publicly agree to a classical cryptosystem's key, under the security of a public key system.

First the two parties agree to a discrete field and a generator, g, of a cyclic subgroup, $\langle g \rangle$, of the multiplicative group of units. Preferably, g is chosen to generate the entire group of units, so as to permit the greatest number of possible classical system keys. A mapping from the generated group to the classical systems' keyspace is agreed upon. Each party then selects a random integer, α and β respectively (these are their private keys), in the range $[0, |\langle g \rangle|)$ and calculate the respective powers of g. The parties trade their powers of g, g^α and g^β (these are their public keys). The key to be used in their classical cryptosystem communiqués is the one represented by $g^{\alpha\beta}$, which each may calculate by raising the other parties power of g to their own power. Let's illustrate this exchange with an example.

Example 4.5. Suppose Alice and Bob wish to initiate a classical cryptosystem-encoded conversation with a secret key decided by a Diffie-Hellman key exchange.

Alice and Bob have agreed to use the 26-letter alphabet of Table 2.2 with an affine, character

transformation of the form

$$f(p) = ap + b.$$

Alice and Bob agree to use the field $\mathbb{Z}_{677}$ with $g = 2$, which is a generator of the group of units. The group of units ${\mathbb{Z}_{677}}^*$ has exactly $676 = 26^2$ elements, which, very conveniently, is the exact same size as the keyspace, ${\mathbb{Z}_{26}}^2$. It is not necessary to get a 1-1 correspondence between the generating field and the keyspace, but a roughly equal chance of each key is important for security.

Alice and Bob agree to a mapping of $g^{\alpha\beta}$ to (a, b), *i.e.* from ${\mathbb{Z}_{677}}^*$ to ${\mathbb{Z}_{26}}^2$. The mapping from the field's group of units to the key space will begin by reducing the obtained $g^{ab} \pmod{676}$ (if necessary). The element of $\mathbb{Z}_{26^2}$ will be converted to an element of ${\mathbb{Z}_{26}}^2$ by writing it in base-26, the digits of which will be the 2 elements of the key.

If the obtained a is not an invertible element of $\mathbb{Z}_{26}$ then Alice and Bob will choose two new random exponents.

Alice and Bob choose random exponents, α and β, less than 676: Alice chooses 674 and Bob chooses 136 at random.

Alice and Bob each calculate their public key (g^α and g^β respectively) by raising the generator g to their respective powers (α and β) in $\mathbb{Z}_{677}$: Alice calculates $g^\alpha = 2^{674} = 508$ and Bob calculates $g^\beta = 2^{136} = 148$.

Alice and Bob trade their individual public keys and each calculate their shared private key, $g^{\alpha\beta}$: Alice calculates $g^{\alpha\beta} = (g^\beta)^\alpha = 148^{674} = 189$ and Bob calculates $g^{\alpha\beta} = (g^\alpha)^\beta = 508^{136} = 189$.

Alice and Bob, individually, each write 189 in base-26 (no reduction mod 676 is necessary

this time) as 7:7. They each verify that 7 is an invertible element of $\mathbb{Z}_{26}$, so they will use the encryption key $(7, 7)$. They can each calculate the decryption key with relative ease from the encryption key to decode each other's messages.

This example used a numeric field of prime size. It is also common in practice to use a Galois field with polynomial representation and map the coefficients into the key elements.

If a weak classical system is used after a Diffie-Hellman key exchange, then it would be easily broken. However, if a reasonably strong classical system is used, and a different key is chosen randomly for each conversation between two parties, then their keys will not be easily broken. An adversary will have a significantly reduced chance of accumulating enough cyphertext by the same key for an accurate statistical analysis attack and will have to resort to a more computationally intensive attack against the classical cypher or the discrete logarithm.

The security of the Diffie-Hellman key exchange depends not only on the difficulty of solving the discrete logarithm but the difficulty of the related problem of computing g^{ab} from only g^a and g^b. While, clearly, with a solution to the discrete logarithm to the base g, a and b can be computed separately and then g^{ab} computed, but it remains an open question whether g^{ab} could practically be computed directly from g^a and g^b without solving the discrete logarithm.[3] It is commonly assumed that such a computation would be equally difficult, if not even equivalent to solving the logarithm.

Definition 4.6 (Diffie-Hellman Assumption)**.** It is computationally infeasible to compute g^{ab} knowing only g^a and g^b.

[3]Koblitz, 99. [4]

4.4 Signatures

Signatures provide a digital method of source verification. Rather than information security or secrecy, the purpose of signatures is information authenticity. A message's signature affirms that whom the message claims as its author is actually the message's author.

Signatures are useful when it isn't important for a message to be secret, but it is important that it not be tampered with during transmission. This is becoming more common as more media is being purchased digitally and downloaded over the Internet. It may not be important to encrypt an ebook, song, video, or piece of software, but it is important to know that the file downloaded is exactly the same as was sent: that an interlocutor has not injected a virus or other malware or in any other way tampered with the recently purchased merchandise.

Signatures use a type of cryptographic function called a **hash function**.

Definition 4.7 (Hash function)**.** Let $\mathcal{P}$ be the set of all plaintext messages and S a finite set. Then a **hash function**, $f\colon \mathcal{P} \to S$, is a function such that

- $f(P)$, for $P \in \mathcal{P}$, is "easy" to compute.
- $y = f(P)$, for a fixed $y \in S$, is "difficult" to solve for P.

The primary difference between a hash and a trap-door from Page 85, is that a hash is many-to-one, while the trap-door functions are one-to-one. Being many-to-one, means that the word "solve" must be used in a more broad sense. It must be "difficult" to find a message for a given hash, find a message whose hash matches a given message, or find any two messages with the same hash. When two inputs to a hash function have the same output it is called a hash **collision**.

Example 4.8. Polynomials are well-known by elementary algebra students to be easy to evaluate but hard to solve. Consider $p(x) = 7x^2 + 9x + 15 \in \frac{\mathbb{Z}_{26^3}[x]}{\langle x^3-1\rangle}$ which we intend to use for messages in our standard 26-character alphabet of Table 2.2 to generate hashes three characters long. We will use this polynomial in the following manner: We will initialize the running hash, h_0, to 0, then for each character, x, of a message, we compute the next running hash value with the formula $h_i = p(h_{i-1} + x)$ (in $\mathbb{Z}_{26^3}$).

Let us use this hash function to calculate a hash for the message

`START.`

i	x	$h_i = p(h_{i-1} + x)$	h_i as text
1	S = 18	$p(18) = 7(18^2) + 9(18) + 15 = 2445$	DQB
2	T = 19	$p(2445 + 19) = 7(2464^2) + 9(2464) + 15 = 4919$	HHF
3	A = 0	$p(4919 + 0) = 7(4919^2) + 9(4919) + 15 = 5149$	HQB
4	R = 17	$p(5149 + 17) = 7(5166^2) + 9(5166) + 15 = 8945$	NGB
5	T = 19	$p(8945 + 19) = 7(8964^2) + 9(8964) + 15 = 16307$	YDF

The final hash of the message is "YDF." Naturally, this example hash is not strong enough for real-world use. It's small size makes a brute-force search for messages whose hashes collide a short exercise and the use of a quadratic p make direct algebraic solving attempts practical.

Real-world hashes are significantly longer (256 to 1024 bits) and tend to make heavy use of vectors of words (16-, 32- or 64-bits each), with bitwise operations (or, and, not and xor),

bitwise rotations, and permutations being performed on the elements of the running hash (state) vector and message block vector.

The general method used to sign a message starts with the author generating the hash, $h = f(P)$, of the plaintext message, performing a private-key encyphering on the hash value, and sending the encyphered hash with the message. The recipient computes the hash of the received plaintext, possibly after decyphering the cyphertext, uses the author's public key to decypher the signature, and compares the two values. If the two values match then the authenticity of the received message is confirmed.

Example 4.9. Suppose Alice wants to send the message

```
Once more
```

to Bob. She plans to use the standard 26-character alphabet of Table 2.2. To ensure that the message cannot be tampered with or forged, she will sign her message with her RSA private key of $(7, 143 = 11 \cdot 13)$. Her public key $(103, 143)$ is already known to Bob who is expecting all messages from Alice to be signed using her key. Alice and Bob also planned to use the hash of Example 4.8.

Alice does not plan on encyphering the whole message. In this case it is not important to be secret, as long as Bob can be guaranteed of the authenticity of any message claiming to be from Alice.

Alice first calculates the 3-character hash of the message using the same method described above. This comes out to IHB. Alice may then encypher the hash (going from 1-graphs in $\mathbb{Z}_{26}$ to 2-graphs in $\mathbb{Z}_{26^2}$), and get the signature of CFAGAB. Alice appends the signature to the

message and either sends them as-is, or may encypher the whole packet with Bob's public key.

Bob recieves the message. If Alice encyphered with his public key, he decyphers with his private key. He segregates the signature from the message. He hashes the message to get IHB. He applies Alices public key to the signature and also gets IHB. Since the hashes match, he may consider the message to be authentic.

There are only two ways that a plaintext message could be tampered with such that the signature matches. First, an imposter could resign the tampered message with the "author's" key, but that would require breaking the "author's" key, which is "difficult." Second, the imposter could modify the message in a way that leaves the hash unchanged, but that would require a solution to the hash function, which is also "difficult." Thus, signatures affirm the authenticity of a received message.

Chapter 5

Solving "Hard" Problems

This chapter explores a small sample of the known ways to solve the two "hard" problems from Chapter 4 upon which the security of the related private-key cryptosystems depends. A "success" is considered when a method is found that grows logarithmically with the length of the inputs, *i.e.* when a method runs twice as long when the input is squared, rather than when the input is doubled. Success is also counted when the time requirement for an algorithm is a polynomial of the logarithm of the input, *e.g.* $\mathcal{O}((\ln n)^k)$. These are labeled as running in **polynomial time**, being referenced to the logarithm of the input. Failures, whose running time of a polynomial of the inputs, are labeled as running in **exponential time**, also with regard to the logarithm of the input, *e.g.* $\mathcal{O}(n^k)$.

5.1 Prime Factorization

RSA encryption depends on the difficulty of factoring large integers. One that possesses another's public key, but not private, knows a composite integer n, but not any of its factors.

Definition 5.1. The integer factorization problem. Given a composite integer n, find a (not necessarily prime) $1 < p < n$ such that p is a factor of n.

By repeatedly finding proper divisors, the full prime factorization will eventually be found. A sample of common methods of integer factorization follows.

5.1.1 Naïve Trial Division

Naïve factoring works rather quickly on very small integers (less than 20 digits or so). It has no initial setup or overhead, and jumps straight into the work. The principle is simple: try dividing the integer, n, by every prime less than its square root, $\sqrt{n}$. Some implementations forgo the effort of identifying prime numbers by trying to divide by every odd integer.

Example 5.2. Factor 611 by trial division.

p	$\frac{n}{p}$
2	305.5
3	$203.\overline{66}$
5	122.2
7	$87.28\ldots$
11	$55.\overline{54}$
13	47

Thus, $611 = 13 \times 47$.

Again, that may not seem so bad for such a small n (or more particularly, for an n with such small factors), but when n is 123463, then the first prime divisor found won't be until 331. In an RSA application, n has only two factors that differ in length by only a few digits. The running time will then grow roughly with the square root of n.

5.1.2 Pollard's $p-1$ Algorithm

Pollard's $p-1$ factoring algorithm depends on Fermat's Little Theorem and the Chinese Remainder Theorem.

Theorem 5.3 (Chinese Remainder Theorem)**.** *If R is a Principal Ideal Domain (which $\mathbb{Z}$ is), $\{u_1, u_2, \ldots, u_n\}$ are pairwise relatively prime ($i \neq j \Rightarrow \langle u_i \rangle \cap \langle u_j \rangle = \langle u_i u_j \rangle$) and $u = u_1 u_2 \cdots u_n$, then*

$$\frac{R}{\langle u \rangle} \cong \frac{R}{\langle u_1 \rangle} \times \frac{R}{\langle u_2 \rangle} \times \cdots \times \frac{R}{\langle u_n \rangle}.$$

The isomorphic transformation from left to right is simply reduction mod u_i at each coordinate. The transformation from right to left is much trickier. It usually occupies the bulk of the discussion of the Chinese Remainder Theorem in an introductory text on number theory, and the formulas involved in solving such a system of modular congruences is irrelevant to this theoretical discussion. This vast difference in difficulties in navigating the isomorphism in the two directions make this a trap-door function.

The Chinese Remainder Theorem may be stated more simply with regard to an RSA appli-

cation:

$$\mathbb{Z}_{pq} \cong \mathbb{Z}_p \times \mathbb{Z}_q,$$

where

$$a \mapsto (a \mod p, a \mod q).$$

Fermat's Little Theorem, when stated in a number theoretic context, can look rather impressive.

Theorem 5.4 (Fermat's Little Theorem)**.** *If p is a prime integer and a is relatively prime to p (gcd(a, p) = 1), then $a^{p-1} \equiv 1 \pmod{p}$, or in other words, p divides $a^{p-1} - 1$.*

In an algebraic context, Fermat's Little Theorem merely says that the order of each element of the group of units of the ring $\mathbb{Z}_p$, divides $p - 1$ if p is prime. But if p is prime then $\mathbb{Z}_p$ is a field and the size of its group of units is $p - 1$. LaGrange's Theorem says the order of each element of a group divides the size of the group.

There are composite numbers, n, such that every $a \in \mathbb{Z}_n{}^*$ has an order dividing $n - 1$, even though $n - 1 \neq |\mathbb{Z}_n{}^*|$. This will happen whenever n is the product of distinct primes and each $p - 1$ divides $n - 1$.[1] If $n = \prod_{i=1}^{l} p_i$ then every $a \in \mathbb{Z}_n*$ will have an order dividing $\gcd(\{p_i\})$ which itself divides $n - 1$ because each $p - 1$ does. For example, $561 = 3 \cdot 11 \cdot 17$ is one such n: $\phi(561) = 2 \cdot 10 \cdot 16$ where each $p - 1$ divides 560.

Pollard's $p - 1$ method of factoring works best when one of the factor fields is smooth: its group of units has many subgroups. Specifically, for an RSA number $n = pq$, either $p - 1$ or $q - 1$ has only small prime factors. Suppose $n = pq$ and $p - 1$ is smooth. Let a be any integer

[1] Koblitz, 128. [4]

strictly between 1 and n. Let k be a product of lots of small prime numbers, for example $k = K!$ or $k = \text{lcm}(1, 2, \ldots, K)$. Then $a^k \mapsto (a_p, a_q)^k = ({a_p}^k, {a_q}^k)$. If $p-1$ divides k but $q-1$ does not, then ${a_p}^k$ will equal 1 (in $\mathbb{Z}_p$), ${a_q}^k$ probably won't (in $\mathbb{Z}_q$), so $a^k - 1$ will be a multiple of p (in $\mathbb{Z}_n$). Thus $p = gcd(n, a^k - 1)$ and a factoring of n has been found. If $p-1$ does not divide k either, then $gcd(n, a^k - 1) = 1$ and a larger (with more bigger factors) k should be tried.

Example 5.5. Factor $n = 123463$ with Pollard's $p-1$ method.

First, pick an a from $\{2, \ldots, n-1\}$. This may be done randomly or systematically. Let us use $a = 2$. Quickly check the gcd of a and n in case of a lucky guess. In this case the gcd is 1, so 2 is not a factor of 123463.

Second, find a product of a lot of small numbers. Let's use $k = \text{lcm}(1, 2, \ldots, 4) = 12$.

Calculate $a^k - 1$ in $\mathbb{Z}_n$. In practice this is done using the repeated squaring method described in Section 1.6.2 in $\mathcal{O}((\lg k)(\lg n)^2)$ time. $2^{12} - 1 = 4095$.

Calculate the gcd of $a^k - 1$ and n. This can be done in $\mathcal{O}((\lg n)^3)$ time with the Euclidean Algorithm as shown in Theorem 1.20. $\gcd(4095, 123463) = 1$.

The gcd came out to 1. This means that the k used was too small, it is missing at least one factor of $p-1$. Let's try again with a larger k. Let's use $k = \text{lcm}(1, 2, \ldots, 8) = 840$.

Calculate $a^k - 1$ in $\mathbb{Z}_n$. $2^{840} - 1 = 72820$.

Calculate the gcd of $a^k - 1$ and n. $\gcd(72820, 123463) = 331$.

If the gcd is strictly between 1 and n, then a non-trivial factor was found. Divide n by that factor to find the other factor. $n = 331 \times 373$.

If the gcd is n, then try a smaller k or a different a. We might have tried $a = 3$ next had our first try been unsuccessful. This example would require a very large k to end up with a gcd

of 123463. 372 factors into $2^2 \cdot 3 \cdot 31$. So k would have to be a multiple of 31 before the gcd would be 123463. In fact, since 2 generates ${\mathbb{Z}_{373}}^*$, k would have to be a multiple of 4, 3 and 31.

The three major time-consuming steps are the calculations of k, a^k and the gcd. Calculating k involves roughly K multiplications of size $K \times k$, running in $\mathcal{O}(K(\lg K)(\lg k))$ time. Calculating a^k runs in $\mathcal{O}((\lg k)(\lg n)^2)$ time. And calculating the gcd runs in $\mathcal{O}((\lg n)^3)$ time. Since $k = \mathcal{O}(e^K)$, the whole algorithm runs in $\mathcal{O}(K^2(\lg K) + K(\lg n)^2 + (\lg n)^3)$ time. Simplifying this estimate requires a relationship between K and n.

The actual K required can vary greatly depending on the smoothness of both n and $p-1$ for each prime factor p. A smoother n has more prime factors, meaning that they will tend to be smaller (at least one will have to be), and a smaller p means a smaller $p-1$. And smaller $p-1$'s mean smaller factors of $p-1$ which require a smaller k and/or K. For each $p \mid n$, the largest factor of $p-1$ is the critical one in determining the required K.

If R is the set of greatest prime factors of the $p-1$'s, and r is the least of these greatest prime factors, then this r is the key determining factor and the running time will be $\mathcal{O}(r\frac{\ln n}{\ln r})$ multiplications, according to Menezes.[2] The worst case running time for Pollard's $p-1$ algorithm is when $n = pq$ (only two factors, exactly what RSA is), and $p-1$ and $q-1$ each have a very large factor, being of the form $2 \cdot r$. The smallest r, a required factor of k, will be roughly equal to $\frac{1}{2}\sqrt{n}$. This case is not very likely, but still a possible worst-case, with a running time of $\mathcal{O}(n\frac{1}{2}\lg n + \sqrt{n}(\lg n)^2 + (\lg n)^3) = \mathcal{O}(n \lg n)$. Plugging in $r = \sqrt{n}$ to Menezes' estimate and multiplying by the time of each multiplication would give $\mathcal{O}(\sqrt{n}(\lg n)^2)$ if it were presumed that each multiplication were mod n, or at least on the order of n, which is slightly

[2] Menezes, 93. [5]

better than our estimate. Menezes actually gives a slightly more efficient method of computing $k = \text{lcm}(1, 2, \ldots, K)$ and a^k that does keep each multiplication on the order of n, while our estimate of multiplications on the order of $K \times k$ to calculate the lcm is higher and ended up dominating our estimate.

5.1.3 Lenstra's Elliptic Curve Algorithm

The points of an elliptic curve,[3] including ones over finite fields, form an additive group structure. The group of points of an elliptic curves over $\mathbb{Z}_{pq}$ are isomorphic to the cross product of the groups of points of the elliptic curves over $\mathbb{Z}_p$ and $\mathbb{Z}_q$. This allows the equivalent of Pollard's $p-1$ algorithm to be used on the group of points on an elliptic curve over $\mathbb{Z}_n$ to find a factorization: with multiples of a chosen point replacing powers of a.

The computations involved in working with points on an elliptic curve are more intense than computing with integers, but this extra computational intensity is made up for by a key difference in the group structures of elliptic curves and integral groups of units. The size of the group of units of a field $\mathbb{Z}_p$ is always $p - 1$, and that is why Pollard's $p - 1$ algorithm depends on at least one smooth $p - 1$ from the factors of n, which is not often enough the case. The size of an elliptic curve over the field $\mathbb{Z}_p$ can vary from $p + 1 - 2\sqrt{p}$ to $p + 1 + 2\sqrt{p}$,[4] and will vary within this range over different curves even for the same p.

This freedom in the actual size of the curve over $\mathbb{Z}_p$ increases the chance of finding an elliptic curve whose number of points is a smooth enough number to permit a successful factoring with

[3] For an easy introduction to elliptic curves and a detailed explanation of Lenstra's Elliptic Curve Factoring Algorithm, please see Silverman's *Rational Points on Elliptic Curves.* [8]

[4] Silverman, 107-110. [8]

a significantly smaller K than would be required by Pollards $p-1$ Algorithm. This increased chance, however, is not a guarantee, so, while it is significantly more efficient than Pollard's $p-1$ algorithm, Lenstra's Elliptic Curve Algorithm still has a running time dependent on the nature of the factors of n and will run too slowly on non-preferred values of n. In it's worst case, where n is the product of two prime factors of very close size, this algorithm's running time is $\mathcal{O}(e^{\sqrt{(\ln n)(\ln(\ln n))}})$.[5]

5.1.4 Fermat's Factorization Methods

The running times of Naïve Trial Division, Pollard's $p-1$ Algorithm and Lenstra's Elliptic Curve Algorithm are dependent on the character of the factorization of the integer in question. Because of this, they tend to lose effectiveness on larger numbers, where such special characteristics become rarer, and are thus considered special purpose algorithms. Their running times grow with either the smoothness of n or of the groups involved ($p-1$ or the smoothest size of a used elliptic curve over $\mathbb{Z}_p$), rather than directly with the size of n.

Fermat's Factorization Method serves as the framework for the currently used general purpose algorithms. The running times of these algorithms tend to be more dependent on the size of n than on the characterization of its factorization. This is beneficial for attacks on RSA keys, where the p and q are intentionally chosen so that special purpose factoring algorithms like the ones previously described are impractical in an attack.

The basis of Fermat's Factorization Method is the Difference of Squares special form bino-

[5]Menezes, 94. [5]

mial:

$$a^2 - b^2 = (a + b)(a - b).$$

All odd numbers can be expressed as a difference of squares, $2k + 1 = (k + 1)^2 - k^2$. ("Trial" division can extract all factors of 2 quite easily.) Once such a difference-of-squares expression is found, then the number can be factored as a difference of squares. In practice an exact difference of squares representation, $n = a^2 - b^2$, is not needed, but merely a congruence of squares, $a^2 - b^2 \equiv 0 \pmod{n}$, such that a and b themselves are neither congruent nor congruently opposite ($a \not\equiv \pm b$), otherwise one of the two factors would be congruent to n.

Once an appropriate congruence of squares is found, $a^2 \equiv b^2 \pmod{n}$ where $a \not\equiv \pm b \pmod{n}$, then a proper factorization is found. Since $a \not\equiv \pm b \pmod{n}$, then n will divide neither $a + b$ nor $a - b$. But since $(a + b)(a - b) \equiv 0 \pmod{n}$ n does divide their product. Therefore, each of the difference-of-squares factors contains a proper factor of n, which can be found by finding the gcd of the factor and n.

Example 5.6. The number $15 = 3 \times 5$ is odd.

15 may be expressed as $16 - 1 = 4^2 - 1^2 = (4 + 1)(4 - 1) = 5 \cdot 3$.

15 may also be expressed as $64 - 49 = 8^2 - 7^2 = (8 + 7)(8 - 7) = 15 \cdot 1$. However, this expression of 15 as a difference of squares will not yield a factorization because $8 \equiv -7 \pmod{15}$, $8 + 7 \equiv 0 \pmod{15}$.

The congruence of squares $49 \equiv 4 \pmod{15}$ holds. $45 = 49 - 4 = 7^2 - 2^2 = (7 + 2)(7 - 2) = 9 \cdot 5$. Once we have this factoring, calculating $\gcd(9, 15) = 3$ which properly divides 15 and $\gcd(5, 15) = 5$ which also properly divides 15.

The congruence of squares $49 \equiv 484 \pmod{15}$ holds. $435 = 484 - 49 = 22^2 - 7^2 = (22+7)(22-7) = 29 \cdot 15$. Calculating the gcd's reveals 1 and 15. This was because $22 \equiv 7 \pmod{15}$ and therefore $22 - 7 \equiv 0 \pmod{15}$.

Naïve Search

If p and q are close to each other, then they are also each close to $\sqrt{n}$ (as $\sqrt{n}$ is the geometric mean of p and q). Then a quick search along the sequence $(a^2 - n)_{a>\sqrt{n}}$ will find a square, b^2, such that $a^2 - b^2 \equiv 0 \pmod{n}$. Knowing this, RSA pairs of primes are chosen so that p and q differ in size by at least a couple digits, rendering such a naïve search a linear-time pursuit. Thus, merely searching the sequence $(i^2 - n)_{i>\sqrt{n}}$ for a square is considered a special purpose algorithm ultimately on par with trial division on worst-case running time, yet with significantly higher rates of worst-case occurrences.[6]

Example 5.7. Factor 123463 by naïvely searching for a difference of squares representation.

Start with $a = \lceil \sqrt{123463} \rceil = 352$.

$$352^2 - 123463 = 441 \qquad = 3^2 * 7^2$$

Therefore 123463 factors as $(352 - 21)(352 + 21) = 331 * 373$.

This example aptly demonstrates the ease by which a number may be factored with a naïve Fermat factorization search when p and q are close. When p and q are not close a naïve search will be impractical. In such cases a congruence of squares may be found by multiplying an

[6] Pomerance, 1474. [6]

appropriately chosen set of $a^2 \equiv a^2 - n \pmod{n}$ relations. Clearly the product of the first components, which are all squares, will also be one. Finding a subset of pairs whose product of ${a_i}^2 - n \pmod{n} \in \mathbb{Z}_n$ is a square is usually accomplished by factoring them in $\mathbb{Z}$ and inspecting the parity (even vs. odd) of the exponents on their prime factorizations. A set whose product contains only even exponents in the prime factorization will yield a congruence of squares relation which, if the roots are not themselves equal or opposites in $\mathbb{Z}_n$, will allow a factoring of n in $\mathbb{Z}$ as a difference of squares.

Example 5.8. Factor 36181 with Fermat Factorization.

Start searching with $a_0 = \lceil \sqrt{36181} \rceil = 191$.

$$191^2 \equiv \ \ 300 = 2^2 \cdot 3 \cdot 5^2$$

$$192^2 \equiv \ \ 683 = 683$$

$$193^2 \equiv 1068 = 2^2 \cdot 3 \cdot 89$$

$$194^2 \equiv 1455 = 3 \cdot 5 \cdot 97$$

$$195^2 \equiv 1844 = 2^2 \cdot 461$$

$$196^2 \equiv 2235 = 3 \cdot 5 \cdot 149$$

$$197^2 \equiv 2628 = 2^2 \cdot 3^2 \cdot 73$$

$$198^2 \equiv 3023 = 3023$$

$$199^2 \equiv 3420 = 2^2 \cdot 3^2 \cdot 5 \cdot 19$$

$$200^2 \equiv 3819 = 3 \cdot 19 \cdot 67$$

$$201^2 \equiv 4220 = 2^2 \cdot 5 \cdot 211$$

$$202^2 \equiv 4623 = 3 \cdot 23 \cdot 67$$

$$203^2 \equiv 5028 = 2^2 \cdot 3 \cdot 419$$

$$204^2 \equiv 5435 = 5 \cdot 1087$$

$$205^2 \equiv 5844 = 2^2 \cdot 3 \cdot 487$$

$$206^2 \equiv 6255 = 3^2 \cdot 5 \cdot 139$$

$$207^2 \equiv 6668 = 2^2 \cdot 1667$$

$$208^2 \equiv 7083 = 3^2 \cdot 787$$

$$209^2 \equiv 7500 = 2^2 \cdot 3 \cdot 5^4$$

A congruence of squares is found with only 2 relations: $a = 191$ and $a = 209$. Thus $(191^2)(209^2) \equiv (2^2 \cdot 3 \cdot 5^2)(2^2 \cdot 3 \cdot 5^4) = 2^4 \cdot 3^2 \cdot 5^6 \pmod{36181}$. $191 \cdot 209 \equiv 3738 \pmod{36181}$

and $2^2 \cdot 3 \cdot 5^3 = 1500$, which are neither equal nor opposite (mod 36181), so a helpful congruence of squares is found and a proper factoring follows. $\gcd(3738-1500, 36181) = 373$ and $\gcd(3738+1500, 36181) = 97$. Therefore $36181 = 97 \cdot 373$.

Often times, for at least two reasons, when looking for a congruence of squares, a "small" set of primes is chosen as a **factor base**, and only the $a^2 - n$ that factor over this set of primes are retained to find an appropriate subset to multiply together. First, the point of this Fermat Factorization is to avoid factoring large prime factors out of a number, it would be costly to factor large primes out of the reductions of the a^2. Second, by restricting the search to a^2 whose reductions factor over a predetermined set of primes one limits the difficulty of finding a set whose product is a square.

This allows each $a^2 - n$ to be represented by a finite-length vector of the exponents of its prime factorization. And since only the parity of the exponents is a concern, they may be reduced mod 2 and considered members of ${\mathbb{F}_2}^m$ where m is the number of primes in the factor base. Now finding a set of relations whose product is a square is equivalent to finding a set of exponent vectors in ${\mathbb{F}_2}^m$ whose sum is 0, or, in other words, is linearly dependent.

Example 5.9. Let's factor $n = 1234589$ by finding a system of prime factorizations to multiply together. Notice that $n = 277 \cdot 4457$. These two factors are separated enough that a linear search for a straight $a^2 - b^2 = n$ will take a long time. Trial division, actually, would work very quickly with such a small factor as 277. But let's collect the factorizations of the modular residues of a^2 that have no prime factors over 50, to find a square product thereof.

Start searching with $a_0 = \lceil \sqrt{1234589} \rceil = 1112$. But only record when the prime factorization

after the modular reduction has only factors less than 50.

$$1112^2 \equiv \quad 1955 = 5 \cdot 17 \cdot 23$$

$$1113^2 \equiv \quad 4180 = 2^2 \cdot 5 \cdot 11 \cdot 19$$

$$1129^2 \equiv \quad 40052 = 2^2 \cdot 17 \cdot 19 \cdot 31$$

$$1132^2 \equiv \quad 46835 = 5 \cdot 17 \cdot 19 \cdot 29$$

$$1142^2 \equiv \quad 69575 = 5^2 \cdot 11^2 \cdot 23$$

$$1157^2 \equiv 104060 = 2^2 \cdot 5 \cdot 11^2 \cdot 43$$

$$1158^2 \equiv 106375 = 5^3 \cdot 23 \cdot 37$$

$$1165^2 \equiv 122636 = 2^2 \cdot 23 \cdot 31 \cdot 43$$

$$1208^2 \equiv 224675 = 5^2 \cdot 11 \cdot 19 \cdot 43$$

$$1217^2 \equiv 246500 = 2^2 \cdot 5^3 \cdot 17 \cdot 29$$

$$1227^2 \equiv 270940 = 2^2 \cdot 5 \cdot 19 \cdot 23 \cdot 31$$

At this point a square congruence is found with the following lines:

$$1112^2 \equiv 5 \cdot 17 \cdot 23$$

$$1129^2 \equiv 2^2 \cdot 17 \cdot 19 \cdot 31$$

$$1227^2 \equiv 2^2 \cdot 5 \cdot 19 \cdot 23 \cdot 31$$

giving the congruence

$$(1112 \cdot 1129 \cdot 1227)^2 \equiv (2^2 \cdot 5 \cdot 17 \cdot 19 \cdot 23 \cdot 31)^2 \pmod{1234589}.$$

But $1112 \cdot 1129 \cdot 1227 \equiv 2^2 \cdot 5 \cdot 17 \cdot 19 \cdot 23 \cdot 31 \pmod{1234589}$ so the search continues.

$$1243^2 \equiv 310460 = 2^2 \cdot 5 \cdot 19^2 \cdot 43$$

Quickly another congruence is found with 1157 and 1243, $(1157 \cdot 1243)^2 \equiv (2^2 \cdot 5 \cdot 11 \cdot 19 \cdot 43)$. This time the two roots are not congruent or congruently opposite, so a proper factorization will follow.

$$\gcd(1157 \cdot 1243 - 2^2 \cdot 5 \cdot 11 \cdot 19 \cdot 43, 1234589) = 277$$

$$\gcd(1157 \cdot 1243 + 2^2 \cdot 5 \cdot 11 \cdot 19 \cdot 43, 1234589) = 4457$$

The factor base may be freely chosen. There are trade-offs between large and small factor bases, and an efficient balance is desirable. Too small a factor base will yield very few ${a_i}^2 - n$ that factor completely over the factor base, lengthening the time of that search. It would have required a lot more searching to find congruences that factored into primes only up to 11 or 19 in Example 5.9. Too large a factor base will increase the number of congruences necessary to find a linearly dependent set, not necessarily increasing the length of the search for the congruences, as more tried elements of the sequence will be factorable over the base, but certainly increasing the size of the linear system and, thus, the computations required to solve it. There would have been many more congruences in Example 5.9 if factoring into primes as high as 100 were allowed, but then the system would have been much harder to deal with in searching for a set to multiply.

An exploration into the time complexity of this algorithm is beyond the scope of this paper. Pomerance discusses the running times of these more advanced factorization algorithms. The time does depend on the chosen factor base, and will be too long if the factor base is either too small or too large. The running time will be minimized when the chosen factor base is the set of

all primes less than or equal to approximately $P = e^{\frac{1}{2}\sqrt{\ln 2\sqrt{n} \ln\ln 2\sqrt{n}}}$ and that minimal running time will be $\mathcal{O}(e^{\sqrt{\ln n \ln\ln n}})$.[7]

The Quadratic Sieve

The Quadratic Sieve is an efficient means of identifying which elements of the sequence ${a_i}^2 \pmod n$ factor completely over the chosen factor base. Rather than trial dividing each ${a_i}^2 \pmod n$ by all elements of the factor base, all the elements of a pre-compiled range of ${a_i}^2 \pmod n$ that can be divided by each prime are identified, and that prime is divided out from all of them.

A set of a_i's are chosen ahead of time, as is the factor base. Koblitz recommends a factor base of $B = \{p \leq P\}$ where P is approximately $e^{\sqrt{\ln n \ln\ln n}}$ and the set of a_i's of $\{\lfloor\sqrt{n}\rfloor + 1, \lfloor\sqrt{n}\rfloor + 2, \ldots \lfloor\sqrt{n}\rfloor + A\}$ where $P < A < P^2$.[8] After running through all of the primes in the factor base, B, any ${a_i}^2 \pmod n$ that end up divided all the way down to 1 factors over the factor base and passes to the next stage to find a square product.

Considering whether $a^2 \pmod n$ is divisible by a prime p is equivalent to considering when $p \mid a^2 - n$ or $a^2 \equiv n \pmod p$. Therefore, if n is not a square in $\mathbb{Z}_p$, then p cannot divide $a^2 - n$ for any a. This means that only the primes for which n is a square need to be retained in the factor base. This piece of information could be used in the previous algorithm to reduce the number of primes by which to trial divide, here it will be used to limit the number of sieving steps.

[7] Pomerance, 1477,1478. [6]

[8] Koblitz, 161. [4]

Limiting the factor base this way will roughly cut the size of the factor base in half, as half of all elements of a field (of odd characteristic) are squares. Also, limiting the factor base this way is only valid if all a's used are between $\sqrt{n}$ and $\sqrt{2n}$, where the reduction of a^2 is equivalent to subtracting n. If any a's are used where $xn < a^2 < (k+1)n$ then those would have to be sieved with $kn \pmod p$ in the following manner rather than just n. Having to use a's all the way to $\sqrt{2n}$ would end up with a running time of at least $\%_0(\sqrt{n})$ in which case trial division would have been just as well. This may only happen for small enough n where a simpler algorithm would be faster. For large enough n, the asymptotic running time given at the end of this section will be better than $\%_0(\sqrt{n})$.

Further $a^2 \equiv n \pmod p \Leftrightarrow a \equiv \pm\sqrt{n} \pmod p$. Thus $a^2 \pmod n$ will be divisible by p if, and only if, a is congruent to one of the square roots of $n \pmod p$. The congruence $x^2 \equiv n \pmod p$ need only be solved once for each prime. Then for each root, x_i, identify an a_i that is congruent to it $\pmod p$. Not only will that ${a_i}^2 \pmod n$ be divisible by p, but so will every $a_i + kp^2 \pmod n$ be divisible by p. Thus roughly $\frac{2}{p}$ of all the ${a_i}^2 \pmod n$ can be divided by each prime, and they can all be easily (faster than by trial division) identified. "Trial" divisions now need only be done when divisibility is already known, and are thus no longer "trial."

The above will identify only one factor of each prime. To identify prime factors of higher multiplicities (such as the 4 that divides 60), the above steps should be repeated, for each prime, modulo each successive power of the prime, until a power α is found for which no a_i is congruent to a square root of $n \pmod{p^\alpha}$. Once p is divided out for all a_i's congruent to a $\sqrt{n} \pmod p$, the search should be repeated for a_i's that are congruent to a $\sqrt{n} \pmod{p^2}$ and a second p divided from their entries. Repeat this for p^3 and so on, until a power is reached for which no

more congruent a_i's are found. After which, move on to the next prime.

It is a basic fact of number theory that for the odd primes, being a square in any particular $\mathbb{Z}_{p^\alpha}$ guarantees being a square in all $\mathbb{Z}_{p^\alpha}$, thus the same set of primes are to be checked regardless of power. This will have two important implications for us. Firstly, that if (the reduction of) n is a square in $\mathbb{Z}_{p^\alpha}$, then it is also a square in $\mathbb{Z}_p$, meaning that once the set of primes are set for the factor base considering the primes' first powers, no extras are needed for higher powers. Secondly, once we find square roots mod p, we should keep looking for, and will keep finding square roots mod each successive p^α until they've spread far enough out that no more are found in the chosen range of a_i's.

Theorem 5.10. *For any odd prime, p, if n (not a multiple of p) is a square in one $\mathbb{Z}_{p^\alpha}$ then it is a square in all $\mathbb{Z}_{p^\alpha}$. Only 1 is a square is $\mathbb{Z}_2$, $\mathbb{Z}_4$ and $\mathbb{Z}_8$, and being a square in one $\mathbb{Z}_{2^\alpha}$ is equivalent to being a square in all other $\mathbb{Z}_{2^\alpha}$ for $\alpha \geq 3$.*

Proof. This will be a simple proof by induction on α. First we'll show that if n is a square in $\mathbb{Z}_{p^\alpha}$ then it is a square in $\mathbb{Z}_{p^1}$. This includes $p = 2$.

If n is a square in $\mathbb{Z}_{p^\alpha}$ then there exists an x such that $x^2 \equiv n \pmod{p^\alpha} \Rightarrow p^\alpha \mid x^2 - n$. That also means that $p \mid x^2 - n$ and so n is also a square in $\mathbb{Z}_p$.

And now the inductive step: if n is a square in $\mathbb{Z}_{p^\alpha}$ then it is also a square in $\mathbb{Z}_{p^{\alpha+1}}$, when p is an odd prime.

If n is a square in $\mathbb{Z}_{p^\alpha}$ then there exists an x such that $x^2 \equiv n \pmod{p^\alpha} \Rightarrow p^\alpha \mid x^2 - n$. But that x is not unique, any number congruent to $x \pmod{p^\alpha}$ will also work, so we can say that $p^\alpha \mid (x + kp^\alpha)^2 - n$ for any $k \in \mathbb{Z}$. For the following simplification, we'll use the fact that p^α divides $x^2 - n$ to write $x^2 - n$ in the form $p^\alpha l$.

$$\begin{aligned} p^\alpha \mid & (x + kp^\alpha)^2 - n \\ &= x^2 + 2xkp^\alpha + k^2p^{2\alpha} - n \\ &= p^\alpha l + 2xkp^\alpha + k^2p^{2\alpha} \\ &= p^\alpha(l + 2xk + k^2p^\alpha) \end{aligned}$$

Because 2 and x are units in $\mathbb{Z}_p$, k may be chosen so that $2xk = -l$, or in other words $p \mid l + 2xk$. Therefore, an additional p may be factored out from $(l + 2xk + k^2p^\alpha)$ and $p^{\alpha+1} \mid (x + kp^\alpha)^2 - n$. Therefore n is also a square in $\mathbb{Z}_{p^{\alpha+1}}$.

In $\mathbb{Z}_2$, 1 is the square of itself. In $\mathbb{Z}_4$, 1 is the square of 1 and 3. In $\mathbb{Z}_8$, 1 is the square of 1, 3, 5 and 7. Thus, while all odd n will be squares in $\mathbb{Z}_2$ ($n \equiv 1 \pmod 2$), only half of those will be squares in $\mathbb{Z}_4$ ($n \equiv 1 \pmod 4$, but not 3), and only half of those will be squares in $\mathbb{Z}_8$ ($n \equiv 1 \pmod 8$, but not 5), though they will have 4 square roots.

The inductive step above only worked for odd primes because 2 is not a unit in $\mathbb{Z}_{2^\alpha}$. The above inductive step will only yield a square in $\mathbb{Z}_{2^{\alpha+1}}$ if l is even, if n were already a square in $\mathbb{Z}_{2^{\alpha+1}}$: $x^2 - n = l2^\alpha = \frac{l}{2}2^{\alpha+1}$.

Suppose now that l is odd. Note that x is also odd. Let k be another odd number and

consider

$$\begin{aligned}(x + k2^{\alpha-1})^2 - n &= x^2 + 2xk2^{\alpha-1} + k^2 2^{2\alpha-2} - n \\ &= x^2 - n + xk2^{\alpha} + k^2 2^{2\alpha-2} \\ &= l2^{\alpha} + xk2^{\alpha} + k^2 2^{2\alpha-2} \\ &= (l + xk)2^{\alpha} + k^2 2^{2\alpha-2}.\end{aligned}$$

Because l, x and k are all odd, $l+xk$ is even and so contains another factor of 2. If $2\alpha-2 \geq \alpha+1$, which happens whenever $\alpha \geq 3$, then the entire expression will be a multiple of $2^{\alpha+1}$. Therefore $2^{\alpha+1} \mid (x + k2^{\alpha-1})^2 - n$, meaning that $(x + k2^{\alpha-1})^2 \equiv n \pmod{2^{\alpha+1}}$, than n is a square in $\mathbb{Z}_{2^{\alpha+1}}$. □

As primes are divided out of the ${a_i}^2$'s, record should be kept of how many of each prime in the factor base were divided out, and the remaining quotient after all primes in the factor base are divided out. The set of all a_i which factor completely over the factor base (leave a quotient of 1 after all primes in the base have been fully divided out during the sieving) should then be segregated out for further computations. These are the a_i's whose prime factorizations of ${a_i}^2$ should be examined for a set whose product is a square.

Being a square means that all of the exponents in the prime factorization are all even. When sets of congruences are multiplied together, the exponents of the prime factors get added. In identifying which ones will multiply to a square (only even exponents) only the mod 2 reductions of the exponents need to be considered: p^2 is as much a square as p^4 and as p^6, likewize p is as much not a square as p^3 and p^5.

The simplest way to find the set of prime factorizations which is a square, is by converting the prime factorizations into vectors of the exponents (mod 2). At that point, all that needs to be found is a linearly independent set of vectors whose sum is the zero vector. Once that is found, then those vectors included in the sum will correspond precisely with the factorizations whose product is a square.

Example 5.11. Use the quadratic sieve to filter the congruences when factoring 36181.

First a factor base should be chosen. As mentioned earlier, a good choice is the set of all primes less than or equal to $B = e^{\frac{1}{2}\sqrt{\ln 2\sqrt{36181}\ln\ln 2\sqrt{36181}}} \approx 5.09$. So our factor base will be all primes less than or equal to 5 for which 36181 has quadratic residue, which is all of them because $36181 \equiv 1$ mod each of them. For out set of a_i's, a quick glance up at Example 5.8 shows that only 18 were needed, which is in the earlier stated range, but for pedagogical reasons, let us choose $A = 25$ for maximum impact. Thus, our table would start as follows.

a_i	${a_i}^2$
191	300 =
192	683 =
193	1068 =
194	1455 =
195	1844 =
196	2235 =
197	2628 =
198	3023 =
199	3420 =
200	3819 =
201	4220 =
202	4623 =
203	5028 =
204	5435 =
205	5844 =
206	6255 =
207	6668 =
208	7083 =
209	7500 =
210	7919 =
211	8340 =
212	8763 =
213	9188 =
214	9615 =
215	10044 =

Sieving shall be performed by increasing primes. 36181 is congruent to 1 (mod 2) and is also congruent to 1 (mod 4), but to 5 (mod 8). 36181 is congruent to 1 (and is therefore a square) mod 2 and 4, but not 8. For each a_i congruent to 1 (mod 2) (which are the square roots of n in $\mathbb{Z}_2$, which are all the odd a_i, ${a_i}^2 - n$ will be divisible by 2. Next, for each a_i congruent to 1 or 3 (mod 4) (which are the square roots of 1 in $\mathbb{Z}_4$), also all the odd a_i, ${a_i}^2 - n$ is divisible by another 2. However, since 5 is not a quadratic residue (mod 8), no ${a_i}^2 - n$ is divisible by a third 2. Therefore 2^2 can be factored out of every other entry leaving odd cofactors behind. The table may be updated as follows.

a_i	${a_i}^2$
191	$300 = 2^2 \cdot 75$
192	$683 = 683$
193	$1068 = 2^2 \cdot 267$
194	$1455 = 1455$
195	$1844 = 2^2 \cdot 461$
196	$2235 = 2235$
197	$2628 = 2^2 \cdot 657$
198	$3023 = 3023$
199	$3420 = 2^2 \cdot 855$
200	$3819 = 3819$
201	$4220 = 2^2 \cdot 1055$
202	$4623 = 4623$
203	$5028 = 2^2 \cdot 1257$
204	$5435 = 5435$
205	$5844 = 2^2 \cdot 1461$
206	$6255 = 6255$
207	$6668 = 2^2 \cdot 1667$
208	$7083 = 7083$
209	$7500 = 2^2 \cdot 1875$
210	$7919 = 7919$
211	$8340 = 2^2 \cdot 2085$
212	$8763 = 8763$
213	$9188 = 2^2 \cdot 2297$
214	$9615 = 9615$
215	$10044 = 2^2 \cdot 2511$

Because $36181 \equiv 1 \pmod 3$, for any a_i that is congruent to 1 or 2 (mod 3), ${a_i}^2 - 36181$ will be divisible by 3: $191 + 3k$ and $193 + 3k$. 36181 is also congruent to 1 (mod 9), so a second 3 can be factored out for the ones congruent to 1 and 8: $197 + 9k$ and $199 + 9k$. 36181 is also congruent to 1 (mod 27), so a third 3 can be factored out for 215, which is congruent to 26 (mod 27). 215 is also congruent to 53 (mod 81), whose square, 55, is congruent to 36181, so a fourth 3 can be factored out for 215. 36181 is congruent to 217 (mod $3^5 = 243$), whose square roots are 109 and 134. None of the a_i are congruent to 109 or 134 (mod 243), so no more 3s can be factored. The quadratic sieve table now looks as follows.

a_i	${a_i}^2$
191	$300 = 2^2 \cdot 3 \cdot 25$
192	$683 = 683$
193	$1068 = 2^2 \cdot 3 \cdot 89$
194	$1455 = 3 \cdot 485$
195	$1844 = 2^2 \cdot 461$
196	$2235 = 3 \cdot 745$
197	$2628 = 2^2 \cdot 3^2 \cdot 73$
198	$3023 = 3023$
199	$3420 = 2^2 \cdot 3^2 \cdot 95$
200	$3819 = 3 \cdot 1273$
201	$4220 = 2^2 \cdot 1055$
202	$4623 = 3 \cdot 1541$
203	$5028 = 2^2 \cdot 3 \cdot 419$
204	$5435 = 5435$
205	$5844 = 2^2 \cdot 3 \cdot 487$
206	$6255 = 3^2 \cdot 695$
207	$6668 = 2^2 \cdot 1667$
208	$7083 = 3^2 \cdot 787$
209	$7500 = 2^2 \cdot 3 \cdot 625$
210	$7919 = 7919$
211	$8340 = 2^2 \cdot 3 \cdot 695$
212	$8763 = 3 \cdot 2921$
213	$9188 = 2^2 \cdot 2297$
214	$9615 = 3 \cdot 3205$
215	$10044 = 2^2 \cdot 3^4 \cdot 31$

Proceeding the same way with powers of 5, a 5 can be factored out for $191+5k$ and $194+5k$. 36181 is congruent to 6 (mod 25), whose square roots are 9 and 16, so a second 5 can be factored out for 191 and 209. 36181 is congruent to 56 (mod 125), whose square roots are 41 and 84, so a third 5 can be factored out for 209. 36181 is also congruent to 556 (mod 625), whose square roots are 209 and 416, so a fourth 5 can be factored out for 209. But the square roots of 36181 (mod 5^5) are 834 and 2291, so no more 5s can be factored out.

a_i	${a_i}^2$	
191	300 =	$2^2 \cdot 3 \cdot 5^2$
192	683 =	683
193	1068 =	$2^2 \cdot 3 \cdot 89$
194	1455 =	$3 \cdot 5 \cdot 97$
195	1844 =	$2^2 \cdot 461$
196	2235 =	$3 \cdot 5 \cdot 149$
197	2628 =	$2^2 \cdot 3^2 \cdot 73$
198	3023 =	3023
199	3420 =	$2^2 \cdot 3^2 \cdot 5 \cdot 19$
200	3819 =	$3 \cdot 1273$
201	4220 =	$2^2 \cdot 5 \cdot 211$
202	4623 =	$3 \cdot 1541$
203	5028 =	$2^2 \cdot 3 \cdot 419$
204	5435 =	$5 \cdot 1087$
205	5844 =	$2^2 \cdot 3 \cdot 487$
206	6255 =	$3^2 \cdot 5 \cdot 139$
207	6668 =	$2^2 \cdot 1667$
208	7083 =	$3^2 \cdot 787$
209	7500 =	$2^2 \cdot 3 \cdot 5^4$
210	7919 =	7919
211	8340 =	$2^2 \cdot 3 \cdot 5 \cdot 139$
212	8763 =	$3 \cdot 2921$
213	9188 =	$2^2 \cdot 2297$
214	9615 =	$3 \cdot 5 \cdot 641$
215	10044 =	$2^2 \cdot 3^4 \cdot 31$

The sieving is done. Any entry still containing a factor that is not a member of the factor base, will not factor over the factor base and will be discarded. The factorizations presented in this final table are not necessarily prime factorizations, even though many of them are. The final factor is merely the product of all the prime factors that are not members of the factor base. Dropping all entries that do not factor over the chosen factor base leaves the following two lines.

a_i	${a_i}^2$
191	$300 = 2^2 \cdot 3 \cdot 5^2$
209	$7500 = 2^2 \cdot 3 \cdot 5^4$

In an actual RSA scale application, there will be tens of thousands of lines with tens of thousands of prime factors. This example is interesting in that there are not enough lines to guarantee a linearly dependent set of exponent vectors, though such a set does exist, notably because both exponents of 2 and 5 are even. When rewritten as a matrix of exponents reduced mod 2, the following equation arises.

$$\begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_{191} \\ x_{209} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Clearly the not-trivial solution is

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

implying that $191^2 \cdot 209^2 \equiv 2^4 \cdot 3^2 \cdot 5^6$, yielding the factoring described above.

The Quadratic Sieve algorithm has a running time of $\mathcal{O}(e^{\sqrt{(\ln n)(\ln(\ln n))}})$ to factor the integer n.[9] While this growth rate is the same as for Lenstra's Elliptic Curve Algorithm,[10] actual running times will differ by a constant factor. The Quadratic Sieve algorithm has faster practicals, and so gains the benefit of the constant factor between the two running times. This is mostly due to the fact that the Quadratic Sieve is working in a much easier group, basic integers in a $\mathbb{Z}_l$ as opposed to two-dimensional points on an elliptic curve.[11]

The General Number Field Sieve

The Quadratic Sieve algorithm has now been superseded by the General Number Field Sieve with better asymptotic behavior. Its fanciness and complexity are beyond the scope of this paper, but a brief overview will be given.

One chooses a degree, d, and builds an irreducible, monic polynomial, f, whose coefficients are in the range of $\sqrt[d]{n}$, and an integer, m, such that m is also in the range of $\sqrt[d]{n}$ and $f(m) \equiv 0 \pmod{n}$. Then take one of the polynomial's complex roots, α, the extension ring $\mathbb{Z}[\alpha]$, and the natural ring homomorphism $\Phi\colon \mathbb{Z}[\alpha] \to \mathbb{Z}_n$ by $\alpha \mapsto m \mod n$.

With these tools in hand, a set of $\{(a-\alpha b)\}$ is chosen/found such that $\prod(a-\alpha b)$ is a square in $\mathbb{Z}[\alpha]$. Once this set is found, then a natural, and hopefully helpful, congruence of squares follows. The properties of Φ guarantee that $\Phi(\prod(a-\alpha b)) \equiv \prod \Phi(a-\alpha b) \equiv \prod(a-mb)$, and if $\prod(a-\alpha b)$ is a square, then so will be $\prod(a-mb)$.[12]

[9] Pomerance, 1478. [6]

[10] See Section 5.1.3.

[11] Menezes, 97. [5]

[12] See Pomerance [6] and his references for additional practical concerns such as how to identify squares in $\mathbb{Z}[\alpha]$ and, once recognized, finding their square roots.

The General Number Field Sieve has a asymptotic running time of $\mathcal{O}(e^{(\frac{64}{9}\ln n)^{\frac{1}{3}}(\ln(\ln n))^{\frac{2}{3}}})$ to factor the integer n.[13] This is a significant improvement over the Quadratic Sieve method, but the extra complexities over the Quadratic Sieve render it's use to larger n, from around 150 digits and larger.[14]

5.2 Discrete Logarithm

The difficulty of computing logarithms in a discrete field was touched upon in Section 4.3. Solving a discrete logarithm involves finding the exponent required to raise a base to obtain a given element.

Definition 5.12 (The Discrete Logarithm Problem)**.** Given a cyclic group $\langle b \rangle$ and an arbitrary member, $g \in \langle b \rangle$, find a natural number $k \in \mathbb{N}$ such that $b^k = g$ but $i < k \Rightarrow b^i \neq g$

In cryptographic applications, a finite field is usually used and a b is found that generates all of $\mathbb{F}^*$ as the cyclic group. This maximizes the size of the group being used, which is desirable for the security of the cryptosystem being implemented.

A sample of common methods of solving the discrete logarithm problem follows.

5.2.1 Naïve

Just as an integer may be factored by trial division, the logarithm of $g = b^k$ may be found by evaluating all powers of b until the desired one is found, commonly known as trial multiplication. This will require $k-1$ multiplications. If b generates all of $\mathbb{F}^*$ and g is a random non-zero member,

[13] Pomerance, 1482. [6]

[14] Koblitz, 164. [4]

then roughly $\frac{1}{2}|\mathbb{F}|$ multiplications will be required on average, which puts the running time of trial multiplication at $\mathcal{O}(|\mathbb{F}|)$.

5.2.2 Silver-Pohlig-Hellman Algorithm

The Silver-Pohlig-Hellman algorithm requires the knowledge of the size of $\langle b \rangle$, which is usually known when b is a generator of the group of units of a known field, and the factorization of the size. For a field of size q, the ease of finding the factorization of $q-1$ depends on the size and/or smoothness of $q-1$.[15] The efficiency of the Silver-Pohlig-Hellman algorithm also depends on the smoothness of $q-1$. This algorithm will work best when $q-1$ is smooth.

The Silver-Pohlig-Hellman algorithm works by finding the residues of k mod each maximal prime power dividing $q-1$ then computes k by the Chinese Remainder Theorem. Write the prime factorization of $q-1$ as $\prod_{i=1}^{l} {p_i}^{\alpha_i}$. Then if the least positive residue of k mod each ${p_i}^{\alpha_i}$, k_p, can be found, then k can be calculated from the $\{k_p\}$ by the Chinese Remainder Theorem.

The details can get murky, so we'll demonstrate a simple example before describing the algorithm, then show a more involved example.

Example 5.13. We'll use the field $\mathbb{Z}_{101}$. $100 = 2^2 5^2$. At this point, many of the calculations become too tedious to perform by hand, so an advanced calculator or computer algebra software would be handy. To find a generator, b, we need $b^{\frac{100}{2}} = b^{50}$ and $b^{\frac{100}{5}} = b^{20}$ to each not equal 1.

$$2^{50} = 100$$

$$2^{20} = 95$$

[15] See Section 5.1.

Thus 2 is a generator of ${\mathbb{Z}_{101}}^*$. Let's find $\log_2 25$ in $\mathbb{Z}_{101}$.

First we find the sets of square and fifth roots of unity. We'll label the primitive roots r_2 and r_5, and the elements of the set as $r_{p,j} = {r_p}^j$ where $j < p$. Because 2 is a generator of ${\mathbb{Z}_{101}}^*$, $2^{\frac{100}{2}} = 2^{50}$ will be a primitive square root of unity and $2^{\frac{100}{5}} = 2^{20}$ will be a primitive fifth root of unity.

$$2^{50} = 100$$

$$2^{20} = 95$$

$$2^{40} = 36$$

$$2^{60} = 87$$

$$2^{80} = 84$$

$r_{p,j}$	2	5
0	1	1
1	100	95
2	-	36
3	-	87
4	-	84

We're trying to find the value of k where $25 = 2^k$. First we'll find $k_2 = k \pmod 4$

We start finding $k_2 = k \pmod 4$ by first thinking of k_2 as a base-2 number: $x_1 : x_0 = x_0 + 2x_1$. We'll find x_0 by evaluating 25^{50} and comparing the result with the square roots of

unity.

$$25^{50} = 1 = r_{2,0}.$$

Therefore $x_0 = 0$. If it had been non-zero, then we would have divided 2 off of 25 before finding x_1.

To find x_1 we'll evaluate 25^{25} and compare the result with the square roots of unity.

$$25^{25} = 1 = r_{2,0}.$$

Therefore $x_1 = 0$ also. This means that $k_2 = 0$, that $k \equiv 0 \pmod 4$.

We'll do the same work to find $k_5 = k \pmod{25}$. k_5 as a base-5 number would look like $x_1 : x_0 = x_0 + 5x_1$. We'll find x_0 by evaluating 25^{20} and comparing the result with the fifth roots of unity.

$$25^{20} = 87 = r_{5,3}.$$

Therefore $x_0 = 3$. Before finding x_1 we'll have to divide 2^3 off of 25 to get 2^{k-3}.

$$\frac{25}{8} = 41.$$

To find x_1 we'll evaluate 41^4 and compare the result with the fifth roots of unity.

$$41^4 = 84 = r_{5,4}.$$

Therefore $x_1 = 4$. This means that $k_5 = x_0 + 5x_1 = 3 + 5 \cdot 4 = 23$.

$k_2 = 0$ and $k_5 = 23$ gives the following system of congruences.

$$k \equiv \ \ 0 \pmod 4$$

$$k \equiv 23 \pmod{25}$$

This system has the solution $k \equiv 48 \pmod{100}$. We can verify this by evaluating $2^{48} = 25$.

To find each k_p, the p^{th} roots of unity must be calculated. These may be found as $r_{p,j} = b^{\frac{j(q-1)}{p_i}}$ for $j = 0, 1, \ldots p_i - 1$. Observe that this would require a large table for any large primes dividing $q-1$, hence the desire for a smooth $q-1$. Each k_{p_i} may then be written as an α_i-digit number in base p_i.

$$k_{p_i} = x_{\alpha_i - 1} : x_{\alpha_i - 2} : \cdots : x_1 : x_0 = x_0 + x_1 p_i + \cdots + x_{\alpha_i - 1} {p_i}^{\alpha_i - 1}$$

To find x_0 compute

$$g^{\frac{(q-1)}{p_i}} = b^{\frac{k(q-1)}{p_i}} = b^{\frac{k_{p_i}(q-1)}{p_i}}$$

The denominator cancels out with all parts of k_{p_i} except the x_0 leaving

$$g^{\frac{(q-1)}{p_i}} = b^{\frac{x_0(q-1)}{p_i}}$$

the result of which is one of the p^{th} roots of unity and may be compared with the entries in the table of roots of unity and x_0 is the corresponding j.

Then let $g_1 = g \cdot b^{-x_0}$. This subtracts x_0 from k and k_{p_i} so that the next digit may be found. At this point ${g_1}^{\frac{q-1}{p_i}}$ would be 1 because g_1 is a p_i-multiple power of b: $g_1 = b^{p_i m}$ for some m. Finding x_1 will require ${p_i}^2$ in the denominator of the exponent. Compute

$$g_1^{\frac{(q-1)}{{p_i}^2}} = b^{\frac{(k-x_0)(q-1)}{{p_i}^2}} = b^{\frac{(k_{p_i}-x_0)(q-1)}{{p_i}^2}} = b^{\frac{x_1(q-1)}{p_i}} = r_{p_i, x_1}$$

to find x_1 and let $g_2 = g_1 \cdot b^{-x_1 p_i}$. Continue in this manner with increasing powers of p_i until all of k_{p_i} is found.

Repeat the above to find each k_{p_i}, then use the Chinese Remainder Theorem to compute k. Note that if the same group and generator thereof is used multiple times, then the table of prime roots of unity need only be generated once.

Example 5.14. Let $\mathbb{F}_{49} = \frac{\mathbb{F}_7[t]}{\langle t^2-2t+3\rangle}$, and let $b = t$, a generator of ${\mathbb{F}_{49}}^*$. Find the discrete logarithm of $g = 6t+1$ to the base t in $\mathbb{F}_{49}$. As above, all computations should be performed with algebra software for expediency, otherwise make good use of the method of repeated squaring.

Let's verify that t is a generator by ensuring that neither the square nor cube root of t^{48} is 1. $48 = 2^4 \cdot 3$.

$$t^2 = 2t - 3$$

$$t^4 = (2t-3)^2 = 3t - 3$$

$$t^8 = (3t-3)^2 = 3$$

$$t^{\frac{48}{3}} = t^{16} = 3^2 = 2$$

$$t^{\frac{48}{2}} = t^{24} = t^{16}t^8 = 2 \cdot 3 = 6$$

And, indeed, $6^2 = 1$ and $2^3 = 1$. If t were not a generator of ${\mathbb{F}_{49}}^*$, it would have generated a subgroup whose order divides 48 and one of those two powers of t would have equaled 1. So t is a generator of ${\mathbb{F}_{49}}^*$.

Next, we find the p^{th} roots of unity for each prime dividing 48: 2 and 3. The square roots are $r_{2,0} = t^{\frac{0(48)}{2}} = 1$ and $r_{2,1} = t^{\frac{1(48)}{2}} = 6 = -1$. The cube roots are $r_{3,0} = 1$, $r_{3,1} = t^{\frac{1(48)}{3}} = 2$ and $r_{3,2} = t^{\frac{2(48)}{3}} = 2^2 = 4$. Thus our lookup table is as follows.

$r_{p,j}$	2	3
0	1	1
1	6	2
2	-	4

To find k_2, we first think of it as $x_0 + x_1 \cdot 2 + x_2 \cdot 2^2 + x_3 \cdot 2^3$ and compute the "digits" of this "base-2 number." First, compute $g^{\frac{48}{2}} = (6t+1)^{24} = 1 = r_{2,0}$, so $x_0 = 0$. Let $g_1 = \frac{g}{t^{x_0}} = \frac{g}{t^0} = 6t+1$. Second, compute $g_1{}^{\frac{48}{2^2}} = (6t+1)^{12} = 1 = r_{2,0}$, so $x_1 = 0$. Let $g_2 = \frac{g_1}{t^{0 \cdot 2}} = \frac{6t+1}{1} = 6t+1$. Third, compute $g_2{}^{\frac{48}{2^3}} = (6t+1)^6 = 6 = r_{2,1}$, so $x_2 = 1$. Let $g_3 = \frac{g_2}{t^{1 \cdot 2^2}} = \frac{6t+1}{t^4} = \frac{6t+1}{3t+4} = 2$. Lastly, compute $g_3{}^{\frac{48}{2^4}} = 2^3 = 1 = r_{2,0}$, so $x_3 = 0$. Now,

$$\begin{aligned} k_2 &= x_0 + x_1 \cdot 2 + x_2 \cdot 2^2 + x_3 \cdot 2^3 \\ &= 0 + 0 \cdot 2 + 1 \cdot 2^2 + 0 \cdot 2^3 \\ &= 4 \end{aligned}$$

To find k_3, we merely calculate $g^{\frac{48}{3}} = (6t+1)^{16} = 4 = r_{3,2}$, so $k_3 = 2$. Nothing more with higher powers of 3 needs to be calculated, because 3 divides into 48 only once.

To summarize the events so far, $k \equiv 8 \pmod{2^4}$ and $k \equiv 2 \pmod{3}$. This gives the following system of congruences.

$$\begin{aligned} k &\equiv 4 \pmod{16} \\ k &\equiv 2 \pmod{3} \end{aligned}$$

By the Chinese Remainder Theorem, this k, which clearly equals 20, is unique $\pmod{48}$. Therefore $6t + 1 = t^{20}$. This can be verified by evaluating it to check.

Presuming that $q-1$ can be easily factored into $q - 1 = \prod_{i=1}^{k} {p_i}^{e_i}$, then this algorithm has a running time of $\mathcal{O}\left(\sum_{i=1}^{k} e_i \left(\ln(q-1) + \sqrt{p_i}\right)\right)$.[16]

[16] Menezes, 108. [5]

5.2.3 Index-Calculus Algorithm

The Index-Calculus Algorithm for solving the discrete logarithm problem bears a striking resemblance to the Quadratic Sieve integer factoring algorithm. The Index-Calculus Algorithm also uses a factor base and establishes linear relations of exponents of factorings over the selected base. In solving the equation $g = b^x$ for $g \in G = \langle b \rangle$ the following steps constitute the Index-Calculus Algorithm:

1. Select a factor base $B = \{p_i\}$ such that a "significant proportion" of the elements of G will factor over it.

2. Select random (or systematic) k's to find ones such that b^k factors over B, $b^k = \prod {p_i}^{e_i}$, to generate the linear relations of the form $k = \sum e_i \log_b p_i$ until a dependent system is found with which to solve for the $\log_b p_i$.

3. Solve the system to solve for the logarithms of the factor base elements.

4. Select random (or systematic) k's until one is found such that gb^k factors over the factor base, indicating that $k + \log_b g = \sum k_i \log_b p_i$.

5. Evaluate $\log_b g$.

While step 3 depends on step 2, just like in the Quadratic Sieve Algorithm, and the individual relations of step 2 can be found in parallel, just like in the Quadratic Sieve Algorithm, a factoring of a gb^k in step 4 can be found in parallel with the other two. And, similarly to the Quadratic Sieve Algorithm, the Index-Calculus Algorithm also has a trade-off between the size of the factor base, and therefore of the linear system, and the ease of filling the linear system.

Example 5.15. Take the field $\mathbb{F}_{3^5}$ represented by $\frac{\mathbb{F}_3[t]}{\langle t^5-t+1\rangle}$ where t generates ${\mathbb{F}_{3^5}}^*$. Find the $\log_t(t^4 - t^3 + t^2 - t + 1)$.

We compute $t^{22} = -t^3 - t^2 + t + 1$ and $t^{121} = -1$ to verify that t does generate all of ${\mathbb{F}_{3^5}}^*$. $3^5 - 1 = 242 = 2 \cdot 11^2$, so if t wasn't a generator then one of those powers of t would have been 1.

It is customary to choose a factor base that is the set of all irreducible polynomials up to a certain degree. It would be simple to try using a factor base of just the linears, but that turns out to allow no more than 11 linearly independent factorizations of powers of t: -1 is the only constant that can be counted, and has order 2. The modulus polynomial clearly indicates that $t^5 = t - 1$, which gives one linear relation of exponents (in fact it gives the $\log_t(t-1) = 5$, but also prevents higher powers of $(t-1)$ being useful for generating independent linear relations, as $\log(t-1)^\alpha = 5\alpha$. Since much time could be spent looking in a field of 243 elements for one that is a power of $(t+1)$ up to 4 or an opposite thereof, a larger factor base should be considered.

Allowing irreducible monic quadratics adds $t^2 + 1$, $t^2 + t - 1$, and $t^2 - t - 1$ to the factor base. With a total of 6 irreducibles in the factor base, an element of $\mathbb{F}_{3^5}$ that factors thereover could be found by selecting up to 4 of the six base elements (not even counting multiplicity), which yields at least $\binom{6}{1} + \binom{6}{2} + \binom{6}{3} + \binom{6}{4} = 56$ elements, which might be substantial enough.

Our factor base is $B = \{-1, t \pm 1, t^2 + 1, t^2 \pm t - 1\}$ and has 6 elements.

The first step is to find at least 6 k's such that t^k factors over B. In practice, extras are usually found to compensate for the probable discovery of dependent relations. The first four powers of t offer no information, so we'll start at 5. We'll also skip over any powers that give no information, those will be the powers where there is no modular rollover (through a $t^5 = t - 1$)

from the preceding power.

$$
\begin{aligned}
t^5 &= & t-1 &= \\
t^9 &= & -t^4+t-1 &= -(t+1)(t^3-t^2+t+1) \\
t^{10} &= & t^2+t+1 &= (t-1)^2 \\
t^{13} &= & t^4+t^3+t-1 &= (t^2+1)(t^2+t-1) \\
t^{14} &= & t^4+t^2-1 &= \\
t^{15} &= & t^3-1 &= (t-1)^3 \\
t^{17} &= & -t^2+t-1 &= -(t+1)^2 \\
t^{20} &= & t^4-t^3-t+1 &= (t-1)^4 \\
t^{21} &= & -t^4-t^2-t-1 &= -(t^4+t^2+t+1) \\
t^{22} &= & -t^3-t^2+t+1 &= -(t-1)(t+1)^2 \\
t^{24} &= -t^4+t^3+t^2-t+1 &= -(t^4-t^3-t^2+t-1) \\
t^{25} &= & t^4+t^3-t^2+1 &= (t+1)(t^3-t+1) \\
t^{26} &= & t^4-t^3-t-1 &= (t^2+1)(t^2-t-1) \\
t^{27} &= & -t^4-t^2-1 &= -(t+1)^2(t-1)^2 \\
t^{28} &= & -t^3+t+1 &= -(t^3-t-1) \\
t^{30} &= & t^3+t^2-t+1 &= \\
t^{32} &= & t^4-t^3+t^2+t-1 &= \\
t^{33} &= & -t^4+t^3+t^2-1 &= -(t-1)(t^3-t-1) \\
t^{34} &= & t^4+t^3+t+1 &= (t+1)^4 \\
t^{35} &= & t^4+t^2-t-1 &= (t-1)(t^3+t^2-t+1) \\
t^{36} &= & t^3-t^2-1 &= (t+1)(t^2+t-1)
\end{aligned}
$$

Eight k's have been found that are not higher powers of $t-1$ and yield the following linear relations of logarithms.

$$5 = \log(t-1)$$

$$13 = \log(t^2+1) + \log(t^2+t-1)$$

$$17 = \log(-1) + 2\log(t+1)$$

$$22 = \log(-1) + \log(t-1) + 2\log(t+1)$$

$$26 = \log(t^2+1) + \log(t^2-t-1)$$

$$27 = \log(-1) + 2\log(t-1) + 2\log(t+1)$$

$$34 = 4\log(t+1)$$

$$36 = \log(t+1) + \log(t^2+t-1)$$

One of these logarithms, however, is already known. $\log(-1)$ was discovered when t was verified to be a generator of the group of units at the beginning of this example and is 121. This simplifies the system of equations to the following in only 5 unknowns.

$$
\begin{aligned}
5 &= \log(t-1) \\
13 &= \log(t^2+1) + \log(t^2+t-1) \\
138 &= 2\log(t+1) \\
143 &= \log(t-1) + 2\log(t+1) \\
26 &= \log(t^2+1) + \log(t^2-t-1) \\
148 &= 2\log(t-1) + 2\log(t+1) \\
34 &= 4\log(t+1) \\
36 &= \log(t+1) + \log(t^2+t-1)
\end{aligned}
$$

This system now has a pair (third and seventh) that are linearly dependent so one of them may be discarded. We'll discard the one in the seventh position. Our linear system of seven equations now has the following matrix equation in $\mathbb{Z}_{242}$.

$$
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
2 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
2 & 2 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
\log(t+1) \\
\log(t-1) \\
\log(t^2+1) \\
\log(t^2+1-1) \\
\log(t^2-t-1)
\end{bmatrix}
=
\begin{bmatrix}
5 \\
13 \\
138 \\
143 \\
26 \\
148 \\
36
\end{bmatrix}
$$

This system's augmented matrix reduces in $\mathbb{Z}_{242}$ to

$$\left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 1 & 49 \\ 0 & 1 & 0 & 0 & 0 & 5 \\ 0 & 0 & 1 & 0 & 1 & 46 \\ 0 & 0 & 0 & 1 & -1 & 229 \\ 0 & 0 & 0 & 0 & 2 & 202 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

which gives the solution as one of

$$\begin{bmatrix} \log(t+1) \\ \log(t-1) \\ \log(t^2+1) \\ \log(t^2+t-1) \\ \log(t^2-t-1) \end{bmatrix} = \begin{bmatrix} 49-\log(t^2-t-1) \\ 5 \\ 26-\log(t^2-t-1) \\ 229+\log(t^2-t-1) \\ 101 \text{ or } 222 \end{bmatrix}.$$

One quick trial of $t^{101} \neq t^2-t-1$ reveals the solution to be

$$\begin{bmatrix} \log(t+1) \\ \log(t-1) \\ \log(t^2+1) \\ \log(t^2+t-1) \\ \log(t^2-t-1) \end{bmatrix} = \begin{bmatrix} 69 \\ 5 \\ 46 \\ 209 \\ 222 \end{bmatrix}.$$

Now that the logarithms of the factor base elements are known. A search begins for another k

such that $(t^4 - t^3 + t^2 - t + 1)t^k$ factors over B.

$$(t^4 - t^3 + t^2 - t + 1)t^1 = -t^4 + t^3 - t^2 - t - 1 = -(t+1)(t-1)(t^2 - t - 1)$$

And by quite a stroke of luck, such a k was found very quickly. Therefore

$$\log((t^4 - t^3 + t^2 - t + 1)t) = \log((-1)(t+1)(t-1)(t^2 - t - 1)$$

$$\log(t^4 - t^3 + t^2 - t + 1) + 1 = 121 + 69 + 5 + 222$$

$$\log(t^4 - t^3 + t^2 - t + 1) = 174$$

The running time of the Index-Calculus Algorithm depends on the selected factor base. A judicious selection of a factor base is highly dependent on the working field. But if a optimal factor base can be chosen, then the Index-Calculus Algorithm has an expected running time of $\mathcal{O}(e^{c\sqrt{\ln n \ln \ln n}})$, where n is the size of the working field and c is known to vary between about 1.5 and 2 for the common fields of $\mathbb{F}_{2^m}$ and $\mathbb{Z}_p$ respectively.[17] Notice how similar this algorithm is to the Fermat factorization with a factor base, their running times even have the same asymptotic behavior (see page 118)

5.3 Conclusion

As stated at the beginning of this chapter (see page 103) success in the algorithms presented here is considered to be when the running time can be expressed as a polynomial of the length (number of bits/digits) of the inputs, a polynomial of the logarithm of the inputs. While some of the best algorithms known have been presented, none of them are successful in this regard. Nor are any of their more complex slightly faster superiors successful in this regard either.

[17] Menezes, 112. [5]

No algorithm is known yet for either of these problems whose running time can be expressed like $\mathcal{O}((\ln n)^k)$. Though, neither are these more efficient algorithms considered failures, for their running times are not polynomial in the inputs, they are not of the form $\mathcal{O}(n^k)$. These algorithms have running times that are faster than exponential, but still slower than polynomial, in the log of the input. They have been written $\mathcal{O}(e^{(\ln n)^\alpha (\ln \ln n)^{1-\alpha}})$. When written in this form, an exponential-time algorithm will have an α of 1 while a polynomial-time will have an α of 0. The better algorithms shown here and their superiors not examined here have running times with α's of $\frac{1}{2}$ or $\frac{1}{3}$.

The difficulty of finding successful methods of solving these factorization and logarithm problems has helped enforce these problems' use as the foundations of our modern cryptosystems. Without such earnest and fervent attempts, our trust in these systems would be unfounded. And, yet, with such attempts, our trust may eventually be betrayed. It is still yet unknown whether a polynomial-time algorithm is possible for either of these problems.

Appendix A

The Declaration of Independence

A.1 Plaintext

When in the Course of human events, it becomes necessary for one people to dissolve the political bands which have connected them with another, and to assume among the powers of the earth, the separate and equal station to which the Laws of Nature and of Nature's God entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the separation.

We hold these truths to be self-evident, that all men are created equal, that they are endowed by their Creator with certain unalienable Rights, that among these are Life, Liberty and the pursuit of Happiness. --That to secure these rights, Governments are instituted among Men, deriving their just powers from the consent of the governed, --That whenever any Form of Government becomes destructive of these ends, it is the Right of the People to alter or

to abolish it, and to institute new Government, laying its foundation on such principles and organizing its powers in such form, as to them shall seem most likely to effect their Safety and Happiness. Prudence, indeed, will dictate that Governments long established should not be changed for light and transient causes; and accordingly all experience hath shewn, that mankind are more disposed to suffer, while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, pursuing invariably the same Object evinces a design to reduce them under absolute Despotism, it is their right, it is their duty, to throw off such Government, and to provide new Guards for their future security. --Such has been the patient sufferance of these Colonies; and such is now the necessity which constrains them to alter their former Systems of Government. The history of the present King of Great Britain [George III] is a history of repeated injuries and usurpations, all having in direct object the establishment of an absolute Tyranny over these States. To prove this, let Facts be submitted to a candid world.

He has refused his Assent to Laws, the most wholesome and necessary for the public good.

He has forbidden his Governors to pass Laws of immediate and pressing importance, unless suspended in their operation till his Assent should be obtained; and when so suspended, he has utterly neglected to attend to them.

He has refused to pass other Laws for the accommodation of large districts of people, unless those people would relinquish the right of Representation in the Legislature, a right inestimable to them and formidable to tyrants only.

He has called together legislative bodies at places unusual, uncomfortable, and distant from the depository of their public Records, for the sole purpose of fatiguing them into compliance with his measures.

He has dissolved Representative Houses repeatedly, for opposing with manly firmness his invasions on the rights of the people.

He has refused for a long time, after such dissolutions, to cause others to be elected; whereby the Legislative powers, incapable of Annihilation, have returned to the People at large for their exercise; the State remaining in the mean time exposed to all the dangers of invasion from without, and convulsions within.

He has endeavoured to prevent the population of these States; for that purpose obstructing the Laws for Naturalization of Foreigners; refusing to pass others to encourage their migrations hither, and raising the conditions of new Appropriations of Lands.

He has obstructed the Administration of Justice, by refusing his Assent to Laws for establishing Judiciary powers.

He has made Judges dependent on his Will alone, for the tenure of their offices, and the amount and payment of their salaries.

He has erected a multitude of New Offices, and sent hither swarms of Officers to harass our people, and eat out their substance.

He has kept among us, in times of peace, Standing Armies without the consent of our legislatures.

He has affected to render the Military independent of and superior to the Civil power.

He has combined with others to subject us to a jurisdiction foreign to our constitution and unacknowledged by our laws; giving his Assent to their Acts of pretended Legislation:

For Quartering large bodies of armed troops among us:

For protecting them, by a mock Trial, from punishment for any Murders which they should commit on the Inhabitants of these States:

For cutting off our Trade with all parts of the world:

For imposing Taxes on us without our Consent:

For depriving us, in many cases, of the benefits of Trial by Jury:

For transporting us beyond Seas to be tried for pretended offences:

For abolishing the free System of English Laws in a neighbouring Province, establishing therein an Arbitrary government, and enlarging its Boundaries so as to render it at once an example and fit instrument for introducing the same absolute rule into these Colonies:

For taking away our Charters, abolishing our most valuable Laws, and altering fundamentally the Forms of our Governments:

For suspending our own Legislatures, and declaring themselves invested with power to legislate for us in all cases whatsoever.

He has abdicated Government here, by declaring us out of his Protection and waging War against us.

He has plundered our seas, ravaged our Coasts, burnt our towns, and destroyed the lives of our people.

He is at this time transporting large Armies of foreign Mercenaries to compleat the works of death, desolation and tyranny, already begun with circumstances of Cruelty and perfidy scarcely paralleled in the most barbarous ages, and totally unworthy the Head of a civilized nation.

He has constrained our fellow Citizens taken Captive on the high Seas to bear Arms against their Country, to become the executioners of their friends and Brethren, or to fall themselves by their Hands.

He has excited domestic insurrections amongst us, and has endeavoured to bring on the inhabitants of our frontiers, the merciless Indian Savages, whose known rule of warfare, is an undistinguished destruction of all ages, sexes and conditions.

In every stage of these Oppressions We have Petitioned for Redress in the most humble terms: Our repeated Petitions have been answered only by repeated injury. A Prince whose character is thus marked by every act which may define a Tyrant, is unfit to be the ruler of a free people.

Nor have We been wanting in attentions to our British brethren. We have warned them from time to time of attempts by their legislature to extend an unwarrantable jurisdiction over us. We have reminded them of the circumstances of our emigration and settlement here. We have appealed to their native justice and magnanimity, and we have conjured them by the ties of our common kindred to disavow these usurpations, which, would inevitably interrupt our connections and correspondence. They too have been deaf to the voice of justice and of consanguinity. We must, therefore, acquiesce in the necessity, which denounces our Separation, and hold them, as we hold the rest of mankind, Enemies in War, in Peace Friends.

We, therefore, the Representatives of the united States of America, in General Congress, Assembled, appealing to the Supreme Judge of the world for the rectitude of our intentions, do, in the Name, and by the Authority of the good People of these Colonies, solemnly publish and declare, That these United Colonies are, and of Right ought to be Free and Independent States; that they are Absolved from all Allegiance to the British Crown, and that all political connection between them and the State of Great Britain, is and ought to be totally dissolved; and that as Free and Independent States, they have full Power to levy War, conclude Peace, contract Alliances, establish Commerce, and to do all other Acts and Things which Independent States may of right do. And for the support of this Declaration, with a firm reliance on the protection of divine Providence, we mutually pledge to each other our Lives, our Fortunes and our sacred Honor.

A.2 Character Affine Encyphered Cyphertext


```
QV7ŤnŃŤnRV7n-IĞhİ7nIlnVĞ_cŤn7<7ŤRİjnŃRnŸ7MI_7İnŤ7M7İİch[nlIhnIŤ7n~7I~*7nRInĆŃİİI
    *<7nRV7n~I*ŃRŃMc*nŸcŤĆİnqVŃMVnVc<7
    nMIŤŤ7MR7ĆnRV7_nqŃRVncŤIRV7hjncŤĆnRIncİİĞ_7nc_IŤ!nRV7n~
    Iq7hİnIlnRV7n7chRVjnRV7nİ7~chcR7ncŤĆn73Ğc*nİRcRŃIŤnRInqVŃMVnRV7n
cqİnIlntcRĞh7ncŤĆnIlntcRĞh7aİnĄIĆn7ŤRŃR*7nRV7_jncnĆ7M7ŤRnh7İ~7MRnRInRV7nI~
    ŃŤŃIŤİnIln_cŤuŃŤĆnh73ĞŃh7İnRVcRnRV7[nİVIĞ*ĆnĆ7M*ch7nRV7nMcĞİ7İnqVŃMVnŃ_~7*
    nRV7_nRInRV7nİ7~chcRŃIŤTn‘Q7nVI*ĆnRV7İ7nRhĞRVİnRInŸ7nİ7*l§7<ŃĆ7ŤRjnRVcRnc**
    n_7Ťnch7nMh7cR7Ćn73Ğc*jnRVcRnRV7[nch7n7ŤĆIq7ĆnŸ[nRV7Ńhn-
    h7cRIhnqŃRVnM7hRcŃŤnĞŤc*Ń7ŤcŸ*7nHŃ!VRİjnRVcRnc_IŤ!nRV7İ7nch7n
Ńl7jn
ŃŸ7hR[ncŤĆnRV7n~ĞhİĞŃRnIln6c~~ŃŤ7İİTn§§2VcRnRInİ7MĞh7nRV7İ7nhŃ!VRİjnĄI<7
    hŤ_7ŤRİnch7nŃŤİRŃRĞR7Ćnc_IŤ!n?7ŤjnĆ7hŃ<ŃŤ!nRV7Ńhn@ĞİRn~
    Iq7hİnlhI_nRV7nMIŤİ7ŤRnIlnRV7n!I<7hŤ7Ćjn§§2VcRnqV7Ť7<7hncŤ[nLIh_nIlnĄI<7
    hŤ_7ŤRnŸ7MI_7İnĆ7İRhĞMRŃ<7nIlnRV7İ7n7ŤĆİjnŃRnŃİnRV7nHŃ!VRnIlnRV7n^7I~*7nRInc*
    R7hnIhnRIncŸI*ŃİVnŃRjncŤĆnRInŃŤİRŃRĞR7nŤ7qnĄI<7hŤ_7ŤRjn*c[ŃŤ!
    nŃRİnlIĞŤĆcRŃIŤnIŤnİĞMVn~hŃŤMŃ~*7İncŤĆnIh!cŤŃŘŃŤ!nŃRİn~
    Iq7hİnŃŤnİĞMVnlIh_jncİnRInRV7_nİVc**nİ77_n_IİRn*Ńu7*[nRIn7l17MRnRV7Ńhn}cl7R[
    ncŤĆn6c~~ŃŤ7İİTn^hĞĆ7ŤM7jnŃŤĆ77ĆjnqŃ**nĆŃMRcR7nRVcRnĄI<7hŤ_7ŤRİn*IŤ!n7İRcŸ*
    ŃİV7ĆnİVIĞ*ĆnŤIRnŸ7nMVcŤ!7ĆnlIhn*Ń!VRncŤĆnRhcŤİŃ7ŤRnMcĞİ7İĚncŤĆncMMIhĆŃŤ!*[nc
    **n7&~7hŃ7ŤM7nVcRVnİV7qŤjnRVcRn_cŤuŃŤĆnch7n_Ih7nĆŃİ~Ii7ĆnRInİĞll7hjnqVŃ*7n7<Ń
    *İnch7nİĞll7hcŸ*7jnRVcŤnRInhŃ!VRnRV7_İ7*<7İnŸ[ncŸI*ŃİVŃŤ!
    nRV7nlIh_İnRInqVŃMVnRV7[nch7ncMMĞİRI_7ĆTnxĞRnqV7Ťncn*IŤ!
    nRhcŃŤnIlncŸĞİ7İncŤĆnĞİĞh~cRŃIŤİjn~ĞhİĞŃŤ!nŃŤ<chŃcŸ*[nRV7nİc_7n)Ÿ@7MRn7<
    ŃŤM7İncnĆ7İŃ!ŤnRInh7ĆĞM7nRV7_nĞŤĆ7hncŸİI*ĞR7nb7İ~IRŃİ_jnŃRnŃİnRV7ŃhnhŃ!
    VRjnŃRnŃİnRV7ŃhnĆĞR[jnRInRVhIqnIllnİĞMVnĄI<7hŤ_7ŤRjncŤĆnRIn~hI<
    ŃĆ7nŤ7qnĄĞchĆİnlIhnRV7ŃhnlĞRĞh7nİ7MĞhŃR[Tn§§}ĞMVnVcİnŸ77ŤnRV7n~
```

cRŇ7ŤRnİĞll7hcŤM7nIlnRV7İ7n-I*IŤŇ7İĔncŤĆnİĞMVnŇİnŤIqnRV7nŤ7M7İİŇR[

nqVŇMVnMIŤİRhcŇŤİnRV7_nRInc*R7hnRV7ŇhnlIh_7hn}[İR7_İnIlnĄI<7

hŤ_7ŤRTn2V7nVŇİRIh[nIlnRV7n~h7İ7ŤRnUŃŤ!nIlnĄh7cRnxhŇRcŇŤn%Ą7Ih!7

nkkkŔnŇİncnVŇİRIh[nIlnh7~7cR7ĆnŇŤ@ĞhŇ7İncŤĆnĞİĞh~cRŇIŤİjnc**nVc<ŇŤ!

nŇŤnĆŇh7MRnIŸ@7MRnRV7n7İRcŸ*ŇİV_7ŤRnIlncŤncŸİI*ĞR7n2[hcŤŤ[nI<7hnRV7İ7n}

RcR7İTn2In~hI<7nRVŇİjn*7RnLcMRİnŸ7nİĞŸ_ŃRR7ĆnRIncnMcŤĆŃĆnqIh*ĆTn‘67

nVcİnh7lĞİ7ĆnVŇİnCİİ7ŤRnRIn

cqİjnRV7n_IİRnqVI*7İI_7ncŤĆnŤ7M7İİch[nlIhnRV7n~ĞŸ*ŇMn!IIĆTn‘67

nVcİnlIhŸŇĆĆ7ŤnVŇİnĄI<7hŤIhİnRIn~cİİn

cqİnIlnŇ__7ĆŇcR7ncŤĆn~h7İİŇŤ!nŇ_~IhRcŤM7jnĞŤ*7İİnİĞİ~7ŤĆ7ĆnŇŤnRV7ŇhnI~7hcRŇIŤnRŇ

**nVŇİnCİİ7ŤRnİVIĞ*ĆnŸ7nIŸRcŇŤ7ĆĔncŤĆnqV7ŤnİInİĞİ~7ŤĆ7ĆjnV7nVcİnĞRR7h*[nŤ7!*7

MR7ĆnRIncRR7ŤĆnRInRV7_Tn‘67nVcİnh7lĞİ7ĆnRIn~cİİnIRV7hn

cqİnlIhnRV7ncMMI__IĆcRŇIŤnIln*ch!7nĆŇİRhŇMRİnIln~7I~*7jnĞŤ*7İİnRVIİ7n~7I~*7nqIĞ*

Ćnh7*ŇŤ3ĞŇİVnRV7nhŇ!VRnIlnH7~h7İ7ŤRcRŇIŤnŇŤnRV7n

7!Ňİ*cRĞh7jncnhŇ!VRnŇŤ7İRŇ_cŸ*7nRInRV7_ncŤĆnlIh_ŇĆcŸ*7nRInR[hcŤRİnIŤ*[Tn‘67

nVcİnMc**7ĆnRI!7RV7hn*7!Ňİ*cRŇ<7nŸIĆŇ7İncRn~*cM7İnĞŤĞİĞc*jnĞŤMI_lIhRcŸ*7

jncŤĆnĆŇİRcŤRnlhI_nRV7nĆ7~IİŇRIh[nIlnRV7Ňhn~ĞŸ*ŇMnH7MIhĆİjnlIhnRV7nİI*7n~Ğh~

Iİ7nIlnlcRŇ!ĞŇŤ!nRV7_nŇŤRInMI_~*ŇcŤM7nqŇRVnVŇİn_7cİĞh7İTn‘67nVcİnĆŇİİI*<7ĆnH7

~h7İ7ŤRcRŇ<7n6IĞİ7İnh7~7cR7Ć*[jnlIhnI~~IİŇŤ!nqŇRVn_cŤ*[nlŇh_Ť7İİnVŇİnŇŤ<

cİŇIŤİnIŤnRV7nhŇ!VRİnIlnRV7n~7I~*7Tn‘67nVcİnh7lĞİ7ĆnlIhncn*IŤ!

nRŇ_7jnclR7hnİĞMVnĆŇİİI*ĞRŇIŤİjnRInMcĞİ7nIRV7hİnRInŸ7n7*7MR7ĆĔnqV7h7Ÿ[nRV7n

7!Ňİ*cRŇ<7n~Iq7hİjnŇŤMc~cŸ*7nIlnCŤŤŇVŇ*cRŇIŤjnVc<7nh7RĞhŤ7ĆnRInRV7n^7I~*7ncRn*ch

!7nlIhnRV7Ňhn7&7hMŇİ7ĔnRV7n}RcR7nh7_cŇŤŇŤ!nŇŤnRV7n_7cŤnRŇ_7n7&~Iİ7ĆnRInc**

nRV7nĆcŤ!7hİnIlnŇŤ<cİŇIŤnlhI_nqŇRVIĞRjncŤĆnMIŤ<Ğ*İŇIŤİnqŇRVŇŤTn‘67nVcİn7ŤĆ7c<

IĞh7ĆnRIn~h7<7ŤRnRV7n~I~Ğ*cRŇIŤnIlnRV7İ7n}RcR7İĔnlIhnRVcRn~Ğh~Iİ7nIŸİRhĞMRŇŤ!

nRV7n

cqİnlIhntcRĞhc*ŇŘcRŇIŤnIlnLIh7Ň!Ť7hİĔnh7lĞİŇŤ!nRIn~cİİnIRV7hİnRIn7ŤMIĞhc!7

nRV7Ńhn_Ň!hcRŃIŤİnVŃRV7hjncŤĆnhcŃİŃŤ!nRV7nMIŤĆŃRŃIŤİnIlnŤ7qnC~~hI~
hŃcRŃIŤİnIln

cŤĆİTn‘67nVcİnIŸİRhĞMR7ĆnRV7nCĆ_ŃŤŃİRhcRŃIŤnIln ĞİRŃM7jnŸ[nh7lĞİŃŤ!
nVŃİnCİİ7ŤRnRIn

cqİnlIhn7İRcŸ*ŃİVŃŤ!n ĞĆŃMŃch[n~Iq7hİTn‘67nVcİn_cĆ7n ĞĆ!7İnĆ7~7ŤĆ7ŤRnIŤnVŃİnQŃ**
nc*IŤ7jnlIhnRV7nR7ŤĞh7nIlnRV7ŃhnIllŃM7İjncŤĆnRV7nc_IĞŤRncŤĆn~c[
_7ŤRnIlnRV7Ńhnİc*chŃ7İTn‘67nVcİn7h7MR7Ćncn_Ğ*RŃRĞĆ7nIlnt7qn)
llŃM7İjncŤĆnİ7ŤRnVŃRV7hnİqch_İnIln)llŃM7hİnRInVchcİİnIĞhn~7I~*7
jncŤĆn7cRnIĞRnRV7ŃhnİĞŸİRcŤM7Tn‘67nVcİnu7~Rnc_IŤ!nĞİjnŃŤnRŃ_7İnIln~7cM7jn}
RcŤĆŃŤ!nCh_Ň7İnqŃRVIĞRnRV7nMIŤİ7ŤRnIlnIĞhn*7!Ńİ*cRĞh7İTn‘67
nVcİncll7MR7ĆnRInh7ŤĆ7hnRV7n?Ń*ŃRch[nŃŤĆ7~7ŤĆ7ŤRnIlncŤĆnİĞ~7hŃIhnRInRV7n-Ń<Ń*
n~Iq7hTn‘67nVcİnMI_ŸŃŤ7ĆnqŃRVnIRV7hİnRInİĞŸ@7MRnĞİnRIncn@ĞhŃİĆŃMRŃIŤnlIh7Ń!
ŤnRInIĞhnMIŤİRŃRĞRŃIŤncŤĆnĞŤcMuŤIq*7Ć!7ĆnŸ[nIĞhn*cqİĔn!Ň<ŃŤ!
nVŃİnCİİ7ŤRnRInRV7ŃhnCMRİnIln~h7R7ŤĆ7Ćn

7!Ńİ*cRŃIŤPn‘LIhnŞĞchR7hŃŤ!n*ch!7nŸIĆŃ7İnIlnch_7ĆnRhII~İnc_IŤ!nĞİPn‘LIhn~hIR7MRŃŤ
!nRV7_jnŸ[ncn_IMun2hŃc*jnlhI_n~ĞŤŃİV_7ŤRnlIhncŤ[n?ĞhĆ7hİnqVŃMVnRV7[nİVIĞ*
ĆnMI__ŃRnIŤnRV7nkŤVcŸŃRcŤRİnIlnRV7İ7n}RcR7İPn‘LIhnMĞRRŃŤ!
nIllnIĞhn2hcĆ7nqŃRVnc**n~chRİnIlnRV7nqIh*ĆPn‘LIhnŃ_~IİŃŤ!n2c&7
İnIŤnĞİnqŃRVIĞRnIĞhn-IŤİ7ŤRPn‘LIhnĆ7~hŃ<ŃŤ!nĞİjnŃŤn_cŤ[
nMcİ7İjnIlnRV7nŸ7Ť7lŃRİnIln2hŃc*nŸ[n Ğh[Pn‘LIhnRhcŤİ~IhRŃŤ!nĞİnŸ7[IŤĆn}7
cİnRInŸ7nRhŃ7ĆnlIhn~h7R7ŤĆ7ĆnIll7ŤM7İPn‘LIhncŸI*ŃİVŃŤ!nRV7nlh77n}[İR7_nIlnŮŤ
!*ŃİVn

cqİnŃŤncnŤ7Ń!VŸIĞhŃŤ!n^hI<ŃŤM7jn7İRcŸ*ŃİVŃŤ!nRV7h7ŃŤncŤnChŸŃRhch[n!I<7
hŤ_7ŤRjncŤĆn7Ť*ch!ŃŤ!nŃRİnxIĞŤĆchŃ7İnİIncİnRInh7ŤĆ7hnŃRncRnIŤM7ncŤn7&c_~*7
ncŤĆnlŃRnŃŤİRhĞ_7ŤRnlIhnŃŤRhIĆĞMŃŤ!nRV7nİc_7ncŸİI*ĞR7nhĞ*7nŃŤRInRV7İ7n-I*
IŤŃ7İPn‘LIhnRcuŃŤ!ncqc[nIĞhn-VchR7hİjncŸI*ŃİVŃŤ!nIĞhn_IİRn<c*ĞcŸ*7n

cqİjncŤĆnc*R7hŃŤ!nlĞŤĆc_7ŤRc**[nRV7nLIh_İnIlnIĞhnĄI<7hŤ_7ŤRİPn‘LIhnİĞİ~7ŤĆŃŤ!

nIĞhnIqŤn

7!ŃĪ*cRĞh7ĪjncŤĆnĆ7M*chŃŤ!nRV7_Ī7*<7ĪnŃŤ<7ĪR7ĆnqŃRVn~Iq7hnRIn*7!ŃĪ*

cR7nlIhnĞĪnŃŤnc**nMcĪ7ĪnqVcRĪI7<7hTn‘67nVcĪncŸĆŃMcR7ĆnĄI<7hŤ_7ŤRnV7h7jnŸ[nĆ7M

chŃŤ!nĞĪnIĞRnIlnVŃĪn^hIR7MRŃIŤncŤĆnqc!ŃŤ!nQchnc!cŃŤĪRnĞĪTn‘67nVcĪn~

ĞŤĆ7h7ĆnIĞhnĪ7cĪjnhc<c!7ĆnIĞhn-IcĪRĪjnŸĞhŤRnIĞhnRIqŤĪjncŤĆnĆ7ĪRhI[7ĆnRV7n*Ń<7

ĪnIlnIĞhn~7I~*7Tn‘67nŃĪncRnRVŃĪnRŃ_7nRhcŤĪ~IhRŃŤ!n*ch!7nCh_Ń7ĪnIlnlIh7Ń!Ťn?7

hM7ŤchŃ7ĪnRInMI_~*7cRnRV7nqIhuĪnIlnĆ7cRVjnĆ7ĪI*cRŃIŤncŤĆnR[hcŤŤ[jnc*h7cĆ[nŸ7!

ĞŤnqŃRVnMŃhMĞ_ĪRcŤM7ĪnIln-hĞ7*R[ncŤĆn~7hlŃĆ[nĪMchM7*[n~chc**7*7

ĆnŃŤnRV7n_IĪRnŸchŸchIĞĪnc!7ĪjncŤĆnRIRc**[nĞŤqIhRV[nRV7n67cĆnIlncnMŃ<Ń*

ŃŘ7ĆnŤcRŃIŤTn‘67nVcĪnMIŤĪRhcŃŤ7ĆnIĞhnl7**Iqn-ŃRŃŘ7ŤĪnRcu7Ťn-c~RŃ<7nIŤnRV7nVŃ!

Vn}7cĪnRInŸ7chnCh_Īnc!cŃŤĪRnRV7Ńhn-IĞŤRh[jnRInŸ7MI_7nRV7n7&7

MĞRŃIŤ7hĪnIlnRV7ŃhnlhŃ7ŤĆĪncŤĆnxh7RVh7ŤjnIhnRInlc**nRV7_Ī7*<7ĪnŸ[

nRV7Ńhn6cŤĆĪTn‘67nVcĪn7&MŃR7ĆnĆI_7ĪRŃMnŃŤĪĞhh7MRŃIŤĪnc_IŤ!

ĪRnĞĪjncŤĆnVcĪn7ŤĆ7c<IĞh7ĆnRInŸhŃŤ!

nIŤnRV7nŃŤVcŸŃRcŤRĪnIlnIĞhnlhIŤRŃ7hĪjnRV7n_7hMŃ*7ĪĪnkŤĆŃcŤn}c<c!7

ĪjnqVIĪ7nuŤIqŤnhĞ*7nIlnqchlch7jnŃĪncŤnĞŤĆŃĪRŃŤ!ĞŃĪV7ĆnĆ7ĪRhĞMRŃIŤnIlnc**nc!7

ĪjnĪ7&7ĪncŤĆnMIŤĆŃRŃIŤĪTn‘kŤn7<7h[nĪRc!7nIlnRV7Ī7n)~~h7ĪĪŃIŤĪnQ7nVc<7n^7

RŃRŃIŤ7ĆnlIhnH7Ćh7ĪĪnŃŤnRV7n_IĪRnVĞ_Ÿ*7nR7h_ĪPn)Ğhnh7~7cR7Ćn^7RŃRŃIŤĪnVc<7

nŸ77ŤncŤĪq7h7ĆnIŤ*[nŸ[nh7~7cR7ĆnŃŤ@Ğh[TnCn^

hŃŤM7nqVIĪ7nMVchcMR7hnŃĪnRVĞĪn_chu7ĆnŸ[n7<7h[ncMRnqVŃMVn_c[nĆ7lŃŤ7ncn2[

hcŤRjnŃĪnĞŤlŃRnRInŸ7nRV7nhĞ*7hnIlncnlh77n~7I~*7Tn‘tIhnVc<7nQ7nŸ77ŤnqcŤRŃŤ!

nŃŤncRR7ŤRŃIŤĪnRInIĞhnxhŃRŃĪVnŸh7RVh7ŤTnQ7nVc<7

nqchŤ7ĆnRV7_nlhI_nRŃ_7nRInRŃ_7nIlncRR7_~RĪnŸ[nRV7Ńhn*7!ŃĪ*cRĞh7nRIn7&

R7ŤĆncŤĞŤqchhcŤRcŸ*7n@ĞhŃĪĆŃMRŃIŤnI<7hnĞĪTnQ7nVc<7

nh7_ŃŤĆ7ĆnRV7_nIlnRV7nMŃhMĞ_ĪRcŤM7ĪnIlnIĞhn7_Ń!hcRŃIŤncŤĆnĪ7RR*7

_7ŤRnV7h7TnQ7nVc<7nc~~7c*7ĆnRInRV7ŃhnŤcRŃ<7n@ĞĪRŃM7ncŤĆn_c!ŤcŤŃ_ŃR[

jncŤĆnq7nVc<7nMIŤ@Ğh7ĆnRV7_nŸ[nRV7nRŃ7ĪnIlnIĞhnMI__IŤnuŃŤĆh7ĆnRInĆŃĪc<

```
    IqnRV7İ7nĞİĞh~cRŃIŤİjnqVŃMVjnqIĞ*ĆnŃŤ7<ŃRcŸ*[nŃŤR7hhĞ~
    RnIĞhnMIŤŤ7MRŃIŤİncŤĆnMIhh7İ~IŤĆ7ŤM7Tn2V7[nRIInVc<7nŸ77ŤnĆ7clnRInRV7n<
    IŃM7nIln@ĞİRŃM7ncŤĆnIlnMIŤİcŤ!ĞŃŤŃR[
    TnQ7n_ĞİRjnRV7h7lIh7jncM3ĞŃ7İM7nŃŤnRV7nŤ7M7İİŃR[jnqVŃMVnĆ7ŤIĞŤM7İnIĞhn}7~
    chcRŃIŤjncŤĆnVI*ĆnRV7_jncİnq7nVI*ĆnRV7nh7İRnIln_cŤuŃŤĆjnŮŤ7_Ń7İnŃŤnQchjnŃŤn^7
    cM7nLhŃ7ŤĆİTn‘Q7jnRV7h7lIh7jnRV7nH7~h7İ7ŤRcRŃ<7İnIlnRV7nĞŤŃR7Ćn}
    RcR7İnIlnC_7hŃMcjnŃŤnĄ7Ť7hc*n-IŤ!h7İİjnCİİ7_Ÿ*7Ćjnc~~7c*ŃŤ!nRInRV7n}Ğ~h7_7n
    ĞĆ!7nIlnRV7nqIh*ĆnlIhnRV7nh7MRŃRĞĆ7nIlnIĞhnŃŤR7ŤRŃIŤİjnĆIjnŃŤnRV7ntc_7jncŤĆnŸ
    [nRV7nCĞRVIhŃR[nIlnRV7n!IIĆn^7I~*7nIlnRV7İ7n-I*IŤŃ7İjnİI*7_Ť*[n~ĞŸ*
    ŃİVncŤĆnĆ7M*ch7jn2VcRnRV7İ7ngŤŃR7Ćn-I*IŤŃ7İnch7jncŤĆnIlnHŃ!VRnIĞ!
    VRnRInŸ7nLh77ncŤĆnkŤĆ7~7ŤĆ7ŤRn}RcR7İĖnRVcRnRV7[nch7nCŸİI*<7ĆnlhI_nc**nC**7!
    ŃcŤM7nRInRV7nxhŃRŃİVn-hIqŤjncŤĆnRVcRnc**n~I*ŃRŃMc*
    nMIŤŤ7MRŃIŤnŸ7Rq77ŤnRV7_ncŤĆnRV7n}RcR7nIlnĄh7cRnxhŃRcŃŤjnŃİncŤĆnIĞ!
    VRnRInŸ7nRIRc**[nĆŃİİI*<7ĆĖncŤĆnRVcRncİnLh77ncŤĆnkŤĆ7~7ŤĆ7ŤRn}RcR7İjnRV7[nVc
    <7nlĞ**n^Iq7hnRIn*7<[nQchjnMIŤM*ĞĆ7n^7cM7jnMIŤRhcMRnC**ŃcŤM7İjn7İRcŸ*ŃİVn-
    I__7hM7jncŤĆnRInĆInc**nIRV7hnCMRİncŤĆn2VŃŤ!İnqVŃMVnkŤĆ7~7ŤĆ7ŤRn}RcR7İn_c[
    nIlnhŃ!VRnĆITnCŤĆnlIhnRV7nİĞ~~IhRnIlnRVŃİnb7M*chcRŃIŤjnqŃRVncnlŃh_nh7*
    ŃcŤM7nIŤnRV7n~hIR7MRŃIŤnIlnĆŃ<ŃŤ7n^hI<ŃĆ7ŤM7jnq7n_ĞRĞc**[n~*7Ć!7
    nRIn7cMVnIRV7hnIĞhn
Ń<7İjnIĞhnLIhRĞŤ7İncŤĆnIĞhnİcMh7Ćn6IŤIhT‘
```

A.3 Multidigit digraph Affine Encyphered Cyphertext

```
&<:ĘX3ĎD!<UD.}5baWi}LDsGŃ{ĎDQ>:ĘĄYVD’PČrYiİŔĹY’ĘYiĹYU{%#ŔNkbi}ŐW,t|}QŸUDu}Ľ‘dY~}Š
    >UD!<UDŹ},3d3e{ŰDn{K‘WDĘ<5i:D8{FWFi‘ĘŐW]P ‘8PDW
DI3!<@{+}!<EbVD^Ę^Du}@{
YgŔUDŻŔ‘ĘCD!<UDŹ}=WŞYi}LD!<UDS{VP]X8PDWuY?tiboPUD^Ę^DĆkC{ŰDMPoPX}ĎDu}Ą5P3:<8
    PDWC885WDHNI&oP5bUD^Ę^DHNI&oP5b*ĚWD
```

}^D:Ęd3-ŸUD!<wŔVDyDhWqWzP2bĹY|W]P8P{D!<UDetŰĘX}7Yi}LDŃ{ !ŰĘ^DjW"G!bĹY8P8{ND!<Ž#
uYa}$Ÿ^DhWFŸibUD!<UDe{xYĹYĄ5P3:<X3wt4Ÿ8PDW

Du}8PDWuY?tiboPX}-FĆ

]WŢ<ZŸ^D!<ĹYUD>b;PmY8P{DzWuY4Ÿ2OQ>x‘:ĘqX8P8{NDXŸŰDŮWĎDibUDWbS{XW^DĆkC{9X8P8{ND!<Ž
#@{jWLWK‘:5 ‘Čr!D!<k3‘DwbS{u}‘DI3!<FiEbL{ŰĘ{GĆ{,3:Ę:r WUĆAE0PzX8P8{NDŻŔ‘ĘCD!<
ĹYUDibUDL3VWVDL3zWVP!D^Ę^D!<UD+GŞY[3NDHN7\ctĹ3ŐW

YDD3OA<oP8P{DaW GjW8PDWaW2bAE0PzXtew>Eb&Ř:ĘĄY@{jWX37Yd3ĞGXW^DŻŔ‘ĘCD7W’XĽ‘Eb->ŰĘCD
!<k3‘DaGMP,t:5EbWD<bİŘ8PDWFi‘ĘaWzPi}LD!<UDj}FWEĘ ‘VD3OA<oPĄ5DWŐWFW‘D^Ę!DŞ}ĆŔi
}LD

}FWEĘŮWzPČrYiİŔĹYĽ‘ĹY>bIid3FWi}LD!<ĹYUD:ĘŚYVD’PX3WD!<UDŰ3Ű<NDHN8PDWOŤ|}QŸUDu}@{
ŇPEbi}‘Du}@{Ů},3*<X3qX@{K‘8P{DŰĘMP’P;PUDŐW3D

}FWEĘŮWzPVDŤ{73ŤEX3ĄYŔN4GK‘oPX}ĎD‘ĘuYIi:DbbŰĘ}3QŸĹY@{K‘i}pE^Ę9ŽŰĘCD’PWDŹ}=
WŞYX3ĎDŘG:<ŔNkb0X@{WDu}8PDW

D*<XŸŰDaWwŔdŔ.YND,3)W[#8P{D"NVW]P8PDW!bŮyFNKP!D^Ę^DX{\tŰĘĹYĂFOŤŹGhWĹixXX3K‘
_WĄXĄ5ŘŸŰDt3]PoPUD!<oPtew>Eb&Ř:ĘĄY!Ÿ‘ĘCDĹYL{OŸdYDW^D*<4G]‘’ĘqPČrUD:<^ĘMW^Ds}‘
D,3Ű<ND^Ę^D>b^Ęm3:ĘNDe{xYĹYOD^Ę^D}iŐ}’‘ŰĘ"Ÿ!DXŸŰDW,|Wv3:ĘqWŢ<oP:D*<Ť5’X8P8{
NDŃ{ !ŰĘ^DibUD4}jWĽ‘dYŹ}aW^Du}uYŠNVWČXĄ5P3 WLWR3IY@{jWuYŠNVW^{OŸxX8P8{ĎDu}2
bAE0P8PDW@Y4ŸFWWD5#@{Ů},3*<ŰĘCD!<UDs}ĆŔWDu}Ą5P3:<8PDW!DibUD}i GMPİŔ ‘DDIGNDĘ
<:Ę@{!Ÿ‘ĘCD>bŔ3ĎDHN@{)GaWWD^Ę^DxY5bp{d3‘ĘzX,t5bŘGŰĘCDŰĘ:{v3:r[#8PDWuYŻŔUD\r2W
]PLWR3ĹiĹY@{Ľ‘ĹYAEĎDu}2b ‘IiUD!<wŔ{GK‘Eb@{#YZŸ;PUDĹWAtqPdY0XX3NDdY8PDW!
b2bAE0PVD’PX3WD!<k3‘DŮGŞ#VDu}8P*b:5i}ŹNuYIi:D

}FWEĘŮWzPVD^Ę^Du},tĞ}R3hW’ĘŤ5teC{’‘WDs}‘D!<k3‘DĚGĞGjWuYYi5b’PJF(O"yIi:D8{WDzW:
Ę8PDW,toP;WzPuYŠNVW^{ĹiUDHN8PDWaWiĽZŸ‘Ę;WfQ@{K‘uYIi:DdY’Ę:58PDW’ĘYiĹYm3Ş#Ą5P3
:<Fi‘ĘMP^{ŰĘWD!<wŔ8P{DXŸXW‘D!<k3‘Ds}ĆŔEbŮyTYXW@Yi}LD

}FWEĘŮWzPDDA<UDP3MPkb!DHN8PDW,tjWaWzPĂAŰĘCDHNtejWoP&Šv3L{ŰĘ/1mWkbMWzSĽSİDdY@{Ţ<
dYu}%#i}LDjW|WoP ‘X3]*5b;WWD^Ę^DxY5bp{d3‘ĘzX@{uŸŢ<u>ŰĘCDŰĘĽ‘!bYiND<r2W]
P8PDWLWMP:r,3*<ŮWzPi}LD^Ę@{#YZŸ;PUD3#^{iĘ!Dw>Eb8PDWaWŮyL{XWĂFZp{Dbbw>UD!<dYVD

```
 WNDj{]PWDzWuYĘr#3DP ‘8P{DyDe{K‘x‘Ą5kb]‘DD}\UD8{WDjWĔGaW^DP3WDLYaWzP8P{D4{
fYVD!<UD4}MPĄ5a} W~}ŮW@{K‘’ĘYiĹYU{%#ŔNkb8PDW,tĘr,3gDj}B‘DD}\UD8{WDs}!rx‘
hWĎDP3WD
}FWEĘkbWDu},t,YWD4{fYi}LDSŔŮWt3oPUD^Ę^DbbĹYm3ŤEX3wtkbL{ĹixX{GcŸĹYWDŘGAt:ĘhW^
DŰĘ8PDW!bi}|W^{d3‘Ę8PŘŸŰDP3WDLYaWzPuYa}$Ÿ^DzWi}fPŔ3ŐWmQ@{K‘Ą5DWĎD~}uYxY|WK‘ ‘
VDDWŢ<,Y{GDPEb[#’ĘeE W]P ‘8P{DoPXWK‘8P{D!<wŔDD}\UD8{WDjWĔGaW^Du},t,YWDqPDW‘D4
{fYŔNkb8PDW@{kiİŔ4}\{d3‘Ęi}LDŤ{pEUDt3MPv3]PWDHN,t|}QŸxX{GcŸĹYWD!<.YUD|Wet
WĄ54G]‘2b4ŸŰĘ"GdY:D!<UDv3Ű<NDHNUĆ?tjWaWzPoPX}ĎDŰĘ8PDWC8eEdYŤ{ĞGjWVDyDv3Ű<
NDŰĘĹYd3Ń{OŸUDu}8PDW
D^Ę^Ds}ĆŔx‘:r W8P{DŞ#^{zPWD‘Ę[#DD}\UD8{WDe{uŸ ‘8PŃEKPDW‘D WY36ŸoP->UDŮ}t3ĹY@{NDQŸ
}iĹY{G=GŘGXŸVD*ĘŐ}ZNkbL{OŸxX@{K‘Ľ‘dYL{zPŔNĞ}
D!<UDhWŹ}m3u}%#i}LD!<k3‘D+GOŸ5iUĆYikbŚYVDs}‘D!<UD~} W,t5bŹ}aWi}LDJ{d3|GŰĘCD!<
wŔX3zP{DŐ}wt,3^ĘqWĄ5’P:DP3WDŮW,Y5bĹYDD}\UD8{WDt3
YZŸFW^D
WbbĹY:ĘL{d3FW7\4GaWWDjW|WoP ‘[#VDs}‘DetŹ}m3ŤEĄ5’P:DŃ{cŸ!Db3ĆŔŐW
YŢ<dYX3Ă>,YX}7Yi}ĎD!<UDv3Ű<ĄYi}LD!<UD|Wet WDD}\UD8{WDjWĔGaW^Ds}‘DyD=}ŤE8PSŔxX@{
BPEbuYIi:Dt3
YZŸ;PX}7YVDu}Fi2GaWi}!<EbWDu}ČrUD4ŸYiXWmQĄ5DWjW5#8PDWC8eEdYŤ{d3FW,t:5EbzXX3Ĺict:r
 Wi}LD~ĘŽ3P3Ť{d3‘ĘVD8{FW2bKP5bŐW^Du}8PDWOŤ|}QŸUDoP!ŸibMWŔNkb8PDW!
bLW4WdidYdQ8PDWŮyL{XW2bwŔŔ3Ž3ŤEX3ĎD!<UDŮW^Ę8PSŔUDW,Ź}aW^Du}@{uŸ8PDWĽ‘^ĘMWŞYi}
LDŰĘ:{m3‘ĘŔNĞ}
DI3!<4GqX@{K‘Fi‘ĘuGIYX}7YĄ5’PP3-FĆ
dWŢ<,YLWK‘S{c}5b ‘8P{DbbQ>:ĘND!<UDŹ}+GŤ{d3‘Ęi}LD!<ĹYUDmPoPĹYODs}‘D!<oP,t5bŹ}aWi}#
Y>bIid3ŤE8PDWC885WDs}‘D"{ĞG^{,3Ű{d3‘Ęi}LDŞ}jWAEŐWŞYODjWĔGm3ŤE8P{Dp{
Yi}!<EbWDu}LWĹi4G^{MW8PDW!bdŔAE^{d3‘ĘWDP3!<EbVD^Ę^D^{dYŰĘCD!<UDŐ}K‘’PX}7Yi}
LDŐW3DČtbbetv3oPX}7Yi}LD4{K‘ĂFĆ
dWŢ<,Yi}#Y>bIiXW^D!<UD‘‘#3Ž3MP^{d3‘Ęi}LDĄGMP5ixXČr!DjWĔGm3ŤEŢ<dYcŻ
```

Y:ĘNDu}C885WDs}‘DĹYL{OŸdYP3ŤE=JŇ‘5i/{%#,t:5EbĂFĆ

dWŢ<,YdŔ@‘UDĄGnEĹYĽ‘?t:ĘhWzPi}ĎDP3WDi3uŸ@{=}ŐWVDs}‘D!<UDXW=GjWi}LD!<k3‘DHNb3qWzX@
{K‘8PDW@{4}*ĘND^Ę^Dp{CŔ:ĘNDHN8PDW!buYXŸib;WĂFĆ

dWŢ<,YLWjW]P ‘@{dŔ$Ÿd3ĞGhWi}LD.W3DhNb3qWzX@{K‘uY:ĘNDP3!<EbuY1{ĆŔWDHNŇİŹN5iEbWDu}Ţ
<ib,YWD4G‘D|Wet WVD^Ę^DS{ND4GND!<k3‘DŘG#YL{Ĺi~FĆ

dWŢ<,Y^!?tNDŻŔ‘ĘCDxYVDŰĘ8PSŔĹYi}LD|W}ixXŮyL{K‘ŰĘCDĽb#3ĹYĄ5’Pa};P8PDWFi‘ĘaWzPi}
LD4G‘D WY36ŸoP5bĹYDD}\UD8{WDFNVW]P ‘8P{DjWK‘Eb8PDWĘ/ŘŸ’Pib!DŰĘhW|WK‘:ĘNDHN@{K
‘uY/tEbX}‘Du}8PDWiĽ->ŘŸ,t:5EbDD}\UD8{WDŐ}NrŰĘ ‘Ą5’P:DqPDWŞY8P{DŘGI*YiNDxY8P{
DyDaGv3đ‘5id3‘ĘŔNkbk3(Ę8P{D4G‘DŐ}7Yd3ĞGd3‘Ę@{K‘{GĆ{Č!+}ŠŸ ‘MW^D5#i}5b!Ÿ85fQRE
->ŰĘCDP3WDLYaWzP8P{D!<k3‘DİiĄYi}LDbbKP:ĘhW^D@WY36ŸoPX}PZĆ

Ş}‘DBGibXWv3ŤE!ŸibMWČrB‘;WWDHN@{ĆŔ ‘8PĞ}etWDŻŔ‘ĘCDxYXDwnkb,tĞ}XW]PŰĘCD!<wŔVD5#@{
dŔ-i§D^b/{9XŔNĞ}

D+GŽ3*<ŮWzPŔNkb@{I#Ę/5bhWŞYĄ5P3:<8PDW!D*<4G]‘FiİŔ#3ND‘Ę8PDWzSW<:r’P^ĘĄYi}LD!<
ĹYUDmPoPĹYXDwnkbFi;Pd3ŤEi}ŹNi}5bZp^{hWĄ5’P:DXŸŰDp{VPWDHN8PDWĄ5kb]‘XDwnkbX3wt.
YŰĘCDl{4WWD‘Ę{GWDI3!<4GND4G‘D.}7Y:ĘŽZĆ

Ş}‘DhWbb->ŰĘCDxYVDŰĘdŔ^Ę!De{aWzXi}LD!<UDzWŐWb3ĄYi}LD^b/{ŰD5#=J5bmZĆ

Ş}‘D>b^ĘAtkbd3ŤE{GWDzWH}K‘ŮyS{WDu}ČrUD>b;W^Ds}‘DbbKP:ĘhW^DHNVWĹiĹYXDwnkb@{Ů},3*<
ŰĘCD!<UD<b_WŮyTYXW

DHNnwŤE,3*<C885WDŰĘ@{’Ęk3Ű<Ů}5bŰĘCDĆbw>ŰĘqWVDĹYL{OŸdYP3ŤE8PDWjWŰĘ@{ĎDĽbĘ3>bib!Dj}
FWEĘŮWzPVD^Ę^D:ĘŤ{pEŰĘCD’PWD7}*Ę\{v3ĹYuY{D,Y8P{DjWK‘EbX3NDoPi}ĹiUD^ĘLW({wt W@
{K‘ŔN’PX37Y>bgŔ:ĘNDs}‘DŰĘ>bB‘IiŰĘCD!<UDU{ŮW@{#YZŸ;PUDŹG WX3zP{D!<ĹYUD.}=}
Ž3ĹYXDwnkb8PŢ!ŰĘCD85>#i}5biĽ8{VPEbzX@{Ů},3*<ŰĘCD4G‘D4}MP>>XŸC{OŸUD4{fYVD^Ę^
DXŸXWv3ŤEŔN*Ę\{ŮWzPXŸ[#8PDW1nkb@Yi}LD4G‘D

}FWEĘŮWzP#ZĆ

Ş}‘DŘGAt:Ęt3ŤEi}5bi}ŸĘC8eEdYŤ{ĞGjWzX@{K‘Ľ‘YiŤ{v3ŤE8PDW@Y4ŸFWWDŰĘFWMP ‘Ą5’P:DŹ}=W‘
Du}!ŸeEdYŤ{XWŔNkb{GWDŰĘ@{uŸFi,YĹYĄ58{ĄYĚWFWĽFĆ

dWŢ<,Y@{7‘5ioP ‘tew>Eb&Ŕ:ĘNDDWjWVD5#Ľ‘YiŤ{v3ŤE{GWD4GNDHNŢ<dYOŤĞ}XW]PX}ĎD^Ę^D1{

```
    Y3ŤE#Uib@{A{ŰĘMP{GĂFĆ
dWŢ<,Y,tOGK‘Eb ‘i}5buYS{zX2bu>ĽE ‘i}5biĽy{MPzXČr5bzPi}5b8P:57YVD^Ę^DhWMPĞ}+W^D!<
    UD,3FWWDHNi}5b,t|}QŸ~FĆ
dWX3WDoP8PP3WDd3ŮW8P^{7YŹ}VPŰĘCDŤ{pEUDĽb#3ĹYi}LDs}jWAEĎD7Wdi:Ęib;WWDu}FiİŔQŸS{ND
    !<UDZ}|!WDHNĽ‘S{!<VDhW~}Ť{d3‘Ę@{K‘8PŚb^ĘI#VDXŸjW@‘!DzW|GĎDI3!<Fi!b G@YL{ĹiĹYi
    }LDwbOWŇP!D^Ę^D|W-Nx‘!D[iibqW[#,tibXŸ W W^DŰĘ8PDWdŔ.YNDn{!rib4GWDĽEĹYVD^Ę^Du}
    L{uŸ!D*ĘZ}VP-#8PDW7\S{^DHN@{Fi->ŘŸ9Ž ‘’ĘoPX}-FĆ
dWŢ<,YFi‘ĘMP^{ŰĘ ‘i}5bŔN4Ÿ=}3Dİ3d3"W7Y8PŢ!:ĘiĽctd3FWi}ĎD!<UDP3Ű<ŮyS{WDu}ČrS{‘
    DĽb@Y@{A{ŰĘMP8PDW!biĽ4GzP%#VDu}ČrYiİŔUD!<UDW,Yi;PX}ŐWŞYi}LD!<k3‘D<b;WK‘WD^Ę^
    DĂbKP*b:ĘVDkb8P{DJ{uŸ8PDW@Y4ŸFWWD5#8PDW!b7\^ĘŚYDD}\UD8{WDW,}3XW^DĚ}
    ŮWMP5iX37Y5bjW]PX}7Y@{4}ŤEMP{GzX@{K‘Ţ<,YLWK‘S{c}5b ‘8P{D‘bŰĘCD‘Ę8PDWX3W<:r’P^
    ĘĄYi}LD4G‘D<b‘Ęd3EbzX8PDWdŔEb}3 W
YzSK‘/{ĎDu{:{MWzXĄ5a}aW^!+}ŸĘ2b$ŸUDHNĄ5ibJ{jWVDdY@{ĎD*Ęt3MPŰĘ|GdYDW^DhWMPŹG]PX}
    ĎDHN@{uŸ@{MWzXuYW,ĹY@{K‘Fi‘Ęt3d3‘ĘĂFĆ
6ĘLWFW%#uYL{MWi}LD!<ĹYUDĚtbbĹYm3‘ĘWD]WŢ<u>UD Wd3d3‘Ę ‘ŔNkbUĆ ‘jW
YX3ĎD!<UD4}MPŢ<gŔOŸUDXWĆŔ#ZŇİ5b2b?tS{XW^D Wd3d3‘ĘWD8{FWČr_WĎD^ĘŰ5Eb ‘i}cŸ!D5#2b?
    tS{XW^DŰĘaG%#DDŹDĆbŰĘqWĄ5a}aWFi8{^{]PEbX3WD!<xYdŔib)W^D5#LWFW%#@{]PĄ5P3:<dŔ>#
    Ľ‘"NŰĘUDyD3#^{zPVDdY{GQN’P8P{DzW8PDW2b$ŸEbi}LDyD<b_W,t|}QŸ~FĆ
K}‘D8{FW#UUDzW:ĘĄ5^Ęd3ŤEX3ĎDoPXWzPX}7Y8P{D4G‘DĂb’PdY:D‘bKP*b:ĘDD]WŢ<u>UD1{EĘ ‘8
    PDW
D<bİŔ8PSŔUDu}8PSŔUDHN@{DPwŔhPWD5#8PDW!b!ŸeEdYŤ{ĞGjW8P{DW,XWK‘@{ĎD*Ę1{Pb^ĘL{
    OŸUDaGv3đ‘5id3‘Ęi}FW‘DxYDD]WŢ<u>UDjW#3K‘ ‘8PDW
DHN8PDWFi!b G@YL{ĹiĹYi}LD4G‘DwŔAE^{d3‘Ę@{K‘uYKP-ŸwŔ:ĘNDDWjWDD]WŢ<u>UDct|WXŸ ‘8P{D
    !<k3‘DĆ{d3FWŻ*xYd3qW@{K‘dŔĽEĆ{Ž3#3Ş#VD^Ę^D=WŢ<u>UDŐ}]*5b ‘8PDW
D5#8PDW8P;WWDHNi}5bFiİŔ4}ĎD53K‘jW^Du}Ľ‘dYu>:58PDWaW{GŘGJtoPX}7YVDĘ<5i]XĄ54G]‘
    X3ŐWR3L{OŸ!DŰĘXWPb/tND4G‘DŐ}iĘYid3‘ĘWD^Ę^DŐ}PbĹYŹ}K‘:ĘqWDDA<Ž#8P"}Ţ<u>UDzW:ĘĽ
    ‘S{LDu}8PDW>>Ś3qWi}LDaGMP5iUD^Ę^DHNFi‘ĘU{ŤE[3Ž3Ş#DD]WdŔxYqX8PDWjWs}jWVD}i"G;W
```

[iUDŰĘ8PDW’ĘYiĹYm3Ş#VDĘ<5i:DhW+}*ĘqWWD4G‘DĄWp{^{d3‘ĘVD^Ę^Da}]‘8PDW0X@{WD=WŢ<
ZŸ^D!<UDjWMPi}LDŃ{ !ŰĘĄXnwŐW#3ĹYX3ĎDQ{ČXX3ĎD W}iUD\b;WK‘ĂFĆ
]WVD!<Eb"NkbxX8PDWUĆ?tjWaWzPoP->ĹYi}LD!<UD*Ę’P ‘ŮyL{XWWDHNcŻŮWv3e{VDŰĘte:ĘEbXŸiĽ‘
Ę3bĹYzXcŻ
YwŔOŸ ‘VDct|WXŸŰĘCDu}8PDWŮy/tjWŮW=JŃ‘MWi}LD!<UDZ}?Ÿ^Ds}‘D!<UDjW]P’PŃ‘UDHNi}5bX3zP
:Ęd3‘ĘzXĽ‘đXX3ĎD!<UD"{ŮWVD^Ę^D5#8PDWcŻ;Pa}v3Ş#i}LD!<UDj}B‘OŤ|}
QŸUDHN8PDWaWiĽZŸ‘Ę;WzXuYZŸwŔcŸ!D+GOŸdY:D^Ę^DhWFŸibxXZp8{ND!<ĹYUDJĘ’P ‘iĽZŸ‘Ę;
WWDibxX@{K‘i}LDŰ3Ű<ND4GŰ<NDu}ČrUD\b_W@{K‘zSK‘?t:ĘhWzPŮyL{XWfQ8P8{ND!<Ž#@{jWcŻ
#YZŸFW^D<bİŔ@{uŸcŻuŸeE/{ĹiUDu}8PDW&Šv3d3*<iĽĞ}ŸĘVD^Ę^D!<oP@{uŸ,tZŸ’P5iXŸFi‘
ĘŐW]PX}ĎDzW
5_WĎD!<wŔ@{K‘8PDWŮyL{XWi}LDSbS{NDĂb’PŔ3’XX3WD^Ę^D4GŰ<NDu}ČrUDu}L{uŸ!Dt3
YZŸFWmQ@{K‘8P8{ND,Y1njWUD^Ę^D6ĘhW|WK‘:ĘNDmPoPĹYVD!<Ž#Ţ<u>UDĒGuŸOŤ:5Eb8P{D WĄ##
UibVDŐ}ĹiOGhWOŤS{qWVDŐ}zP^{]PcŻuŸ/{ĹiĹYVDĹYL{OŸdY:D.}/ŔEbqWVD^Ę^Du}Ľ‘{
DXŸŰDqPDW‘DİiĄY@{K‘ZpP3ŤEWDĘ<5i:D6ĘhW|WK‘:ĘNDmPoPĹYdŔ>#i}LDv3Ű<NDĒ}DD~Ę^Ds}‘D
!<UDŘG\tkbNDHN8PP3WDĹWFŸiboPX}’XĄ5’P:DyDb3ĆŔ2b4Ÿ/{ĹiUD‘Ę8PDW,tĞ}XW]PX}ĎDHNĽ
‘->ŰĘUDĆbw>x‘:ĘqWVD=WdŔ;PC{uŸ!DQŸ ‘MW8P{DS{:<i}!<Ebi}5bC8->ĹYVD4G‘DŞ}VP*ĘĹY@{
K‘i}5buY}ijW^DĄ}+}ĽF

A.4 Vector digraph Affine Encyphered Cyphertext

C V Vi3i3_Ĺ_Ĺ4Ń4Ń=F=FŠiŠijKjK^X^XĄ1Ą1t*t*wBwBŘ/Ř/-p-p7,7,%d%d Ž Ž ,w,w[d[
duĽuĽq4q4o*o*_Ĺ_Ĺt6t6Ő@Ő@h}h}]Ď]ĎŞŞŞŞpŢpŢ8J8JC#C#VpVpTETE’Ľ’Ľ|-|-Ż]Ż]ŠZŠZđ^đ
^X~X~YjYj?U?Urjrj_Ĺ_ĹŰđŰđPXPXwŤwŤ3939ĞŤĞŤTETEWĒWĒŃĆŃĆđ.đ.~9~9^Č^Č.p.pŐ-Ő-
Ć$Ć$kokoTETE,3,3Ÿ>Ÿ>ŮvŮv,đ,đđ^đ^.t.t^n^n\i\i % %i?i?oorMrMoĂoĂ*Ĺ*Ĺİ
İ
YNYNDÑDÑDDHXHXŹIŹI<Ÿ<Ÿ(i(i’Ń’Ń\!\!:%:%JEJEMSMSfyfyFXFXzPzPCwCwH^H^_Ĺ_Ĺ{B{
BgPgPkoko-§-§ntnt’3’3eŹeŹ"z"zZBZBĽ3Ľ3m8m8ŸŮŸŮppp,c,cĹ~Ĺ~
OsOsoĎoĎDŽDŽ1İ1İŘnŘnF

F

XaXaŇlŇlĆ-Ć-g"g"NZNZy y ~9~9R/R/XaXa}t}tUŤUŤb<b<>3>3 z z6D6DJŞJŞŚŮŚŮ

m

mPČPČUGUG]e]e8 8 m

m

Y'Y'$-$-3@3@XaXajEjErjrj,c,cŽŠŽŠ!=!=C/C/(@(@MŇMŇđ^đ^_Ĺ_Ĺ^r^rN[N[t^t^u=u=nFnFDUDUČ

,Č,oĎoĎEĹEĹssss^Ĺ^Ĺ$

$

3@3@eOeOYgYg§9§9LĹLĹ6Ő6Ő4X4XyoyojajaĽ]Ľ]K4K4.Ť.ŤŤ

Ť

viviĆqĆqŸ(Ÿ(ŘCŘC"f"f${${2j2jEŮEŮĂ-Ă-wŤwŤŹ_Ź_!%!%Z8Z8n?n?ŮwŮw\Ť\Ť/Ğ/Ğ@Š@ŠzRzR1}1}Ř

*Ŕ*ŸoŸo-z-zwBwBZ5Z5ŤcŤcMRMR,v,vwŤwŤ Ě Ě-y-ykTkTk'k'_Ĺ_Ĺ%§%§=N=NTETE,Ů,Ů[d[dX~

X~U^U^i-i-X~X~nVnVOeOeŇĽŇĽŻLŻL'n'n~Ř~Ř4l4lĽ{Ľ{Ŕ*Ŕ*ŸoŸog8g8zhzhR.R.

CWCW@Ż@ŻĽ3Ľ3W6W6$@$@ŔđŔđC[C[(?(?Ě§Ě§B\B\ Ę Ę Ť

Ť

ddŰđŰđŰBŰBX5X5bZbZŠ

Š

KoKoeŹeŹ'3'3ĘTĘTđĚđĚŹlŹlŸĄŸĄ2D2De§e§ĂŰĂŰĚ_Ě_TETE[%[%D>D>ĆnĆn U UE.E.>[>[8A8A

"

"YaYaŇĽŇĽf/f/ŐOŐOK[K[P,P,#,#,eOeOYgYg§9§9AjAj+Ż+ŻiwiwİnİnŞŘŞŘŇSŇSČNČN#^#^

tĘtĘŸ§Ÿ§Ğ

Ğ

g?g?IđIđcncn;Ż;ŻR^R^{B{B\İ\İđ^đ^iĎiĎ_z_zŰ$Ű$f0f0-Q-QKmKmTWTWđXđXČ7Č7;D;DMSMS?^?^F

^F^H^H^%z%z'Ů'Ů.Ő.ŐFcFcAKAKS3S3^ ^Ď\Ď\p&p&LĹLĹAKAK~ ~ I0I0XaXa8Č8Č$

$

}\}\tŐtŐŹUŹUSESEbBbB§o§oŤ

Ť

_Ĺ_ĹŽRŽRqSqS^ ^Ť8Ť8_c_cH^H^#Ž#ŽĄyĄyŐĂŐĂ§-§-@V@VŚoŚorjrj6_6__n_nXXZ;Z;đ^đ^~G~
GŇ4Ň4đFđFiBiBĹaĹa8n8n6L6LEŻEŻTETERjRjVzVzŹ§Ź§@Ÿ@ŸzĂzĂŹ’Ź’
aDaD8Č8Č§r§r0Ţ0ŢrMrMU+U+yCyCi)i)4l4l?Ż?ŻŃwŃwĄaĄat{t{TETE\l\l~Q~QkTkTk’k’
vivifŹfŹŹŻŹŻeŹeŹŸuŸu+S+S$@$@HŔHŔ!%!%&Ż&ŻađađFcFc~Ń~Ńn(n(qq6Z6ZqDqDĞ
Ğ
viviŽaŽa@T@TĎĂĎĂ@b@bĄŞĄŞĚ0Ě0pŢpŢgQgQvBvBĚoĚoS9S9|c|c1+1+ĂfĂfŠ^Š^TETEiziz,Z,Zm~m~
Ě§Ě§$@$@8;8; Y YTETE0}0}#^#^****koko]K]K[y[yL:L:rMrMoĎoĎQMQM3T3TJŞJŞŽRŽRGeGe%
đ%đŮĹŮĹ~o~o!%!%&Ż&Ż§Ť§Ťy-y-\6\6vđvđTETEHŚHŚĎ{Ď{aKaK\~\~dŠdŠl
l
~D~DCQCQĎmĎmgPgP? ?QjQjp3p3Ť^Ť^1}1}Ć|Ć|ŇPŇP_m_m/+/+$Ő$Ő\+\+iČiČŹuŹugygy}8}82(2(8
n8nĄ>Ą>fqfq+?+?DŇDŇpp Ă.Ă.§o§o&Ż&Ż1}1}UgUgi)i)4l4l4Ň4Ň!b!bL\L\TETEY
Y
$Ř$Ř B BŠŐŠŐ^Z^ZŤsŤs(I(I8
8
ĘDĘDDŽDŽS/S/KĂKĂoĎoĎk1k1Ţ@Ţ@);); ŚpŚp
Ĺ
Ĺ3o3on$n$T_T_S’S’s:s:[<[<ŻŢŻŢpapa_Ĺ_ĹPŢPŢVtVt(B(B-Ĺ-Ĺa6a6-p-p7,7,%d%dTETEy2y2G’G’
ŽŢŽŢ;Ň;ŇĄeĄeĚ[Ě[!2!2-W-WS’S’Ŕ*Ŕ*§<§<qSqS@K@K/U/UĽ(Ľ(4Ě4Ě
‘
‘P,P,đ^đ^@)@)AŠAŠ-§-§s3s3!Q!QMsMsŇ}Ň}J{J{MŻMŻI>I>TETE?a?ap@p@ŹuŹuĞŸĞŸ|İ|İ?K?KĎ{Ď{
FXFXzPzPH6H6$
$
r rĚ§Ě§ z zđ§đ§ŤPŤP%;%;Z8Z8
’
’Ţ^Ţ^}^}^+Ľ+ĽAjAjVpVpoĎoĎqĚqĚŻLŻLDDr§r§
Ĺ
Ĺ3o3on$n$T_T_i5i5x=x=?5?5

E

E2Ğ2Ğx~x~t8t8-"-"=đ=đŹuŹu1}1}ĘzĘzŚİŚİSASAĘDĘDLFLFb!b!M

M

^§^§XdXdŠ%Š%3K3K2đ2đŇSŇSŮ:Ů:#-#-ĆqĆq_Ĺ_ĹE§E§=N=NŞ>Ş>P(P(u u Ő-Ő-

Ć

Ć)k)k% % l l -o-oiUiUĎŤĎŤ-İ-İŠ(Š(UĎUĎeŹeŹ'3'3đ^đ^ŽRŽRNđNđŢŞŢŞŞ

Ş

OOOOŘĘŘĘ%c%c

Ĺ

Ĺ3o3oxOxON<N<ŐĚŐĚ)')'6|6|Ű8Ű8ĂzĂzđ6đ6=*=*ŚHŚHŹOŹO$Ť$ŤzŔzŔininŘ?Ř?s3s3IOIOĚ§Ě§'Ů'

ŮŹoŹoVĚVĚzĎzĎŢ9Ţ9*r*rŤ^Ť^1}1}vivišđIđIŽ_Ž_@9@9Ą5Ą5BkBk3T3TQŔQŔŹ/Ź/ŻaŻa=F=Fr-r-

bDbDŤ

Ť

rMrMŢgŢg#o#oImIm§w§wKŢKŢh]h]ĚİĚİŽĞŽĞ

Ľ

Ľ

Ĺ

Ĺ3o3oŇBŇBŃuŃu! !@)@)qwqwxŘxŘJ]J]Z;Z;Ć|Ć|ŇPŇPqŻqŻĞ

Ğ

5353AŠAŠKŢKŢrĽrĽŹŤŹŤĆhĆhnFnFp3p3SŹSŹELEL<Ÿ<Ÿ§Ť§Ť}Ţ}ŢQoQo1}1}_Ĺ_Ĺ3K3KUWUW

Ĺ

Ĺ3o3on$n$T_T_Ğ

Ğ

ĞŞĞŞgĄgĄŘĞŘĞ%Ŕ%Ŕ'Ě'Ě~A~AŇBŇBĆHĆH[Ĺ[Ĺ.c.c

B

B=F=FM

M

qSqS#^#^!-!-~R~R z zĈxĈxTETE$3$3ĽŽĽŽxŘxŘ('('<B<BF~F~VrVr:V:VČ>Č>* *xtxtt^t^

ŸŸŔĆŔĆ1§1§FcFcTETE,Û,Û[d[dĈ$Ĉ$Û#Û#%d%dTETEĽgĽgVrVr9(9(TETELJLJ-a-

a7l7lĹBĹBi3i3_Ĺ_ĹŽqŽqfqfq<8<8.Ő.ŐFcFcI§I§TETE5:5:QrQr1}1}9-9-teteVĚVĚ'Ľ'Ľn}n}

ŇĽŇĽLcLcČSČS\|\|4'4']9]9m8m8m'm'%N%NŘ%Ř%Ş;Ş;N<N<ŠŐŠŐ^X^X}t}tČcČcŤXŤXŹuŹu1}1},

c,clLlLİĞİĞm[m[3T3T~Ĺ~ĹŻaŻa=F=F%3%3MRMRgĄgĄ)k)kĹYĹYĞ

Ğ

Ű@Ű@Ę4Ę4~T~T

%

%dŃdŃ

r

rTbTboŚoŚP<P<gĄgĄC%C%uhuhM

M

qSqS6b6bŃ Ń \C\C-%-%P

P

VTVT*Ž*Ž%§%§ŠZŠZđ^đ^nĆnĆŤ

Ť

_Ĺ_Ĺlqlq1313<Ÿ<Ÿ>1>16Ă6Ă'Ę'ĘİsİsĹ[Ĺ1}1}a_a_&Ĺ&Ĺm'm'[H[H%3%3QnQn-t-t(4(4!d!d[y[

yŹuŹu1}1}YOYOhĄhĄ+.+.nnAŠAŠp3p3ŸjŸjR^R^

1

1ĹYĹYĞ

Ğ

EzEz

o

o*b*b§Ž§Ž{~{~GŻGŻŘCŘC&Ĺ&Ĺm'm'İŰİŰŘ<Ř<ĂnĂnZ5Z5FnFnYŃYŃĄİĄİS'S'\G\Gf*f*k/k/Ğ

Ğ

_Ĺ_ĹĹdĹdŸ6Ÿ6wŤwŤJ§J§O/O/~g~gĚ§Ě§TETEĂŐĂŐ"^"^đ^đ^

Ň

ŇR^R^H*H*-%-%sŚsŚKkKk&Ĺ&Ĺm'm'%N%NA§A§A!A!cwcw4X4XŠVŠV~1~1l~l~ĎDĎDgygy<s<sR^R^%§%§

'Ĕ'ĔW@W@Ÿ>Ÿ>+s+sbWbWqSqSA:A:x=x=lĞlĞ3K3K!Ŕ!Ŕđ^đ^^#^# O OvivilululuVJVJ_Č_Čl§l§

§ §7M7MjEjErjrj&A&A

Ć

Ći6i61}1}tNtN/;/;)_)_Ť

Ť

Ŕ_Ŕ_gPgPŮOŮO^,^,2j2jEŮEŮĂ-Ă-H%H%F)F)bČbČ^n^n

Ľ

Ľ

Ĺ

Ĺ3o3oMkMkVpVpoĎoĎo[o[Ę^Ę^ĹDĹDl8l8P.P.a[a[%;%;R^R^;Ŕ;Ŕ<s<sŠRŠRv=v=eŽeŹRJRJvv('('

F"F"

Ĺ

Ĺ3o3oŤ[Ť[ffKŢKŢ3l3lvĆvĆoĎoĎVAVAfqfqŽEŽEđ0đ0JŸJŸo7o7ŹuŹu%d%dđeđeoĎoĎlĞlĞ}(}(4

X4XŹuŹuĚ§Ě§8 8 q(q(ŽxŽxŸ§Ÿ§ŸBŸB^Ű^Ű

<

<IWIWŤ

Ť

S'S's:s:[<[<DDJŞJŞ-Ń-Ń1}1}§Ć§ĆMŻMŻĂIĂIbČbČĽ3Ľ3İŇİŇg

g

JĚJĚ|H|Ho9o9Ů#Ů#cjcjhŻhŻ;Ŕ;Ŕ%Ŕ%ŔŐ'Ő'2h2hjajaK[K[9Ž9Ž|Ž|ŽŠŠŠŠtztz$$>W>Wfdfdp~p

~&^&^ŰoŰoVĚVĚeyeyiŘiŘ&Ż&Ż%d%dPlPlİ-İ-bBbB^ ^Ť8Ť8_c_cImImĆCĆCa<a<5Ű5ŰTETE![![

2 2ŃyŃy1}1},c,clLlLİĞİĞ9Ž9Ž)Ţ)ŢQ)Q)ŇDŇD]]v>v>:0:0KŢKŢ0,0,TuTu1414H*H*đ^đ^

@đ@đ9Ž9Žc+c+MŞMŞŤ

Ť

l l ŘĘŘĘŻ{Ż{PČPČ O OlĞlĞ](](xhxhnQnQvĽvĽEDEDŤ

Ť

&A&Aq§q§@Š@ŠŢŢŹWŹWwŤwŤaqaqf8f8ŽxŽxmŐmŐ"C"C]e]eZxZxnQnQiNiNT

T

k{k{FZFZgQgQŽQŽQFnFnqSqS#^#^

Ż

ŻZkZkĄ>Ą>V

V

)o)oSKSKzĂzĂ9Ž9ŽkckcnVnVŤ&Ť& $ $ Ę ĘS *S*%đ%đ:^:^2f2f§o§o,3,3t.t.O!O!dOdO/{/{Ť

Ť

A A GwGw$k$kiwiwŹRŹRgĄgĄDD}g}g'"'"}-}-P.P./*/*ŸoŸog8g8đ^đ^!W!W+/+/rĎrĎ]K]K

!7!7&*&*+İ+İ6Ė6ĖsQsQ_Ő_ŐĚŃĚŃ§/§/-%-%QKQKŮgŮgĹxĹxinin

Ź

ŹasasJ9J9*k*kOđOđGŽGŽ

C

C $ $§1§1 -Ĺ-Ĺi)i)4l4lZ\Z\ŢgŢgDD Ě Ě-|-|ŠjŠj9Ž9Žx~x~F;F;?o?os2s2'U'UE?E?<B<

BbBbB§o§oŤ

Ť

lĞlĞPŢPŢ?F?FŐĞŐĞ)u)upapađ^đ^QMQM Ć ĆEXEXđTđT}X}Xe§e§oUoUk9k91}1}lĞlĞŔ*Ŕ*ŸoŸoC~C~

nQnQluluLNLNŰ"Ű"%[%[]?]?$3$3ĽŽĽŽ@T@Tgygy+<+<5j5j Ć ĆTETE|q|qIkIk5c5c&p&

pKŢKŢrŞrŞTŮTŮmQmQ% % |Ř|Ř%d%dŻ{Ż{}g}g"{"{Y Y $Ű$ŰjZjZaeael§l§ Ş

ŞMoMo8y8yŻŮŻŮL\L\GIGI(9(9-o-oČoČo5j5j Ć ĆŻ{Ż{ O OT~T~ĆĄĆĄŠŠŠŠSSSSk>k>b1b1ĚRĚR

=c=c2X2XxnxnMYMYl§l§ Ş Ş I I_Ő_Ős-s-wĚwĚ8k8ke$e$ŸŢŸŢ%[%[

f

fJpJp1 1 Ă-Ă-V^V^Y Y ađađp)p)

W

W2?2?_Ĺ_ĹPŘPŘŸ>Ÿ>%[%[,w,w

B

Bl§l§k'k'zlzlS'S'ŐŇŐŇ\I\I\X\X§j§j3Ţ3ŢQĄQĄŔ_Ŕ_oĽoĽ$Ň$Ň

r

rŇMŇM3Č3ČKkKkqSqS~r~rz}z}}t}t9§9§MŢMŢXŽXŽŰ+Ű+X)X)kďkďŹuŹuĚ§Ě§anan)7)7#1#1İGİGE1E1

\w\wPČPČ=W=W{.{.JĘJĘoĹoĹ3j3jŽSŽSĆnĆn(Y(Y1N1NA#A#/Ş/Şi;i;Ť:Ť:<Ň<ŇI>I>TETE‘Ő‘ŐŘ

}Ř}ČpČpĘkĘk)L)LaďaďFcFcm_m_

>

>(Q(QQ_Q_-Ğ-Ğ>c>c;Ŕ;Ŕ9;9;&N&NŤ&Ť&Ľ3Ľ3m8m8m’m’g g VNVNŞEŞEs-s-$Ă$ĂKgKgQYQY‘ ‘ ŮUŮU

^+^+/%/%xŘxŘĄİĄİ_Ĺ_ĹS-S-FnFnqSqSBwBw;b;be*e*|Ţ|ŢŤŇŤŇ-%-%vZvZ

Ą

Ą.c.cĂ)Ă)z;z;_Ĺ_ĹQSQSmsmsTbTb1}1}vivi}V}V"K"K$Ő$ŐOfOf Ž ŽĆ>Ć>x~x~?=?=

qdqdtŰtŰŹUŹUSESETETEFRFRk9k9;Ň;ŇĄeĄeBŸBŸO8O8&#&#(Ŕ(ŔĽBĽB{W{WA§A§[Ĺ[ĹĂŐĂŐUmUmŔ

’Ŕ’Ě§Ě§ § §Ţ;Ţ;cbcbhBhBoĎoĎGGTFTFTETE!;!; 2 2ŇyŇy1}1}1Ğ1Ğw4w4ŚqŚqtŤtŤhdhd

?7?7$&$&%w%wVŃVŃŮdŮdŠ’Š’VtVtAbAb#§#§7z7zŠ^Š^=C=CjhjhŇ{Ň{I{I{Ź-Ź-DzDz

’^’^=/=/[-[-ĄŔĄŔŐ-Ő-;Ŕ;ŔdIdIŠ’Š’OMOM=*=*LcLc;§;§ŹuŹu&Ĺ&ĹYKYK6x6xL*L*ĄĘĄĘwŤwŤ

Ě Ě2>2>#^#^*Ž*ŽŸŮŸŮXzXz|P|PŹuŹu^ ^Z,Z,Ś

Ś

;R;R§Ť§Ť5757 ‘ ‘>[>[5y5y{m{m%{%{ŞďŞďy y hYhY@,@,*Ž*Ž Ÿ Ÿ >[>[k>k>$Č$Čs-s-

ŹŤŹŤ5M5MsisiyoyomCmCČČČČ,.,.GwGwVtVtIŽIŽ8$8$ĹNĹNk’k’L#L#Ţ=Ţ=,_,_bHbH6x6xWcWc

^ ^{I{I^_^_ĘxĘx,Ř,Ř4Ň4Ňg8g8=^=^

ď

ďoĎoĎ#s#sC!C!Ć6Ć61}1}ĹFĹFŰ?Ű?b!b! ((Ŕ

Ŕ

Ÿ Ÿ7q7qkQkQA_A_f"f"i3i3qĚqĚjČjČŮUŮUŢ5Ţ5UFUF-Ů-Ů#&#& Ž Ž6Ę6ĘŸŮŸŮXzXz6Ź6ŹN<N<’3’3Ÿ

[Ÿ[fqfqksksfqfqŠ^Š^ŚĞŚĞRŻRŻSESETETE"V"V% % 1 1 ssss.X.X9{9{I{I{Ŕ#Ŕ#G|G|iwiw

!!!!ĄkĄkŚŠŚŠ

%

%/Ğ/ĞZĽZĽŸŮŸŮXzXz0404-0-0TETE:^:^TETE=W=W{.{.JĘJĘoĹoĹH%H%

Ĺ5Ĺ5uďuďŹuŹuĚ§Ě§SiSiĆiĆiR^R^DDP5P5=ď=ď\‘\‘KoKoN<N<DDJŞJŞBŤBŤQ9Q9

B

B;h;hČ]Č]eŐeŐ,Ř,ŘŹŻŹŻđ^đ^8Ů8ŮXzXz&Ğ&ĞhBhBTETEueueTETEn)n)Ÿ>Ÿ>%[%[~r~r"m"m|Ğ|Ğ/O/

OuĽuĽ6Ż6ŻLĄLĄ^Ĺ^Ĺ&Ĺ&ĹyDyD[Ĺ[Ĺc c 6-6-FFFF<Y<YĘxĘxiwiwRŰRŰSiSi*,*,lĞlĞ"k"k}R}R

*Ž*Žđ^đ^Ę_Ę_ŔHŔHc*c*ĹGĹGFyFy@{@{=đ=đaqaqĘ2Ę2r+r+eŹeŹ?Ě?ĚQTQT1}1}yOyOtUtUđ^đ^

yAyAI3I3tbtbR\R\P5P5|f|fŹ$Ź$DD"

"

LKLKg g fufu

Ć

ĆG=G=-p-pŃ,Ń,c c BĄBĄ~N~NČ7Č7)u)u<e<e‘$‘$ŹuŹuađađŐ3Ő3Q,Q,

Ż_Ż_KoKo8Ů8Ůpp_Ĺ_Ĺa_a_1}1}?^?^:

:

t4t4mZmZi3i3Ÿ’Ÿ’i3i3TNTN\Ę\Ę"K"K&Ĺ&ĹŸ?Ÿ?M

M

đ4đ4n4n4cĄcĄAaAaĎ{Ď{""""oĹoĹwŤwŤ&~&~4P4PLJLJ-K-K[o[oyXyXDcDcW]W]:Ď:Ď++++‘l‘lĂ)Ă)

ŸjŸjn[n[Z8Z8yYyY

"

"5§5§TETEwewe4P4P§Ž§ŽĄĘĄĘwŤwŤ9§9§Ń Ń Ğ

Ğ

_Ĺ_ĹA§A§TŘTŘŠ^Š^%[%[ŢgŢgi?i?đFđF#@#@i3i3_Ĺ_Ĺptptađađ(U(UTETE+Ń+

ŃhŐhŐ2b2bwŤwŤŹ_Ź_dXdXb!b!Š^Š^TETEiziz,Z,Z\G\G_Ľ_Ľ&Ń&Ń!9!9

o

o

^

^p)p)ĂĘĂĘŐŤŐŤMSMS,c,cf~f~4P4P§9§9A/A/ r raŹaŹ@Ŕ@Ŕb%b%ĎŤĎŤĘCĘCŰcŰc#^#^|+|+Ě§Ě§%w%

wFnFnYŃYŃLJLJĎ.Ď.XaXa}t}tUŤUŤĘ8Ę8i)i)CŘCŘŸ[Ÿ[I§I§I?I?wZwZ-%-%eŹeŹsqsqvŃvŃ5

/5/7_7_ađađ-t-tmQmQuĂuĂJŸJŸ@1@1]K]K-_-_Ő-Ő-HNHNRgRgrMrMĚ§Ě§TETELJLJbVbVŻzŻz

^#^#a2a2BoBok’k’đ^đ^ĘCĘCŰcŰc#^#^‘N‘NOtOtŇBŇBŃuŃu8Š8ŠŻLŻLMSMS$Ğ$ĞI3I3đ^đ^A[A

```
    [%;%;R^R^ŐpŐpVAVAfyfy=đ=đ9Ś9ŚŢ‘Ţ‘ŘCŘCoĎoĎ,h,h=c=ct?t?nnnnŚŕŚŕ?|?|t?t?")")
    WQWQNMNMzĂzĂ$k$kiwiwĆqĆqĽ.Ľ.CSCSađađFcFc/Ť/Ťf0f0?5?5*+*+hŽhŽ;,;,*b*b&Ř&ŘBĄBĄA
    [A[%;%;R^R^ŐpŐp1V1Vs2s2"%"%ĎŤĎŤU]U]~^~^Ğ
Ğ
_Ĺ_Ĺ|O|OTŽTŽH*H*S'S'#w#wŇĆŇĆĆYĆYKŢKŢŻ@Ż@ĆhĆhŮ:Ů:3V3V
&
&TETE~Ń~ŃŇ?Ń?Ő-Ő-XŽXŽZ(Z(‘n‘nĆŐĆŐ3737FŢFŢ[?[?ŠoŠoJŸJŸsAsAoĎoĎ\Ğ\ĞU}U}+!+!~v~
    vMUMUfĚfĚ K KWŽWŽ=*=*@Ŕ@ŔwĚwĚ9f9fN3N3ak
```

A.5 Vector 4-graph Affine Encyphered Cyphertext

```
 V8Gi322_ĹĂ24Ńs!=FKKŠi.NjK\m^X2ŢĄ1">t*_cwB6ŞŘ/:}-p>o7,7W%d66Ž7&,w=^[d ~uĽ^đq4XĞo
     *~F_ĹĂ2t6ĄŠŐ@rPh}(L]ĎŐ<ŞŞŃgpŢ&\8JUPC#<uVp~cTEvÑ'Ľ§}|-^0Ż]b ŠZ[[đ^'GX~EVYjX!?
     U%ŹrjĘ?_ĹĂ2Űđp8PX§‘wŤcn39ĚAĞŤ/RTEvŇWĚStŇĆ4Yđ.X1~9;
^Č
Ą.p
ŃŐ-B]Ć$'fko§dTEvŇ,3mnŸ>ZoŮvŰz,đD
đ^'G.tqĄ^nde\igK %d[i?*Żo$ĎZrMWjoĂ$"*ĹŽŸİ
qvYN?EDŇŹID$-]HX,]ŹIĽĄ<Ÿ7Ď(iĞĹ'ŃU~\!4Ť:%Y'JEl
MS‘Ffy;
FXSRzPm?Cw-@H^<A_ĹĂ2{BŸMgPĚUko§d-§Š‘ntĚg'3m0eŹŐđ"zYNZBt1Ľ3-Em83:ŸŮŐ§pŚu,cĽzĹ~g'
    OsB§oĎ.RDŽ}Ď1İĘ+ŘnxmF
ČcXa*,ŃlŸ:Ć-/2g"nĂNZF5y 1Ź~9;
R/n!Xa*,}tŕpUŤiAb<S$>3§  zgg6DqČJŞXMŚŮcg
mDĆPČfŕUG)Ď]eg$8 1Bm
[tY'bW$-Ńz3@&2Xa*,jEKRrjĘ?,cĽzŽŠ$Š!=t-C/(p(@İFMŃl$đ^'G_ĹĂ2^r+jN[Žat^?qu=@KnF8.DU;
     Č,8joĎ.REĹpQssPŇ^ĹŠŤ$
+:3@&2eOČĂYgŤ7§9(ŃLĹ-"6Ő;z4Xŕ"yoŇbjaKsĽ]-_K4_V.Ť
```

uŤ

*uviŇ

ĆqĹĘŸ(mYŘCcE"f%-${j72j)NEŮŚŘĂ-X.wŤcnŹ_q5!%x Z8

Ŕn?FŚŮw>#\Ť=z/ĞnL@ŠŇĄzR/X1}pđŘ*UŮŸoŹH-zAvwB6ŞZ5<

ŤcŽEMR_=,v9pwŤcn ĚEG-yŸkkThŠk'ĞL_ĹĂ2%§gr=NT&TEvŇ,Ůİđ[d ~X~EVU^Š"i-<vX~EVnVUŘ0e|9
ŇĽ"XŻLGĘ'n$g~ŘŠŽ4lĘaĽ{s.Ŕ*UŮŸoŹHg8gđzh(ŤR.y|CWHv@ŻpČĽ3-EW6Ř $@L\ŔđŚĎC[qĚ(?%
FĔ§6BB\ađĘ jQŤ

*ud$<IŰđp8ŰBmPX59ŇbZCĄŠ

TsKo-|eŹŐđ'3m0ĘTfSđĚŢđŹ1AiŸĄĹ;2DvĹe§Ĺ

ĂŰH_Ĕ_ŃnTEvŇ[% 'D>1)Ćn'} Uđ@E.n>>[}Ş8A,X

"onYaŤ_ŇĽ_8f/dxŐ0=;K[Ż(P,~Ţ#,xJeOČĂYgŤ7§9(ŃAj0R+ŻŘ_iw

6İn§ŚŞŘ^.ŇSN_ČN$6#^2*tĘ,ŞŸ§ZŮĞ

ŸĂg?KĹIđ~jcnmp;Ż1mR^§C{BŸM\İ{Sđ^'GiĎhŔ_zV4Ű$&wf0mČ-QdŮKm7>TWF3đX1ŃČ7[#;DŔVMS'F?^
MuF^w-H^<A%z-y'ŮAŐ.Ő,pFcT(AKe$S32| ^%jĎ\wZp&ŢPLĹ-"AKe$~ AtI0T Xa*,8ČŸN$

+:}\{\tŐvxŹU>ŇSEF

bBŻr§oŮ'Ť

*u_ĹĂ2ŽRo8qSza^%jŤ8

Q_c9ĹH^<A#ŽŽ§ĄyŞ_ŐĂ/3§-Xd@Vx'Śo ĘrjĘ?6_Śf_nx\ XMĆZ;Śyđ^'G~GKĞŇ4;VđF67iBfFĹaĄ58nT
/6Lu|EŻ^7TEvŇRjvdVzBMŹ§sJ@ŸArzĂĹ^Ź'ĹFaD.78ČŸN§rN*0ŢUŮrMWjU+J

yCĆŻi)%v4lĘa?ŻŻ'Ńww'ĄadŮt{ĔrTEvŇ\lMŔ~QŘĄkThŠk'ĞLviŇ

fŹ"ŽŽŻh:eŹŐđŸuŚđ+Sb.$@L\HŔ?Ž!% x &Ż uađ.rFcT(~ŃŽĹn(Cp qŞJ6Z:tqDvŸĞ

ŸĂviŇ

ŽaD>@TIvĎĂ<<@bD5ĄŞj_Ĕ0^gpŢ&\gQJ-vBG;Ĕo@ĘS9ĄU|c$Ő1+Ĺ_ĂfŮHŠ^.mTEvŇizĞ+,ZVKm~.[
Ĕ§6B$@L\8;M^ Y=~TEvŇ0}Ůj#^>Ż**q

ko§d]KĞX[yb5L:vMrMWjoĎ.RQM4Ň3TŻĆJŞXMŽRo8Ge8K%đŇYŮĹ=M~oK,!% x &Ż u§ŤŞ&y-Pw\6
ŠWvđ69TEvŇHŚ ŸĎ{x?aKĽ|\~"[dŠS^l

/+~D§TCQ4?Ďm+mgPĄ' ?{0QjŃ!p3E6Ť^-Z1}pđĆ|#DŇP

```
u_mp"/+GM$ŐĎ3\+1ZiČr+ŹuU‘gy(!}8wH2(ŐĄ8nT/Ą>Ővfq4u+?15DŇŹIpŔ~Ă.t[§oŮ‘&Żu1}
    pđUgŰŮi)%v41Ęa4ŇA^!b>LL\BŻTEvŇY
7+$ŘWRBS=ŠŐgĚ^ZlkŤsMB(Ið 8
ŇĘD‘-DŽ}ĎS/GUKĂŸ
oĎ.Rk1{AŢ@ph);gĂŚpĚ4
Ĺ\,3oOtn$)ŘT_Š+S’WDs:J-[<W&ŻŢ epaŘc_ĹĂ2PŢ;3VtWz(Bx‘-Ĺ%1a6Ţv-p>o7,7W%d66TEvŇy26FG
    ’/ĂŽŢk!;ŇIrĄe87Ě[Ňe!2Űp-WŰ&S’WDŔ*UŮ§<R$qSza@KEv/UĹAĽ(Ąs4Ě’‘
‘A9P,~Ţđ^’G@)kBAŠe/-§Ś‘s3Z)!Qc6MsbJŇ}3CJ{ŚŹMŻNŻI>f
TEvŇ?a4Gp@ĚŽŹuU‘ĞŸlĘ|İ&x?KckĎ{x?FXSRzPm?H61$$
?\ rŚ Ě§6B zY
đ§Ğ|ŤP|\%;ZŽZ8
Ŕ
’(FŢ^7I}^bZ+ĽVzAjoĚVp~coĎ.RqĚ<[ŻLGĘD$-]r§X-
Ĺ\,3oOtn$)ŘT_Š+i5B>x=Kg?5 Ě
EŐ
2Ğ6Żx~ŔĽt8Cđ-"B%=đsİŹuU‘1}pđĘzK&ŚİŇ/SAzuĘD‘-LF71b! EM
x-^§1XXdŢ9Š%Ľm3KONŇ2đ
ŇSN_Ů:fj#-C!Ćq‘Ź_ĹĂ2E§’R=NT&Ş>Š
P(V:u (xŐ-B]
Ć/Ů)kŤ&% Ę_l 7Ž-o‘ĎiUŠOĎŤM5-İŚŹŠ(ĎUĎp|eŹŐđ’3m0đ^’GŽRo8Nđ ŢŢ§tBŞ
ŘuOOz‘ŘĘr/%c:&
Ĺ\,3oOtxOŽCN<}ŮŐĚ9U)’BJ6|zxŐ8Ć;ĂzŘ!đ6MĂ=*#EŚHŮĹŹOVŞ$Ť-"zŔŞuinĆZŘ?GQs3Z)IOTĚ§6B‘
    ŮAŐŹonNVĚPdzĎ9oŢ9@O*rxPŤ^-Z1}pđviŇ
đInrŽ_
Ň@9{RĄ5‘9Bk6$3TŻĆQŔ=
Ź/(bŻa<{=FKKr-ŘJbD2xŤ
*urMWjŢgnŠ#oqŞIm4Ů§wUmKŢgDh]>ĚİĚ3ŽĞC;
```

```
Ľđ)
Ĺ\,3o0tŇBx0ŇuĆ1 !ŤT@)kBqwŠ[xŘĄFJ]qŠZ;ŚyĆ|#DŇP
uqŻŸĆĞ
ŸĂ53đ]AŠe/KŢgDrĽ{vŹŤ|xĆh)<nF8.p3E6SŹTBEL^Ż<Ÿ7Ď§Ť3&}Ţz+QoV}1}pđ_ĹĂ23K0ŇUW&^
Ĺ\,3o0tn$)ŘT_Š+Ğ
ŸĂĞŞ^sgĄ*ĎŘĞKG%Ŕ^2'ĚŸK~AĹ
ŇBx0ĆHnG[ĹK%.c$
B~Ď=FKKM
VbqSza#^2*!-7i~Rmj zY
Ćx1(TEvŇ$3Ą6ĽŽŮyxŘĄF('-R<Bv9F~ŃzVr\U:VuJČ>2(* Č `xtSzt^ŻY ŸAbŘĆ~*1§gFcT(TEvŇ,ŮİđĽ[
    d ~Ć$
ŻŮ#İ]%d66TEvŇĽg6WVr;Ć9(Ő+TEvŇLJ5Ž-a^J71F[ĹBČi322_ĹĂ2Žq
;fq§đ<8Ži.Ő,pFcT(I§,^TEvŇ5::ĘQrQ[1}pđ9- [te[)VĚPd'Ľ§}n}Ź^ŇĽ"XLcdnČStĚ\| x4 `0D]9
    ŇŻm83:m'hf%NĞIŘ%ŞXŞ;AŰN<}ŮŠŐgĚ^X2Ţ}tŔpČc
ŮŤX/NŹuU`1}pđ,cĽz1L EİĞĞ]m[\]3TŻĆ~ĹY{Ża<{=FKK%3, MR_=gĄ*Ď)kŤ&ĹY0ŔĞ
ŸĂŰ@k(Ę4'4~T-|
%ŇidŃ:Z
rŠ6Tb'\oŚ*EP<z&gĄ*ĎC%ĎmuhŹŮM
VbqSza6bĞRŇ ŻČ\CŠ/-%-ŸP
ĘhVT/D*ŽRŮ%Ş@VŠZ[[đ^'GnĆr.Ť
*u_ĹĂ2lqrs130}<Ÿ7Ď>1d:6Ă2-'Ę|.İs%Ę[ĹK%1}pđa_.6&ĹŢMm'hf[HĚ9%3, QnŞ]-toĂ(4:Ą!doŢ[
    yb5ŹuU`1}pđY0S*hĄĄ#+.ĘŠn$)ŘAŠe/p3E6ŸjhĎR^§C
1Ű^ĹY0ŔĞ
ŸĂEz7-
o=Ň*b'R§Ž6:{~TcGŻŸtŘCcE&ĹŢMm'hfİŰ§ĽŘ<nuĂn=$Z5<
FnlŹYŃRĘĄİx.S'WD\Gj-f*E&k/x(Ğ
ŸĂ_ĹĂ2Ĺd%ŐŸ6QŤwŤcnJŞXM0/xu~gU)Ě§6BTEvŇĂŐVx"^?3đ^'G
```

```
Ñ\PR^§CH*ČŚ-%-ŸsŚCLKk?Ą&ĹŢMm'hf%NĞIA§CĆA!4OcwQ84XŔ"ŠV%Š~1Dzl~"ŃĎDŠ.gy(!<sŤnR^§C%
    Ş@V'ĚŸKW@;ŞŸ>Zo+sBnbWs3qSzaA:
Őx=KglĞ(k3KOŇ!ŔĎ:đ^'G^#r! OJŔviŃ
lu!hVJIĹ_ČIfl§,h Ş=&7MK$jEKRrjĘ?& A u
Ć/Ůi6po1}pđtNJX/;2O)_>NŤ
*uŔ_ŔIgPĚUŮO>+^,-'2j)NEŮŚŔĂ-X.H%%ÑF)z*bČĄ-^nde
Ľđ)
Ĺ\,3oOtMkĂŞVp~coĎ.Ro[NĹĘ^KcĹDG>18F%P.Ř)a[6w%;ZŽR^§C;Ŕ:I<sŤnŠRXiv=Ę%eŹŐđRJIN vxĽ
    ('-RF"}-
Ĺ\,3oOtŤ[{qfŤ1KŢgD3lfbvĆKŠoĎ.RVA>Ûfq4uŽEŇ}đ0ŠĞJŸQ0o7§cŹuU'%d66đe'ÛoĎ.RlĞ(k}(o[4
    XŔ"ŹuU'Ě§6B8 1Bq(§SŽx~_Ÿ§ZŮŸBEl^ŰSŞ
<0ŮIW=TŤ
*uS'WDs:J-[<W&D$-]JŞXM-ŇŢĄ1}pđ§ĆđZMŻNŻĂI.hbČĄ-Ľ3-EİŇ?2g
X JĚŚ7|HĚ6o9ŠDŮ#İ]cjh>hŻMS;Ŕ:I%Ŕ=vŐ'đ02hŤkjaKsK[§-9ŽŻV|ŽC)ŠŠ10tz@Ż$]%>W6Ľfd{ p~
    t&^#1Űo6
VĚPdeyŇ_iŔŃN&Ż u%d66Ple:İ-cĹbBc6 ^%jŤ8
Q_c9ĹIm§'ĆCpĂa<VY5ŰlfTEvÑ![j22,IŃy@/1}pđ,cĽzlL EİĞĞ]9ŽŻV)Ţ,qQ)ŘĞŇD+Ñ ]:*v>ŃS:0
    SDKŢgDO,'&TuŰH14Ę8H*ČŚđ^%.@đka9ŽŻVc+^ĄMŞe}Ť
*ul [:ŘĘr/Ż{ŞDPČfŔ OJŔlĞ(k](/{xhM3nQĹ4vĽrŽEDHRŤ
*u&A uqŞJo@ŠŇĄŢ ,YŹWrqwŤcnaqđ+f8]ŤŽxŞNmŐAđ"CRm]eftZxFenQĹ4iNŹrT
ŰŢk{riFZ] gQJ-ŽQn-Fn6LqSza#^2*
ŻMŰZkf,Ą>U*V
pA)olrSK,ŢzĂĹ^9ŽŻVkcFAnVUŘŤ&TŞ $]% Ęy+S*Ń)%đŃY:^~%2f3r§oŮ',3mnt.ÑCO!Ę6dOuĞ/{9JŤ
*uA n^Gw%%$kq2iw
6ŹRkqgĄ*ĎD$ĄB}gfO'"N@}-b|P.Ř)/*ŢwŸoŹHg8gđđ^'G!W
K+/92rĎĘ?]K(]!7"+&*66+İ
(6ĚĎ1sQrq_ŐvpĚŃŸđ§/&c-%-ŰQKbwŰgyxĹx+zin}6
```

ŹDŃasd4J9)Ň*kĘ OđŠŇGŽĂj

C#/ $]%§1s=-Ĺ%1i)%v4lĘaZ\o ŢgnŠD$ -] ĚEG-|AwŠjNZ9ŽŻVx~ŔĽF;#3?oŮws2K5’U(;E?zĆ<
Bv9bBŻr§oŮ‘Ť

*ulĞ(kPŢ;3?F&JŐĞĎm)u\?paŘcđ^’GQM4Ń ĆkxEXPBđT@$}XŘ8e§Ĺ

oUbŰk9yŻ1}pđlĞ(kŔ*UŮŸoŹHC~LLnQĹ4lu!hLNŸŻŰ"?v%[E]?Ż‘$3Ą6ĽŽŮy@TIvgy(!+<ŮĘ5j§}
ĆkxTEvŇ|q9§Ik-đ5ch;&pđSKŢgDrŞx<TŮhČmQv^% Ę_|ŘĞŽ%d66Ż{ŞD}gf0"{xPY f3$Ű

*jZT,ae8l§ ,h Ş=&Mojj8yŢŻŻŮĘ_L\BŻGIS

(9*P-o‘ĎČoŤd5j§} ĆkxŻ{ŞD 0JŔT~§qĆĄ(ĽŠŠ10SSs3k>" b1=NĚR-Ż=cYD2X&<xn2-MY]Ťl§,h Ş=&
If ,_Ővps--~§wĚ8;8kEoe$ ŠŸŢ]S%[E

fP‘JpdŔ1 ŸzĂ-X.V^kSY /?ađ.rp)FŸ

W,Ě2?nu_ĹĂ2PŘ1ŐŸ>Zo%[E ,w=^

BfNl§,hk’ĞLzl%[S’WDŐŇhd\IT‘\XĞ$§jfT3ŢpmQĄkŽŔ_ŔIoĹmŢ$ŃZŽ

rŠ6ŇMČ‘3ČqBKk?ĄqSza~rT@z}ŔŰ}tŔp9§ĄUMŢ]>XŽSŚŰ+JaX)ŰŘkđWwŹuU‘Ě§6Ban.6)7ŇS#lHBİGq^E1
&w\wŤkPČfŔ=WmL{.$ĚJĘŃ6oĹmŢ3j ,ŽSJĽĆn‘}(YŠp1NŽŸA#a9/ŞYŢi;<*Ť: Ź <ŇČ2I>f

TEvŇ‘ŐPtŘ}xĎČpX0Ęk9=)LZ;ađ.rFcT(m_4j

>4e(QnĞQ_ .-ĞHF>cbĄ;Ŕ:I9;9,&Nx5Ť&KĞĽ3-Em83:m’hfg !ŹVNŐwŞEs6s--~§$Ă!ČKgŠGQYo{‘
EŐŮU6|^+_k/%_LxŘĄFĄİx._ĹĂ2S-c,Fn6LqSzaBw%B;b6 e*s}|Ţ5YŤŇ99-%-ŸvZL@

ĄHb.c$ Ă)Ńrz;Cq_ĹĂ2QSCamsĞŚTb’\1}pđviŃ

}V+v"KĎŹ$ŐĎ3OfŻl ŽoŮĆ>W3x~ŔĽ?=Z9qd[htŐvxŹU>ŃSEF

TEvŇFRo§k90Ŕ;ŇIrĄe87BŸFk08E{&#DĹ(ŔqŔĽBŞA{WuĘA§CĆ[ĹK%ĂŐVxUm-ŽŔ’wdĚ§6B Ş=&Ţ;:jcbQ-
hBPmoĎ.RG ĎŘTF:ŻTEvŇ!;

r 2 ,IŃy@/1}pđlĞ(kw4RXŚqwŇtŤqihd’>?7

Q$&D@%wđRVŃtŚŮds-Š’=ĚVtWzAb-k#§G"7zv

Š^.m=CyĎjhĽ,Ň{5zI{*bŹ-:kDz‘Ń’^HŹ=/TN[-’ ĄŔS"Ő-B];Ŕ:IdIŠ Š’=Ě0Mk4=*#ELcdn;§^!ŹuU‘&
ĹŢMYKr06x)$L*UZĄĘŞ7wŤcn ĚEG2>wE#^>Ż*ŽRŮŸŮŐ§Xz@i|P-đŹuU‘ ^3GZ,IYŚ

rs;RgĞ§Ť3&57W! ‘UŹ>[Ře5yPo{mMĆ%{%\Şđ2Jy 1ŹhYr0@,+Ć*ŽRŮ ŸAb>[q"k>" $ČK!s--~§ŹŤ|x5M
^4siUayoŇbmCLcČCŰp,.[cGw%%VtWzIŽg+8$1:ĹNH3k’ĞLL#ŹŽŢ=/‘,_BsbHG|6x)$Wc+< ^%j{I

()^_Ž-Ęxf=,ŘNa4ŇA^g8gđ=^ĞĄ

đb<oĎ.R#s0 C !&6Ć6TA1}pđĹFŚ0Ű?Rab! E (iKŔ

(X ŸAb7qG\kQŞ&A_-$f"_fi322qĚ<[jČ1ŘŮU6|Ţ5o{UFGn-ŮŻ]#&#Ĺ ŽoŮ6ĘyWŸŮŐ§Xz@i6Źn+N<}Ů'3

m0Ÿ[§@fq§đksN_fq§đŠ^.mŚĞL.RŻv1SEF

TEvŇ"VŻ-% Ę_l 7ŽssPŇ.X^Ŕ9{BvI{*bŔ#eAG| piw

6!!V;Ąk'JŚŠ')

%Ňi/ĞnLZĽ>ŹŸŮŐ§Xz@i04<.-0, TEvŇ:^~%TEvŇ=WmL{.$ĚJĘŃ6oĹmŢH%%ŇĹ5Ż,uđqmŹuU'Ě§6BSi§TĆi

[ŠR^§CD$ĄBP5-T=đVn\'>aKoŢ0N<}ŮD$-]JŞXMBŤGhQ9:x

Bo];haŃĆ]b^eŐ]m,ŘXzŹŻh:đ^'G8ŮN-Xz@i&Ğ$ČhBPmTEvŇuejCTEvŇn)Ğ^Ÿ>Zo%[E ~rT@"m*9|Ğ|//

0ŸtuĽ^đ6ŻŃkLĄ4[^ĹŠŤ&ĹĆ<yDŇ,[ĹK%c rn6-4ŹFFĂH<Y"ŘĘxm4iw

6RŐh

SiPŐ*,ĄĞlĞ(k"kpc}R?M*ŽRŮđ^'GĘ_ĂŃŔH=5c**ŘĹG\oFy{*@{k:=đVnaqđ+Ę2Ů;r+Ď%eŹŐđ?ĚrŰQT\İ1

}pđy0Ş

tUİČđ^'GyA yI3E/tbđ6R\-8P5-T|f6'Ź$k D$ĄB"

WxLK}jg Bofua)

Ć/ŮG=ZV-p>oŃ,aKc rnBĄPy~NSŐČ7]9)uWW<eQ('$ĄrŹuU'ađ.rŐ3ĹPQ,SŸŻ_QPKo-|8ŮN-

pŚu_ĹĂ2a_S"1}pđ?^Mu:

[§t4MBmZT#i322Ÿ'ĹŻi322TN

x\ĘYk"KĎŹ&ĹŢMŸ?Ć7M

Vbđ4ŸĽn4EgcĄF6Aa~kĎ{x?""dmoĹmŢwŤcn&~,Ź4PZ-LJ5Ž-KŃ0[oZIyX/5Dc6#W]8_:Ďv%++Zs'l\oĂ)

ŮcŸjhĎn[puZ8

ŔyYtk

"D]5ŞBMTEvŇweCT4PpN§Ž6:ĄĘŞ7wŤcn9§ĄUŃ -ŽĞ

ŸĂ_ĹĂ2A§CĆTŘjMŠ^.m%[E ŢgnŠi ?*ŻđF67#@ĆRi322_ĹĂ2ptDjađ.r(U TTEvŇ+Ń=

@hŐŮF2bYJwŤcnŹ_q5dXŢ]b! EŠ^.mTEvŇizĞ+,ZVK\G1s_Ľ1H&Ń<M!9Ds

o=Ň

^FVp)FŸĂĘ|ŻŐŤegMS'F,cĽzf~ŇY4PZ-§9ĄhA/g[rPŞaŹ|§@ŔHŐb%T"ĎŤM5ĘCE%ŰctŸ#^2*|+İlĚ§6B%

wđRFnlŹYŃRĘLJ5ŽĎ.l8Xa*,}tŔpUŤiAĘ8’Di)%vCŔĂPŸ[§@I§,^I?ŇđwZp/-%-ŰeŹŐđsqŰ6vŇ7Ť5/

j)7_/~ađ.r-toĂmQ\ŞuĂŔĹJŸ"o@1}[]KĞX-_[&Ő-B]HN]LRgŘ]rMWjĚ§6BTEvŇLJ5ŽbVEĆŻzdt^#r

!a2v@BoŘIk’ĞLđ^’GĘCE%ŰctŸ#^2*‘NY>OtĹ

ŇBxOŃuĆ18Šq^ŻLGĘMS‘F$ĞZaI3)Fđ^’GA[vŮ%;ZŽR^§CŐpJ+VA|Ěfy;

=đVn9ŚŠsŢ‘/ĹŘCcEoĎ.R,h)F=cYDt?Č.nneLŚŕx*?|Ř’t?Č.")Ă"WQpĄNMkĆzĂĹ^$kq2iw

6Ćq‘ŹĽ.lŹCSAqađ.rFcT(/Ťđ2f0mČ?5 Ě*+94hŽI~;,gĘ*b’R&ŘH/BĄPyA[vŮ%;ZŽR^§CŐpJ+1VĞes2K5

"%tbĎŤM5U] J~^ggĞ

ŸĂ_ĹĂ2|O^#TŽgŚH*ČŚS’WD#w?‘ŇĆ4YĆY&}KŢgDŻ@<ĘĆh)<Ů:fj3VMY

&F-TEvŇ~ŃŽĹŃ?7pŐ-B]XŽSŚZ(^9‘nČ§ĆŐ<Ş37=IFŢY&[?ixŠo.!JŸyŻsA7moĎ.R\Ğ)oU}6Z+!Lk~

vsŞMUŹifĚĄŰ KrnWŽ2,=*#E@ŔHŐwĚ8;9fimN3hŘak2Ĺ

Appendix B

Computer Code

The following C++ computer programs were used to perform encyphering and distribution counting for the Declaration of Independence examples.

B.1 encypher.cc

```
//encypher.cc
//runs example cypherings for thesis examples
//by John Szwast


#include <cmath>
#include <stdlib.h>
//#include <stdio.h>
#include <string.h>
#include <iostream>
```

```
//#include <string>
#include <fstream>

#define NUMENCYPHERINGS 4

using namespace std;

int UsageMessage(char* runname);
int FileError(char type);

int main(int argc, char** argv)
{
  ifstream readfile;
  ofstream writefile[NUMENCYPHERINGS];
  string fileprefix,readfilename,writefilename;
  string writefiles[NUMENCYPHERINGS]={"affine", "digraph", "vector4graph", "
      vector2graph"};
  char ReadChar, WriteChar;
  size_t BytesRead;
  unsigned int p, c, dip[2], dic[2], qup[4], quc[4], octp[8], octd[8], octv[8], i
      , j, k;
  unsigned int v2p[2], v2c[2], v2i; //vector2graph variables
  unsigned long int ldip, ldic;
  div_t IntDiv;

  //Check Usage
  if(argc != 2)
```

```
    return UsageMessage(argv[0]);


  fileprefix = argv[1];
  readfilename = fileprefix + ".plaintext.text";


  cout << "File name:  " << readfilename << endl;


  //open input file
  readfile.open(readfilename.c_str(), fstream::in | fstream::binary);
  if(!readfile.is_open())
    return FileError(1);


  for(int counter=0; counter < NUMENCYPHERINGS; counter++)
  {
    writefilename = fileprefix + '.' + writefiles[counter] + ".text";
    writefile[counter].open(writefilename.c_str(), fstream::out | fstream::binary
        );
    if(!writefile[counter].is_open())
      return FileError(1);
  }


  i=j=k=v2i=0;
  while(readfile.good())
  {
    ReadChar = readfile.get();
    if(readfile.good())
    {
```

```
//Read the next character
p = (unsigned)ReadChar;
dip[i]=p;
qup[j]=p;
v2p[v2i]=p;

//Encypher single character.
c = (53*p + 78) % 128;
WriteChar = (char)c;
if(writefile[0].good())
  writefile[0].put(WriteChar);
else
  return FileError(3);

//If a digraph is ready, encypher it.
i++;
if(i==2)
{
  i=0;
  ldip=128*dip[0]+dip[1];
  ldic=(((8567*ldip)%16384) + 612) % 16384;
  dic[1] = ldic % 128;
  ldic = (ldic - dic[1]) / 128;
  dic[0] = ldic;
  if(writefile[1].good())
    for(int counter=0; counter < 2; counter++)
    {
```

```
      WriteChar = (char)dic[counter];
      writefile[1].put(WriteChar);
    }
  else
    return FileError(3);
}

//If a 4-graph is ready, encypher it.
j++;
if(j == 4)
{
  j = 0;
  quc[0] = (37*qup[0] + 68*qup[1] + 26*qup[2] + 95*qup[3] + 89) % 128;
  quc[1] = (16*qup[0] + 103*qup[1] + 100*qup[2] + 89*qup[3] + 92) % 128;
  quc[2] = (122*qup[0] + 33*qup[1] + 17*qup[2] + 51*qup[3] + 59) % 128;
  quc[3] = (55*qup[0] + 42*qup[1] + 82*qup[2] + 24*qup[3] + 92) % 128;
  if(writefile[2].good())
    for(int counter=0; counter < 4; counter++)
    {
      WriteChar = (char)quc[counter];
      writefile[2].put(WriteChar);
    }
  else
    return FileError(3);
}
```

```
      //If a digraph is ready, encypher the vector2graph (some redundancy with
          multidigit digraph above)
      v2i++;
      if(v2i == 2)
      {
        v2i=0;
        v2c[0] = (95 * v2p[0] + 5 * v2p[1] + 43) % 128;
        v2c[1] = (97 * v2p[0] + 58 * v2p[1] + 99) % 128;
        if(writefile[3].good())
          for(int counter=0; counter < 2; counter++)
          {
            WriteChar = (char)quc[counter];
            writefile[3].put(WriteChar);
          }
        else
          return FileError(3);
      }
    }
  }

  readfile.close();
  for(int counter=0; counter < NUMENCYPHERINGS; counter++)
    writefile[counter].close();

  return 0;

}
```

```
int UsageMessage(char* runname)
{
  cerr << "Usage:  " << runname << " <fileprefix>\n" << " Note: fileprefix.
      plaintext.text must exist\n";
  return 1;
}

int FileError(char type)
{
  if(type == 1)
  {
    cerr << "File input error.\n";
    return 2;
  }
  else if(type == 2)
  {
    cerr << "File syntax error.\n";
    return 3;
  }
  else if(type == 3)
  {
    cerr << "File output error.\n";
    return 4;
  }
}
```

B.2 distributioncount.cc

```
//distributioncount.cc
//counts the character distribution of a text file under a specified alphabet
//by John Szwast

//#include <cmath>
#include <stdlib.h>
//#include <stdio.h>
#include <string.h>
#include <iostream>
//#include <string>
#include <fstream>

using namespace std;

int UsageMessage(char* runname);
int FileError(char type);

int main(int argc, char** argv)
{

  ifstream readfile;
  string filename;
  const char AlphabetMap[3][256] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0,
                                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                          0,
```

```
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0,65,66,67,68,69,70,71,
     72,73,74,75,76,77,78,79,
80,81,82,83,84,85,86,87, 88,89,90, 0, 0, 0,
    0, 0,
 0,65,66,67,68,69,70,71,
     72,73,74,75,76,77,78,79,
80,81,82,83,84,85,86,87, 88,89,90, 0, 0, 0,
    0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
```

```
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, //26-letter (A-Z)
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
32, 0, 0, 0, 0, 0, 0,39, 0, 0, 0, 0,44, 0,46,
     0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,58, 0, 0, 0,
     0,63,
 0,65,66,67,68,69,70,71,
     72,73,74,75,76,77,78,79,
80,81,82,83,84,85,86,87, 88,89,90, 0, 0, 0,
    0, 0,
 0,65,66,67,68,69,70,71,
     72,73,74,75,76,77,78,79,
80,81,82,83,84,85,86,87, 88,89,90, 0, 0, 0,
    0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
```

```
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, //32-letter (A-Z ,.?':)
 0, 1, 2, 3, 4, 5, 6, 7, 8,
     9,10,11,12,13,14,15,
16,17,18,19,20,21,22,23,
    24,25,26,27,28,29,30,31,
32,33,34,35,36,37,38,39,
    40,41,42,43,44,45,46,47,
48,49,50,51,52,53,54,55,
    56,57,58,59,60,61,62,63,
64,65,66,67,68,69,70,71,
    72,73,74,75,76,77,78,79,
80,81,82,83,84,85,86,87,
    88,89,90,91,92,93,94,95,
96,97,98,99,100,101,102,103,
    104,105,106,107,108,109,110,111,
112,113,114,115,116,117,118,119,
    120,121,122,123,124,125,126,127,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0,
```

```
                                       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                            0,
                                       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                            0,
                                       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                            0,
                                       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                            0,
                                       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                            0,
                                       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                            0,
                                       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                            0}; //128-letter (0-127)
int AlphabetDistribution[256];
char ReadChar;
size_t BytesRead;
int Alphabet;
int PrintBounds[3][2] = {64, 93,  32, 93,  0, 127}; //lower and upper bounds on
     each alphabet to be printed
                  //This helps generate a clean gnuplot histogram.
//Check Usage
if(argc != 3)
  return UsageMessage(argv[0]);

Alphabet = atoi(argv[1]);
```

```
filename = argv[2];
cout << "# File name:  " << filename << endl;

//open input file
readfile.open(filename.c_str(), fstream::in | fstream::binary);
if(!readfile.is_open())
  return FileError(1);

for(int i=0; i<256; i++)
  AlphabetDistribution[i] = 0;

while(readfile.good())
{
  ReadChar = readfile.get();
  if(readfile.good())
    AlphabetDistribution[AlphabetMap[Alphabet][ReadChar]]++;
}

for(int i=PrintBounds[Alphabet][0]; i<=PrintBounds[Alphabet][1]; i++)
{
  //if(i%5 ==0)
    cout << i;
  //else
  //  cout << ' ';
  cout << '\t' << AlphabetDistribution[i] << '\n';
}
```

```
  readfile.close();


  return 0;


}


int UsageMessage(char* runname)
{
  cerr << "Usage:  " << runname << " <alphabet> <filename>\nAlphabet: 0 - 26-
      letter (A-Z)\n"
    << "          1 - 32-letter (A-Z ,.?':)\n"
    << "          2 - 128-letter (0-127)\n";
  return 1;
}


int FileError(char type)
{
  if(type == 1)
  {
    cerr << "File input error.\n";
    return 2;
  }
  else if(type == 2)
  {
    cerr << "File syntax error.\n";
    return 3;
  }
```

```
}
```

Bibliography

[1] Larry C. Grove. *Algebra.* Dover Publications, Inc., Mineola, New York, 2004.

[2] Darel W. Hardy and Carol L. Walber. *Applied Algebra.* Pearson Education, Inc., Upper Saddle River, New Jersey, 2002.

[3] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis.* Combridge University Press, 40 West 20th Street, New York, New York, 1991.

[4] Neal Koblitz. *A Course in Number Theory and Crypography.* Number 114 in Graduate Texts in Mathematics. Springer-Verlag, New York, New York, second edition, 1998.

[5] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press LLC, Boca Raton, Florida, 1996.

[6] Carl Pomerance. A tale of two sieves. *Notices of the AMS*, 43(12):1473–1485, December 1996.

[7] Kenneth H. Rosen. *Eleentary Number Theory and Its Applications.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.

[8] Joseph H. Silverman and John Tate. *Rational Points on Elliptic Curves*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, New York, 1992.

Index