6-2016

The Evolution of Cryptology

Gwendolyn Rae Souza
California State University - San Bernardino

The Evolution of Cryptology

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

in

Mathematics

---

by

Gwendolyn Rae Souza

June 2016

The Evolution of Cryptology

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Gwendolyn Rae Souza

June 2016

Approved by:

Shawnee McMurran, Committee Chair	Date
Jeremy Aikin, Committee Member	
Corrine Johnson, Committee Member	
Charles Stanton, Chair, Department of Mathematics	Corey Dunn Graduate Coordinator, Department of Mathematics

Abstract

We live in an age when our most private information is becoming exceedingly difficult to keep private. Cryptology allows for the creation of encryptive barriers that protect this information. Though the information is protected, it is not entirely inaccessible. A recipient may be able to access the information by decoding the message. This possible threat has encouraged cryptologists to evolve and complicate their encrypting methods so that future information can remain safe and become more difficult to decode. There are various methods of encryption that demonstrate how cryptology continues to evolve through time. These methods revolve around different areas of mathematics such as arithmetic, number theory, and probability. Another concern that has brought cryptology into everyday use and necessity is user authentication. How does one or a machine know that a user is who they say they are? Living in the age where most of our information is sent and accepted through computers, it is crucial that our information is kept safe, and in the appropriate care.

Acknowledgements

I had started this project around the same time I was reading Stieg Larsson's *The Girl With The Dragon Tattoo*. A lot of my inspiration to research this topic came from the character Lisbeth Salander, a young woman who is a notorious computer hacker who will stop at nothing to find out the truth. I would like to thank my mom for not only helping me take care of my daughter while I worked on my project, but for always believing I could do this. The third woman I owe many thanks to is Shawnee McMurran. Thank you for helping me make this paper the best it can be, and for taking the time to be my advisor. You have been an inspiration to me since I first took your class in 2013. Thanks to Dr. Aikin for looking over everything and making sure all of the sections flowed nicely. I also owe a huge thank you to Dr. Johnson for her meticulous editing and for giving me so many great suggestions. Lastly, I would like to acknowledge the biggest contributor to my interest in mathematics, my dad: problem solver extraodinaire, and the person who showed me my first mathematical magic trick ever. My love for numbers and puzzles would not exist without you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this paper we will analyze a few breakthroughs of cryptology, beginning from the BC era to present day. The earliest development of cryptology stemmed from the need to keep information safe while being sent between two recipients. This is still the primary reason why cryptology is used today. The general process revolves around two actions: disguising the original message a user wants to send and decoding a disguised message. These actions are known as encryption and decryption, or deciphering. The disguised form of the message is known as the cipher text. Throughout my project, I will introduce and examine the methods of ciphering that I found most compelling. While introducing each cipher, I will additionally provide historical information and significance.

Chapter 2 will introduce some of the first forms of encryption, starting with simple substitution methods. These methods evolved with time as users needed more secure ways of communicating. (Fortunately, as cryptology advances, cryptanalysts do as well.) Affine ciphers provided a little more security by enciphering messages under more than one operation. Near the early 1400s, it was time to develop a new means of ciphering, where cipher texts were not as easy to decipher. Thus, the famous Vigenère cipher was born and was considered a breakthrough as one of the first polyalphabetic ciphers. Its complexity is derived from being a composition of multiple substitution ciphers.

The Vigenère cipher will be a prominent focal point in my thesis project. I will explain in depth two methods commonly used for decrypting a text under this cipher, the Kasiski and Friedman Tests. These two tests inspired all cryptologists alike by showing that different areas of mathematics can be used in cryptology, not just simple algebra.

(However, one's underlying knowledge in number theory and probability help tremendously.) The Kasiski Test deals with recurring strings of letters found in the cipher text, giving us clues as to what the length of the keyword might be. (The keyword is a tool for encryption.) The Friedman Test not only provides an estimate of the length of the keyword, but also proves whether or not our cipher text is monoalphabetic or polyalphabetic, which should always be the first question at hand.

When the earliest computers were created, it was much easier to use a binary numeral system, a system in which information is solely expressed in combinations of 0s and 1s, where each digit is referred to as a bit. The use of computers and this system opened many doors for cryptologists. Creating ciphers and algorithms was just a click away, and recreating it on paper was relatively easy once one knew what to do. Thus, another method of enciphering was created using shift registers. Shift registers are able to produce pseudorandom binary sequences, which are very useful in cryptography and provide extra camouflage/security for a message. They work like miniature machines, where the shift register, i.e., the machine, requires the most work to build. The section on shift registers will provide a detailed example, with outlined steps on how our machine is constructed and how it processes.

In Section 4.2 we will see that cryptology can be used for other purposes besides encrypting messages or information. Continuing on the evolution of cryptography with computers, one section will define and focus on message authentication codes (MAC) and digital signatures. MAC and digital signatures deal with verification that a user is in fact who they proclaim to be, a topic that we will revisit in the final sections of the thesis. Public-Key Cryptosystems will also be introduced, as they are crucial to understand before comprehending the final sections. This section, will prepare the reader for the next which will cover significant breakthroughs in cryptology that are still used today.

Section 4.4 will introduce the reader to concepts and methods necessary to comprehend before approaching the most famous algorithm pertaining to cryptology, the RSA algorithm. We will discuss the derivation and significance of the algorithm, as well as the founders and other protocols that the founders contributed to cryptology. Proofs will be provided and outlined, and examples will be included that show that the algorithm does indeed work and can be highly effective and secure if applied appropriately. One of the first commercial applications of the RSA algorithm that will be examined is key exchange,

which consists of deriving a mutual key between sender and receiver by using values that they initially choose. We will primarily focus on the Diffie-Hellman key exchange scheme. The last cryptosystem that will be discussed is the knapsack cryptosystem. In Section 4.5 we will present one method of attack on this cryptosystem, showing that any information being sent through this system is not secure. We will conclude the project with a chapter that gives an overview of current cryptology and how crucial it is that the world of cryptology continues to evolve.

Chapter 2

The First Ciphers

2.1 Monoalphabetic Ciphers: The Caesar and Affine Ciphers

Monoalphabetic ciphers are defined by each letter in the alphabet being mapped to exactly one other letter during the encryption process. These types of ciphers are sometimes called substitution ciphers. One of the most famous and earliest monoalphabetic ciphers was the Caesar cipher, which consists of shifting the letters in the alphabet. Thus, there are a total of 26 possible additive shifts [Beu94].

Example 2.1.1. Encrypting a message with an additive shift of 3: By letting the letters of the English alphabet correspond to the numbers $0, 1, ..., 25$, we get $A = 0$, $B = 1$, ..., $Z = 25$. To encrypt our given cleartext/message with a shift of 3, one would add 3 to each number that represents the letters in the given cleartext.

For example, since $A = 0$, encrypting A from cleartext to ciphertext would give D, since $D = 3$ and $0+3 = 3$. Using the same additive shift of 3, C, which is 2 in cleartext, would translate to F in ciphertext, since F corresponds to 5.

A cipher that is a bit more complex, requiring multiplication as well as addition, is the *affine cipher*. An affine cipher can be represented in the form,

$$ax + b \pmod{26},$$

where a is the multiplicative shift, b is the additive shift, and x is the number representing the letter in cleartext that one is encrypting. There are only 12 possible multiplicative

shifts: a must be an element of the set {1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25}. This set consists of numbers inclusively between 1 and 26 that are relatively prime to 26. One is able to perform multiplicative shifts from the above set because each element of this set has a multiplicative inverse a^{-1}, considered modulo 26, such that $aa^{-1} \equiv 1 \pmod{26}$. This is a necessary condition for a one-to-one mapping, or in this case, for the affine cipher to be monoalphabetic. This condition of having a multiplicative inverse also allows us to decipher the text. The following table shows the set of positive integers smaller than and relatively prime to 26, with their inverses.

possible a's	1	3	5	7	9	11	15	17	19	21	23	25
a^{-1}	1	9	21	15	3	19	7	23	11	5	17	25

It is also worth noting that affine ciphers provide significantly more security than Caesar ciphers alone. Since there are a total of 26 additive ciphers, and 12 possible multiplicative ciphers, there are a total of $26 \cdot 12 = 312$ possible affine ciphers.

In the following formula, and throughout this paper, we will frequently have need to refer to residues in a modular congruence class. In such cases we will use the notation x = y mod n to mean x equals the least nonnegative residue y (mod n).

To decrypt a given ciphertext with a known $[a, b]$ is a matter of substituting a^{-1} and b into the formula:

$$p = a^{-1}(c - b) \bmod 26,$$

where c is the number corresponding to a letter in ciphertext, and p is the desired number that corresponds to a letter in cleartext.

Example 2.1.2. Ciphertext: ***HGGVUB*** and $[a, b]$= [5,7].

HGGVUB	7	6	6	21	20	1
$a^{-1}(c-b) \bmod 26 = p$	21(7-7) = 0	$21(6-7) \bmod 26 = 5$	5	$21(21-7) \bmod 26 = 8$	$21(20-7) \bmod 26 = 13$	$21(1-7) \bmod 26 = 4$
Cleartext	A	F	F	I	N	E

If $[a, b]$ is not known, then *frequency analysis* may be used to assist us in decrypting a given ciphertext.

2.2 Decryption Using Frequency Analysis

Frequency analysis in cryptology is the process of analyzing the frequencies in the letters in ciphertext and making the assumption that the letter that appears most frequently maps from the letter that has the highest frequency in an English text: the letter E. (See Table 2.1 [Beu94].)

letter	**relative frequency(%)**	**letter**	**relative frequency(%)**
a	8.167	n	6.749
b	1.492	o	7.507
c	2.782	p	1.929
d	4.253	q	0.095
e	12.702	r	5.987
f	2.228	s	6.327
g	2.015	t	9.056
h	6.094	u	2.758
i	6.966	v	0.978
j	0.153	w	2.360
k	0.772	x	0.150
l	4.025	y	1.974
m	2.406	z	0.074

Table 2.1: Frequency Distribution of all 26 Letters in English Text

One can conclude that this approach is more accurate if the ciphertext is reasonably large so that it makes sense to consider frequency among letters. Assuming that the most frequent letter in the ciphertext maps from E, if the ciphertext was encrypted using a Caesar cipher, we can count the additive shift used in the cipher from E to the letter we believe maps from E. Thus, we can reverse this shift on all the other letters in the ciphertext to see if one has a coherent message. If an affine cipher was used, one can still use frequency analysis to determine both shifts, a and b.

Example 2.2.1.

Given a text encrypted with an affine cipher:

JZQ DQN IQY UXQRC JZQ CZQN ERN JZQ HLMQ IQY UXQRC JZQ HEKI NUUD

By frequency analysis, the letter that appears the most in our ciphertext is Q. We may assume that Q maps from E. Since $Q = 16$ and $E = 4$, to find our a and b, we set up the mapping equation from E to Q:

$$4a + b \pmod{26} \equiv 16.$$

Now, as stated earlier, a must be an element of the set $\{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$. So let us first assume that $a = 3$:

$$\begin{aligned} 4(3) + b &= 16 \\ 12 + b &= 16 \\ b &= 4. \end{aligned}$$

This gives us a plausible combination of $[a, b]$ to test. Putting [3,4] and each letter in our ciphertext into $p = a^{-1}(c - b)$, we get a coherent message:

THE RED KEY OPENS THE SHED AND THE BLUE KEY OPENS THE BACK DOOR

If $a = 3$ had not worked, the next reasonable choice would be to test $a = 5$, and so on. If $a = 1$, then the cipher would have been encrypted by a simple substitution cipher. This method of trying all possibilities is called a *brute force attack*. A brute force attack of trying all possible a's would take only about 10 attempts.

Frequency analysis will continue to be a valuable tool for decryption in the next chapter, as we discuss one of the first known polyalphabetic ciphers, the Vigenère Cipher.

Chapter 3

Polyalphabetic Ciphers and The Vigenère Cipher

"[Enciphered writing] must not rely upon secrecy, and it must be able to fall into the enemy's hands without disadvantage."

-Auguste Kerckhoff, *La Cryptographie Militaire, 1883.*

3.1 The Vigenère Cipher

The Vigenère cipher was a significant breakthrough for early 16th century cryptology. Prior to this discovery, cryptologists were having difficulty keeping messages secure and undecodable. Since frequency analysis had become a popular code breaking strategy, monoalphabetic ciphers were becoming less of an issue for all cryptanalysts alike, and a new method of ciphering was sought. This led several cryptography enthusiasts to attempt to produce a new cipher where it was possible for two different letters in cleartext to become the same letter in ciphertext through the encryption process. From their efforts, French diplomat Blaise de Vigenère was inspired to devise the new cipher fitting that which the enthusiasts had in mind [Beu94]. Thus the cipher was coined the "Vigenère cipher," and the table that breaks down the cipher that can be used as a tool for encryption and decryption was coined the "Vigenère square." (See Figure 3.1.)

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

Figure 3.1: The Vigenère Square

The square in Figure 3.1 [Beu94] displays all of the possible monoalphabetic ciphers/shifts. In comparison with Caesar ciphers, messages encrypted with Vigenère ciphers are exceedingly more secure. Within a Caesar cipher, for example, if the letter A is encrypted as the letter Y, then it is impossible for another letter to be encrypted as Y. This is not the case with Vigenère ciphers, or any other polyalphabetic cipher. In Vigenère ciphers, each letter in the ciphertext has 26 possible outcomes when decrypted, depending on the keyword that is used during encryption. As with affine ciphers, knowing the different shifts used from each letter in the keyword helps when decrypting a Vigenère cipher, and is essential when encrypting each letter in a message.

Example 3.1.1.

Let our *cleartext* be:

THERESTROUBLECLOSEBY

and our *keyword* be:

LEAVE

Let n denote the length of the keyword. In this case $n = 5$. We will then proceed with encryption by writing out the keyword directly under the first 5 letters of our cleartext, and continue writing it underneath until the string of our repeated keyword is equivalent in length to our cleartext:

T	H	E	R	E	S	T	R	O	U	B	L	E	C	L	O	S	E	B	Y
L	E	A	V	E	L	E	A	V	E	L	E	A	V	E	L	E	A	V	E

In order to obtain the *ciphertext*, the next step is to convert all letters to their corresponding assigned number, 0 to 25, as visible in the previous chapter:

19	7	4	17	4	18	19	17	14	20	1	11	4	2	11	14	18	4	1	24
11	4	0	21	4	11	4	0	21	4	11	4	0	21	4	11	4	0	21	4

The following step is to add the numbers in each column and compute each sum (mod 26). Doing this for each column will give the ciphertext in numerical form. Converting these sums to letters will give us the desired final ciphertext. For example, the first entry would be $(19 + 11) \bmod 26 = 4$, so the first letter would be E. Thus the complete ciphertext would be:

ELEMIDXRJYMPEXPZWEWC

If the keyword is known to both parties, then decrypting the ciphertext is only a matter of reversing the number of shifts used for each letter. If the keyword is *not* known, then the first task at hand would be to find the length of the keyword.

3.2 The Kasiski Test

One method of possibly finding the length of the keyword is the Kasiski test. The Kasiski test was first published by Prussian colonel and cryptologist Friedrich Wilhelm Kasiski in 1863. However, it is known that the earliest discovery of this particular method is credited to mathematician and inventor Charles Babbage. Babbage first thought of

the method of finding regularities and patterns in ciphertext while studying the cipher in the mid-1840s. His first successful attack using this method was made in the 1850s during the Crimean War, however, the method was kept as a military secret by British Intelligence. Despite not being publicly credited for this particular discovery, Babbage continued to invent and is most famous for his work in designing mechanical calculators [Sta10].

The Kasiski test consists of 3 steps: finding repeated strings of letters in a ciphertext, finding the distances between these sequences, and calculating the common divisors of the distances between all sequences [Beu94]. One of these common divisors is suggested to be the length of the keyword. The best way to see *why* this method works is through a particularly (short) example where the method might be of use.

Example 3.2.1.

Let our *cleartext* be:

THECATSEESTHECATNEXTDOOR

and our *keyword* be:

WATER

In this example, the string *THECAT* is repeated twice in the cleartext. When strings of letters or words are repeated in the cleatext, the Kasiski test is more effective since the possibility of seeing a repeated sequence of letters in the ciphertext increases. We can also presume that the larger the ciphertext, the greater the possibility that repeated sequences are likely to occur. This is because there are words in English that occur regularly in text, e.g., *THE*, *AND* and *THAT* [Beu94]. Encrypting our cleartext from Example 3.1.1, we get:

T	H	E	C	A	T	S	E	E	S	T	H	E	C	A	T	N	E	X	T	D	O	O	R
W	A	T	E	R	W	A	T	E	R	W	A	T	E	R	W	A	T	E	R	W	A	T	E

19	7	4	2	0	19	18	4	4	18	19	7	4	2	0	19	13	4	23	19	3	14	14	17
22	0	19	4	17	22	0	19	4	17	22	0	19	4	17	22	0	19	4	17	22	0	19	4

Computing the sums of each column, modulo 26, and converting our numbers back to letters, we get our ciphertext:

PHXGRPSXIJPHXGRPNXBKZOHV

We see that the only repeated sequence is **PHXGRP**, and the distance, or the number of letters, between each set is 10, with prime factors of 2 and 5. So, by the Kasiski test, the length of the keyword is suggested to be 2, 5 or 10. Note that 5 is the length of the keyword in our example.

After finding the length n of the keyword, we can complete the deciphering process by analyzing the frequencies of letters in our ciphertext. If we separate the ciphertext in groups of n, we know that the first letter in each group of n must come from the same column in the Vigenère square because the same shift was used for those letters.

Using the example from above:

PHXGR PSXIJ PHXGR PNXBK ZOHV

We begin by applying frequency analysis for the first letters of every group. Therefore, we consider the set: {P, P, P, P, Z}. We first assume the letter that appears most frequently corresponds to E in the original message, revealing what shift might have been used and thus giving us the first letter of the keyword. So we assume that P corresponds to the letter E in cleartext. We subtract the numbers that represent P and E and reduce modulo 26: $(15 - 4) \bmod 26 = 11$. Since 11 corresponds to the letter L, we assume that the first letter of the keyword is L. To continue this process, frequency analysis would be applied to the second letter of each group, then the third, until we reach the nth letter of each group. If at any point the deciphered message clearly becomes incomprehensible, we try to determine which letter of the key may have been derived incorrectly, and start over from that location using the letter with the next highest relative frequency from Table 2.1. In this example, we would start over from the beginning, and assume that P corresponds to the letter T in cleartext. Subtracting P and T, we get $(15 - 19) \bmod 26 = 22$. Thus, our next assumption would be that the first letter of the keyword is W, which, in this case, it is! We continue by assuming that the most frequent letter in the second set of letters corresponds to E in plaintext, and repeat the process of trying to find a coherent keyword.

This process becomes more reliable and accurate as the ciphertext grows larger and the keyword stays relatively short in length, perhaps of length 8 or less.

3.3 Index of Coincidence and The Friedman Test

One significant contributor to the US government's cryptographic efforts was William Friedman. During World War I, Friedman worked with the US government and led courses that consisted of training prospective cryptanalysts. His cryptological methods became classics in cryptology history, making Friedman an important figure in the early NSA (National Security Agency) [Lev01].

In order to proceed to the next method of solving the Vigenère cipher, the Friedman test, we must first introduce the Index of Coincidence (IOC), a technique also developed by William F. Friedman. The IOC represents the probability that, after selecting a pair of letters from a text, the two letters are the same.

Derivation of the Index of Coincidence [Beu94]:

Consider an arbitrary sequence of letters of length n, $n \geq 2$. Let n_1 denote the number of a's, n_2 the number of b's, ... , n_{26} the number of z's. We are interested in how often a randomly selected pair of letters would both be a.

There are n_1 ways to choose the first a. There are $n_1 - 1$ possibilities for choosing the second a. Since we neglect the order of the letters in the pair, the number of pairs of a's equals:

$$\frac{n_1(n_1-1)}{2}.$$

Therefore, the total number of pairs that consist of equal letters (that is both a, or both b, ..., or both z) equals:

$$\frac{n_1(n_1-1)}{2} + \frac{n_2(n_2-1)}{2} + \cdots + \frac{n_{26}(n_{26}-1)}{2} = \sum_{i=1}^{26} \frac{n_i(n_i-1)}{2}. \tag{3.1}$$

We compute the probability of choosing a pair of equal letters at random by dividing Formula (3.1) by the number of all possible cases:

$$\frac{\sum_{i=1}^{26} n_i(n_i-1)/2}{n(n-1)/2} = \frac{\sum_{i=1}^{26} n_i(n_i-1)}{n(n-1)}$$

Thus we have,

$$I = \frac{\sum_{i=1}^{26} n_i(n_i - 1)}{n(n-1)}, \tag{3.2}$$

where $0 \leq I \leq 1$.

Another representation of the IOC is:

$$p_1 \cdot p_1 + p_2 \cdot p_2 + \cdots + p_{26} \cdot p_{26} = \sum_{i=1}^{26} p_i^2,$$

where $p_i \cdot p_i$ is the probability that two chosen letters in the text will be the same letter.

For the English language, the IOC is:

$$\sum_{i=1}^{26} p_i^2 \approx 0.065.$$

By this calculation, one can determine if a monoalphabetic or polyalphabetic cipher was used by comparing the IOC to 0.065. If a ciphertext has an IOC close to 0.065, it is likely that the text was encrypted with a monoalphabetic cipher. On the other hand, if the text was encrypted using a polyalphabetic cipher, each letter theoretically occurs with the same probability of $\frac{1}{26}$. In this case the IOC would be estimated at:

$$\sum_{i=1}^{26} p_i^2 = \sum_{i=1}^{26} \frac{1}{26^2} = 26 \cdot \frac{1}{26^2} = \frac{1}{26} \approx 0.038. \tag{3.3}$$

The Index of Coincidence can also assist us in finding the length of our keyword and is a shorter process than the Kasiski test. The formula to find the length of a keyword is

$$l = \frac{0.027n}{(n-1)I - 0.038n + 0.065}, \tag{3.4}$$

where l is the length of the keyword, I is the calculated IOC of the ciphertext, and n is the number of letters in our ciphertext.

Before we derive Formula (3.4), we must first visualize what exactly the length of a keyword does: it divides our ciphertext into groups of length l. If l is known, we can

organize our ciphertext into l columns, such that the first letter in every group of length l belongs in one column, the second letter of each group belongs in the second column, and so on. If we organize the ciphertext from Example 3.1.1 in this manner, knowing that the length of the keyword is 5, the ciphertext would look like this:

P	H	X	G	R
P	S	X	I	J
P	H	X	G	R
P	N	X	B	K
Z	O	H	V	

Note that there are the same number of rows as groups of l's; this will always be true. Since we are now able to visualize a ciphertext divided into columns, we may begin the steps to derive the formula used to find the length of the keyword.

Steps for the derivation of Formula (3.4) [Beu94]:

We start by deriving how many pairs of letters must be in each column. Let n denote the number of letters in the ciphertext. Then each column has n/l letters. There are exactly n possibilities for a randomly chosen letter. In the column the chosen letter is located, there are exactly $n/l - 1$ other letters. Thus there are $n/l - 1$ ways to choose the second letter from the same column, and the number of pairs of letters that are in the same column equals

$$n \cdot \frac{\left(\frac{n}{l} - 1\right)}{2} = \frac{n(n-l)}{2l}. \tag{3.5}$$

Since there are exactly $n - n/l$ letters outside the chosen letter's column, the number of pairs of letters that are in different columns equals

$$n \cdot \frac{\left(n - \frac{n}{l}\right)}{2} = \frac{n^2(l-1)}{2l}. \tag{3.6}$$

Now, recall that the English language has an IOC of 0.065 and a completely jumbled text, where every letter occurs with the same probability, has an IOC of approximately 0.038 (by Formula 3.3). Combining these IOCs with our observations, we see that the expected number, A, of pairs of equal letters is

$$A = \frac{n(n-l)}{2l} \cdot 0.065 + \frac{n^2(l-1)}{2l} \cdot 0.038. \tag{3.7}$$

Therefore, the probability that a pair of letters chosen at random are the same letter equals

$$\frac{A}{n(n-1)/2} = \frac{n-l}{l(n-1)} \cdot 0.065 + \frac{n(l-1)}{l(n-1)} \cdot 0.038$$

$$= \frac{1}{l(n-1)} \cdot [0.027n + l(0.038n - 0.065)].$$

Since the IOC approximately equals the above probability, we obtain

$$I \approx \frac{0.027n}{l(n-1)} + \frac{0.038n - 0.065}{n-1}. \tag{3.8}$$

Solving for our variable l, we finally obtain the desired formula by Friedman,

$$l = \frac{0.027n}{(n-1)I - 0.038n + 0.065}. \tag{3.9}$$

To apply Formula (3.9), we first determine the length n of the text and the frequencies, $n_1, \ldots, n_{26}$, among the letters. We next substitute these values into Formula (3.2) to obtain I. Lastly, we substitute the values I and n into Formula (3.9) to obtain an estimate of the length of the keyword.

Example 3.3.1. In the following table [Bar02], we have the counts n_i of the various letters in a given ciphertext:

A	B	C	D	E	F	G	H	I	J	K	L	M
3	2	7	2	4	1	8	9	10	5	5	5	11

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
5	5	5	3	8	8	12	2	8	9	13	1	1

The total number of letters is $n = 152$, and

$$\sum_{i=1}^{26} n_i(n_i - 1) = 3 \cdot 2 + 2 \cdot 1 + 7 \cdot 6 + \cdots + 13 \cdot 12 + 1 \cdot 0 + 1 \cdot 0 = 1048.$$

Thus the IOC is

$$I = \frac{\sum_{i=1}^{26} n_i(n_i - 1)}{n(n-1)} = \frac{1048}{152 \cdot 151} \approx 0.0457.$$

Since I is not estimated at about 0.065, it is likely that we were given a text enciphered with a polyalphabetic cipher. By putting n and I into the formula for deriving l (3.9), we get

$$l = \frac{0.027(152)}{(152-1)0.0457 - 0.038(152) + 0.065} = \frac{4.104}{1.1897} \approx 3.45,$$

suggesting that the length of the keyword must be the next whole number, 4. After obtaining the length of the keyword, one can apply the same steps used in Section 3.1 to find the shifts/letters of our keyword.

3.4 Other Classical Ciphers

A cipher that inspired many other cryptological ideas is the *Hill Cipher*, sometimes known as *Hill's System*. This cipher was invented by mathematician and educator Lester S. Hill in 1929. Hill served as a lieutenant in the US Navy in World War I, and continued to invest time and send suggestions regarding cryptological techniques to naval communications until retiring in 1960 [Kah67].

The Hill Cipher was one of the first known *polygraphic* ciphers, a cipher in which it is possible to operate on large groups of symbols at once. Like the Vigenère Cipher, the Hill Cipher requires a key for both encryption and decryption. Unlike the Vigenère cipher, this key is in the form of a matrix.

The key for this cipher will be in the form of a 2×2 matrix, consisting of the elements a, b, c, and d, such that these elements are in $\mathbb{Z}_{26}$. We then allow p_k and p_{k+1} to be two characters in the kth and $(k+1)$st positions of our plaintext message. Two ciphertext characters in corresponding positions, c_k and c_{k+1}, will be enciphered

by multiplying our key matrix by our two plaintext letters as a 2×1 matrix as shown [Lew00]:
For k an integer,

$$\begin{pmatrix} c_k \\ c_{k+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} p_k \\ p_{k+1} \end{pmatrix} \mod 26.$$

To decrypt a given ciphertext while knowing the key matrix, we simply put each pair of ciphertext characters into the equation

$$\begin{pmatrix} p_i \\ p_{i+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \begin{pmatrix} c_i \\ c_{i+1} \end{pmatrix} \mod 26.$$

A classical cipher that one might glimpse in the morning paper, widely known as an *anagram*, is the *transposition cipher* (or permutation cipher). A transposition cipher consists of rearranging the positions of the elements of the message without changing the identities of the elements [Mao04]. The German ADFGVX cipher (a cipher used during World War I), is a transposition cipher that consists of substitutions as well. Though it was considered to be difficult to crack at the time, the cipher was broken by French cryptanalyst Georges Painvin [Sch94].

We present one example of a transposition cipher as follows [Mao04]:

Example 3.4.1. Consider that the elements of a plaintext message P are numbers in $\mathbb{Z}_{26}$. Let b be a fixed positive integer representing the size/length of a message, let C be our ciphertext so that $P = C = (\mathbb{Z}_{26})^b$, and let k (our key) be all permutations, i.e., rearrangements of $(1, 2, \ldots, b)$. Let $\pi \in k$, for a permutation π, such that $\pi = (\pi(1), \pi(2), ..., \pi(b))$. For a plaintext string $(x_1, x_2, ...x_b) \in P$, the encryption algorithm of this transposition cipher is

$$e_\pi(x_1, x_2, ..., x_b) = \big(x_{\pi(1)}, x_{\pi(2)}, ..., x_{\pi(b)}\big).$$

Let π^{-1} denote the inverse of π, i.e., $\pi^{-1}(\pi(i)) = i$ for $i = 1, 2, ..., b$. Then the corresponding decryption algorithm of the transposition cipher is

$$d_\pi = (y_1, y_2, ..., y_b) = \big(y_{\pi^{-1}(1)}, y_{\pi^{-1}(2)}, ..., y_{\pi^{-1}(b)}\big).$$

If the message is larger in length than b, the message is divided into multiple strings and the same procedures are repeated for each string.

Let $b = 4$ and $\pi = (\pi(1), \pi(2), \pi(3), \pi(4)) = (2, 4, 1, 3)$. Then the plaintext message

proceed meeting as agreed

is first divided into 6 strings consisting of 4 letters each:

proc eedm eeti ngas agre ed__

Looking at the first string, **proc**, and at π, we see that the letter in the first position will shift to the third position, and the letter in the second position will shift to the first position, etc. Applying π to each group of four, we get the following ciphertext:

rcpoemedeietgsnagearde

(We delete the underscores at the end of our ciphertext to avoid giving information about our key.) The decryption key for deriving the original message is:

$$\pi^{-1} = \left(\pi(1)^{-1}, \pi(2)^{-1}, \pi(3)^{-1}, \pi(4)^{-1}\right) = \left(2^{-1}, 4^{-1}, 1^{-1}, 3^{-1}\right) = (1, 2, 3, 4).$$

One can deduce that for a message of length b, there are $b!$ different keys. Therefore, a plaintext message string can be enciphered to $b!$ possible ciphertexts. However, one should know that this cipher can be easily decrypted since the identities of the letters do not change. One is able to apply frequency analysis since the characters hold the same frequencies shown in Table 2.1.

A cipher that is very similar to the Vigenère Cipher is the *Beaufort Cipher* which was developed by Admiral Sir Francis Beaufort, R.N. [Kah67]. The Beaufort Cipher enciphers text by subtracting the number corresponding to a plaintext letter from the number corresponding to a letter in our keyword. Many know of this cipher as the Vigenère cipher applied backwards. The difference will be computed modulo 26 and will be converted to its corresponding letter, thus completing the encryption process.

Example 3.4.2. Encrypting a message with the Beaufort Cipher

Keyword	W	A	R	W	A	R	W	A	R	W	A	R
Plaintext	H	O	L	D	Y	O	U	R	F	I	R	E

Keyword	22	0	17	22	0	17	22	0	17	22	0	17
Plaintext	7	14	11	3	24	14	20	17	5	8	17	4

Computing the difference of each column modulo 26, and converting our numbers back to letters, we get our ciphertext:

PMGTCDCJMOJN

The decryption process of this cipher relies on using the Vigenère square, the same square used in a Vigenère cipher. Simply take a letter of position k in a ciphertext, go down that letter's column beginning on the first row and stopping at the kth letter of the keyword, and move left to see which letter is in front of that row. This letter is in the k position in our cleartext message.

David Kahn presents a third cipher in his text, *The Codebreakers* [Kah67]. This cipher can be applied using the Vigenère square, known as the *Variant Cipher*, sometimes referred to as the *Variant Beaufort Cipher*. Kahn shows the differences between all three ciphers (using the notation P for plain, K for key, and C for cipher) [Kah67]:

	enciphering	*deciphering*
Vigenère	$P + K = C$	$C - K = P$
Variant	$P - K = C$	$C + K = P$
Beaufort	$K - P = C$	$K - C = P$

Though classical ciphers played an important role in history, they are of very little use in today's ever-evolving world of technology and computers. It once took a small ciphering machine hours to encipher a secret message. Today there are computers that are able to generate thousands of random numbers in a matter of seconds. There are also computers that can generate very nice prime numbers and have proven to be very useful in cryptology. The next chapter holds the turning point of cryptology itself—the building blocks of public-key cryptosystems.

Chapter 4

Cryptology and Computers

"Systems are organic, living creations: if people stop working on them and improving them, they die."

-Steven Levy, *Hackers: Heroes of the Computer Revolution, 1984.*

4.1 Shift Registers and Pseudorandom Sequences

As cryptologists depended more on machines to ease the monotonous task of encryption and decryption, changes were made in terms of representing the English alphabet. Converting each letter to a sequence of *bits*, 0's and 1's, made ciphering more efficient through means of computers, and also consolidated the number of operations used during encryption. American Standard Code for Information Interchange, commonly known as *ASCII*, translated the English alphabet into bits and was the universal language between the first computers. (Table 4.1 shows some ASCII binary representations [Beu94].)

The single operation used to encrypt and decrypt follows [Beu94]: Let $a_1, a_2, ..., a_i$ be bits from a given cleartext/message, and $k_1, k_2, ..., k_i$ be bits from our known key. Then our ciphertext is obtained from $a_1 \oplus k_1$, $a_2 \oplus k_2$,...,$a_i \oplus k_i$, where $\oplus$ represents bitwise addition modulo 2. So,

$$1 \oplus 1 = 0 \oplus 0 = 0 \quad \text{and} \quad 1 \oplus 0 = 0 \oplus 1 = 1.$$

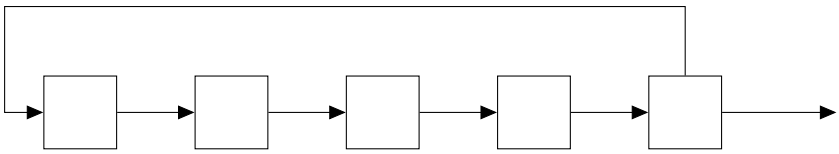
For additional security, cryptologists prefer the key to be derived from a pseudorandom sequence. These sequences may be generated by *shift registers*. Shift registers,

ASCII character	binary representation	ASCII character	binary representation
A	01000001	N	01001110
B	01000010	O	01001111
C	01000011	P	01010000
D	01000100	Q	01010001
E	01000101	R	01010010
F	01000110	S	01010011
G	01000111	T	01010100
H	01001000	U	01010101
I	01001001	v	01010110
J	01001010	W	01010111
K	01001011	X	01011000
L	01001100	Y	01011001
M	01001101	Z	01011010

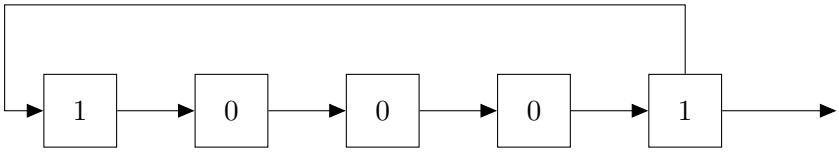
Table 4.1: Some ASCII Binary Representations

for cryptological purposes, are represented as cells that each hold one bit. Example 4.1.1 shows a shift register of length 5, however, shift registers can range from 2 to hundreds of cells in length.

Example 4.1.1. A shift register of length 5



If we put an initial sequence of bits into this particular shift register, for example, 10001, after 5 shifts we return to the initial state of the sequence.

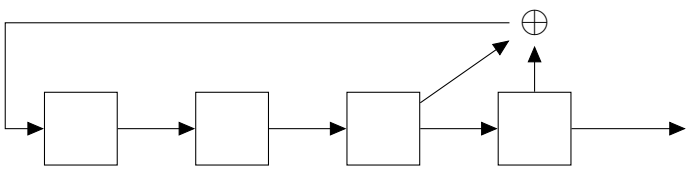


cells	5	4	3	2	1
initial state	1	0	0	0	1
after 1st shift	1	1	0	0	0
after 2nd shift	0	1	1	0	0
after 3rd shift	0	0	1	1	0
after 4th shift	0	0	0	1	1
after 5th shift	1	0	0	0	1

The initial state is chosen by the user and can be any combination of 0's and 1's except the *zero state*, which is a string of only 0's. This is because the zero state is permanent and each shift will produce only another sequence of 0's. The key consists only of bits that are being outputted from the first (far right) cell with each shift. Therefore the key using this particular shift register with the initial state of 10001 would be 10001, and may be repeated if the cleartext is longer than the key. Coincidentally, from the example above, the output happens to be the same as the initial state of the bits inserted into the shift register. This will not always be the case, and one will not always obtain such a short key with which to work. For example, an initial state of 11010 would produce the key 01011 using the above shift register.

Linear feedback shift registers (LFSR) are a little more complex as they make it more difficult to detect a pattern in a sequence. They are also more likely to supply a longer key than an average shift register. Unlike Example 4.1.1, LFSR allow the last cell to receive input from more than one cell. This value, which is fed back into the last cell, is called the feedback.

Example 4.1.2. Linear feedback shift register of length 4



If we input in an initial sequence of 1001, the sequences that follow would be:

cells	4	3	2	1
initial state	1	0	0	1
after 1st shift	1	1	0	0
after 2nd shift	0	1	1	0
after 3rd shift	1	0	1	1
after 4th shift	0	1	0	1
after 5th shift	1	0	1	0
after 6th shift	1	1	0	1
after 7th shift	1	1	1	0
after 8th shift	1	1	1	1
after 9th shift	0	1	1	1
after 10th shift	0	0	1	1
after 11th shift	0	0	0	1
after 12th shift	1	0	0	0
after 13th shift	0	1	0	0
after 14th shift	0	0	1	0
after 15th shift	1	0	0	1

The repeating output/key may now be read off the right-most column: 100110101111000....

Because these pseudorandom sequences are finite, there is a limit to how many bits can be in a single cycle. The maximum length of every cycle is $2^n - 1$, where n is the number of bits in the initial sequence. We subtract 1 from 2^n possible combinations of 0's and 1's because the zero state cannot be produced unless we had started with the zero state as our initial sequence; therefore, we exclude this combination. So when considering the LFSR in the previous example, since the initial sequence had $n = 4$ bits, the corresponding cycle could not have consisted of more than 15 bits. Since this particular LFSR yields a cycle of maximal length, it is considered to be a maximal LFSR.

Another way to represent a unique LFSR is through polynomials, defined as *characteristic polynomials*. The characteristic polynomial of an LFSR is derived from the *taps* of the LFSR, which are the positions of the LFSR that influence the next state of the sequence. A characteristic polynomial that represents an LFSR that outputs a cycle of maximal length is called a *primitive polynomial*.

Positions of the cells are considered when obtaining the characteristic polynomial and are ordered differently than the cell number. Though *cell 1* is the rightmost cell,

the cell in the *first position* would be the leftmost cell. Therefore, the characteristic polynomial (that is also a primitive polynomial) for Example 4.1.2 would be $1 + x^3 + x^4$, since the third and fourth positions of the LFSR influence the bit in the first position in the upcoming shift. If we look at the sequences of the 15 shifts above, after the 1st shift, Cell 4 holds 1 because this is the sum of Cell 2 and Cell 1 from the previous state. The bits that were in Cells $4, 3,$ and 2 in the initial state have moved one cell over after the 1st shift and now reside in Cells $3, 2,$ and 1, respectively. (One can find a list of primitive polynomials modulo 2 in [Sch94].)

One method for deriving the characteristic polynomial of a given sequence starts with first considering the possibilities of the length of the shift register. We know that for any LFSR of length n, the period cannot exceed $2^n - 1$. So, if we let p be the period of a given sequence, then $2^n - 1 \geq p$. This necessary condition helps us eliminate some possibilities for n. Once we reach a possible n, we can use the method in the following example to see if this n is suitable to determine a possible characteristic polynomial.

Example 4.1.3. Deriving a Characteristic Polynomial

Given the output sequence $1, 1, 0, 1, 0, 0, 1, 0$, with a period of 8, find the characteristic polynomial with the lowest possible degree.

Since $2^n - 1 \geq 8$, we see $n \neq 2$ and $n \neq 3$. We first check if $n = 4$. The given output sequence contains the bits from all the shifts that are positioned in the last cell:

$$\begin{array}{l} \_\_\ \_\_\ \_\_\ 1 \\ \_\_\ \_\_\ \_\_\ 1 \\ \_\_\ \_\_\ \_\_\ 0 \\ \_\_\ \_\_\ \_\_\ 1 \\ \_\_\ \_\_\ \_\_\ 0 \\ \_\_\ \_\_\ \_\_\ 0 \\ \_\_\ \_\_\ \_\_\ 1 \\ \_\_\ \_\_\ \_\_\ 0 \end{array}$$

Since every shift contains 3 consecutive bits that were in consecutive positions in the previous shift, we can input the missing cells by backtracking the bits from the bottom to the top:

_ _ 11	_ 011	1011
_ _ 01	_ 101	0101
_ _ 10	_ 010	0010
_ _ 01	_ 001	1001
_ _ 00	_ 100	0100
_ _ 10	_ 010	_010
_ _ 01	_ _01	_ _01
_ _ _0	_ _ _0	_ _ _0

Let us consider the bits in the first five shifts. We know the characteristic polynomial represents cells that influence the left-most cell in the next shift. Since the bit in the left-most cell in the second shift is 0, we can test sums of cells in the first shift to see what gives us 0.

Consider the first two shifts:

1 0 1 1

0 1 0 1

Let c_1, c_2, c_3, and c_4 represent the bits in the first shift. Then $c_1 \oplus c_3 = c_1 \oplus c_4 = c_3 \oplus c_4 = 0$ and $c_1 \oplus c_2 \oplus c_3 = c_1 \oplus c_2 \oplus c_4 = c_2 \oplus c_3 \oplus c_4 = 0$. We next attempt these combinations on the bits in the second shift to see if we get the bits in the third.

Consider the second and third shifts:

0 1 0 1

0 0 1 0

We now let c_1, c_2, c_3,and c_4 represent the bits in the second shift and look for a combination of cells to get a binary sum of 0. The only possible combinations using those from above are $c_1 \oplus c_3$, $c_2 \oplus c_3 \oplus c_4$, and $c_1 \oplus c_2 \oplus c_4$. We try the third and fourth shifts with the remaining three combinations:

$$0\ 0\ 1\ 0$$

$$1\ 0\ 0\ 1$$

The only sums that equal 1 are $c_1 \oplus c_3$ and $c_2 \oplus c_3 \oplus c_4$. Finally, we see that the remaining two combinations do not satisfy the bits in the next two shifts:

$$1\ 0\ 0\ 1$$

$$0\ 1\ 0\ 0$$

Both $c_1 \oplus c_3 \neq 0$ and $c_2 \oplus c_3 \oplus c_4 \neq 0$. Therefore, $n \neq 4$.

We next use the same backtracking method for the case $n = 5$ and obtain the following sequences:

$$0\ 1\ 0\ 1\ 1$$

$$0\ 0\ 1\ 0\ 1$$

$$1\ 0\ 0\ 1\ 0$$

$$0\ 1\ 0\ 0\ 1$$

$$\_\ 0\ 1\ 0\ 0$$

$$\_\ \_\ 0\ 1\ 0$$

$$\_\ \_\ \_\ 0\ 1$$

$$\_\ \_\ \_\ \_\ 0$$

Let c_1, c_2, c_3, c_4, and c_5 represent the bits in the first shift:

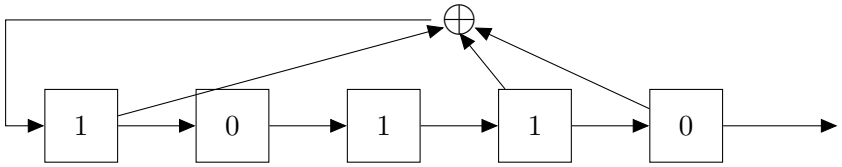
$$0\ 1\ 0\ 1\ 1$$

$$0\ 0\ 1\ 0\ 1$$

The possible combinations of cells to get 0 are:

$$c_2 \oplus c_4 = c_2 \oplus c_5 = c_4 \oplus c_5 = c_1 \oplus c_3 = 0,$$
$$c_1 \oplus c_2 \oplus c_4 = c_1 \oplus c_2 \oplus c_5 = c_1 \oplus c_4 \oplus c_5 = 0, \text{ and}$$
$$c_1 \oplus c_2 \oplus c_3 \oplus c_4 = c_1 \oplus c_2 \oplus c_3 \oplus c_5 = c_1 \oplus c_3 \oplus c_4 \oplus c_5 = 0.$$

Following the same method as before, and reducing the possible combinations, we find the remaining combination $c_1 \oplus c_4 \oplus c_5$ satisfies all shifts and gives us the appropriate output sequence. This confirms that $n = 5$ and the LFSR with its initial sequence looks like:



The characteristic polynomial of this LFSR is $1 + x + x^4 + x^5$.

Though a maximal LFSR provides some security, to ensure its full efficiency, the period should also be reasonably long. A longer period and/or cycle strengthens the defense against an attack on a ciphertext, but additional precautions may be carried out. One can implement an additional LFSR where outputs of both LFSRs are combined. Another possibility would be to use an additional LFSR to control the output from the first LFSR. A method of using two LFSRs in such a way is known as the *Shrinking Generator*. The shrinking generator method was published in 1993 by Don Coppersmith, Hugo Krawczyk, and Yishay Mansour. These three IBM researchers also published various properties and theorems held by this special combination of shift registers [CKM94]. The generator is demonstrated in the following figure and example.

Let LFSR A output the (binary) cycle $a = a_1, a_2, \ldots, a_n$ with period n, and LFSR B output the cycle $b = b_1, b_2, \ldots, b_m$ with period m. The shrinking generator combines the LFSRs to construct a new sequence cycle z via the process shown in Figure 4.1.

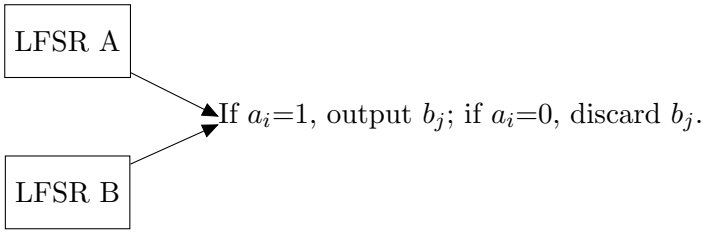


Figure 4.1: A Shrinking Generator Process

Let LFSR A be of length 2 with characteristic polynomial $1 + x + x^2$ and initial sequence 10. Let LFSR B be of length 4 with characteristic polynomial $1 + x^3 + x^4$ and initial sequence 1100. Then LFSR A will output the sequence cycle 011 with a period of 3, and LFSR B will output the sequence cycle 001101011110001 with a period of 15. Thus, the output sequence cycle for this shrinking generator is $z = 0101111001$.

4.2 User Authentication

From the previous sections, we have seen cryptology used for transferring messages and data through given algorithms and keys. We will now cover an entirely different use of cryptology in the way it is very commonly used today: user authentication. In this particular section we will introduce a procedure that verifies a message that has been received has not been altered, thus protecting and validating the integrity of the message. We will also discuss procedures that verify that the user is who they claim to be. Both are important and play a significant role in cryptology today.

Message Authentication Code, commonly abbreviated as *MAC*, runs a message through a *MAC algorithm* to verify that the message is the true message that has come from the appropriate or expected sender. We define MAC as follows:

Definition 4.1. *Let M be the set of all possible messages m, K be the set of all possible keys k, and T be the set of all possible authentication codes t. MAC is defined over (K, M, T) and a pair of algorithms (S, V) so that*

- *$S(k, m)$ returns a message authentication code $t \in T$, and*
- *$V(k, m, t)$ returns a value true or false, depending on the correctness of the received authentication code.*

This type of verification, which requires both sender and recipient to interact, is called a *protocol*. A protocol that uses a proposition of user authentication similar to MAC is the *Fiat-Shamir Protocol*, devised in 1986 by mathematicians Adi Shamir and Amos Fiat. (Shamir is also one of the founders of an algorithm we will encounter in the remaining sections of this chapter.) The Fiat-Shamir Protocol (FSP) is referred to as a *zero-knowledge protocol*, a protocol that carries the following two conditions.

- Both parties make random choices.
- The probability of impersonating and deceiving the other party is related to the number of times we repeat the process. The probability is cut in half each time.

Some observations of the FSP that were suggested by Albrecht Beautelspacher in his text *Cryptology* are [Beu94]:

- Few modulo n computations are needed.
- The verifier will eventually catch on that the "user" is not the expected user, since the probability of impersonating the appropriate user successfully is $(1/2)^t$, with t representing the number of times the FSP is carried out. After twenty rounds of the protocol, the probability is less than one in one million.
- After the process, the computer has learned nothing about the user's secret.

Now that we have defined some underlying features of the FSP, we can look at the complete process [Bar02]. We note that in real implementations, steps such as generating random numbers are automated in hardware or software; these are not usually decided by participants themselves.

Setup:

1. **Verifier:** selects two large prime numbers p and q, calculates $n = pq$, publishes n, and keeps p and q a secret.
2. **User:** selects a number s in the range 1 to $n - 1$ that is relatively prime to n, calculates $v = s^2 \bmod n$, registers v as their public identification, sends v to the verifier, and keeps the password s a secret.

Verifying the User (One Round of the FSP):

1. At a remote site from which the user wishes to log in, the **user** first generates a number r at random in the range 1 to $n - 1$, and calculates $w = r^2 \bmod n$. The user then sends w to the verifier.
2. On receiving w, the **verifier** randomly selects a bit e, either 0 or 1, and sends e to the user.

3. On receiving e, the **user** calculates $y = r \cdot s^e \bmod n$. (If $e = 0$, then $y = r$; if $e = 1$, then $y = r \cdot s \bmod n$.) The user sends y to the verifier.

4. Receiving y, the **verifier** calculates

$$z = y^2 \bmod n$$

and

$$z' = w \cdot v^e \bmod n.$$

If $z \neq z'$, the verifier refuses the login. If $z = z'$, the verifier accepts that round of the protocol and might decide to start the entire process over again to ensure that it is the appropriate user.

Since this process works well if p and q are large *blum* integers, numbers of the form $4t+3$, it is difficult to calculate what p and q might be, confirming that the FSP is a very efficient way to verify a user. However, because of the back–and–forth steps between the user and verifier, many might think of this process as a lengthy one. The next section will discuss the concept of shared and public keys, and of course the advantages and disadvantages of their application.

4.3 Public-Key Cryptosystems

So far we have introduced concepts in cryptology that require a mutual knowledge of a key between both sender and receiver. We will now consider the possibility of a third party always intercepting a cryptic exchange of private information. *Public-key cryptology* (which we will refer to as *PKC*), is a system of cryptology that consists of an asymmetric cipher used by both parties that allows each participant to create a public encryption key and a private decryption key [Sin99]. One particular cryptologist who was very interested in this concept, and thus became one of the first and biggest contributors to PKC, is Whitfield Diffie.

Diffie's concern about information security began when the very first forms of the internet were being established. He presented his idea of two parties securely exchanging their keys to IBM's Thomas J. Watson's Laboratory in 1974. Though the audience was

skeptical about Diffie's prospects, Diffie found that another person had recently presented the same ideas he had in mind. Diffie immediately sought and united with this speaker, Martin Hellman, thus forming one of the most dynamic partnerships in cryptography [Sin99].

Diffie and Hellman's solution to the key exchange problem, consisting of simple modular arithmetic, is called the Diffie-Hellman key exchange. The idea was first published in an article titled *New Directions in Cryptography* in 1976. The Diffie-Hellman key exchange consists solely of the distribution of a key between both parties without the third party obtaining a copy of the key. Thus, this protocol cannot be used to encrypt and decrypt messages.

For this protocol, Diffie thought of using the *discrete logarithm* to tie his idea together. The discrete logarithm revolves around solving for x such that $a^x \equiv b \pmod{p}$, where p is a prime integer, and a, b are nonzero positive integers. This protocol begins with two people we will encounter again in the RSA section: Alice and Bob. As the exchange of information is happening between Alice and Bob, we have Eve who is always intercepting the conversation. The protocol goes as follows [Sch94]:

1. First, Alice and Bob each agree on large integers, n and g, such that $1 < g < n$. (These two integers do not have to be secret.)

2. Alice chooses a random large integer x, which she will keep private, and computes $X = g^x \bmod n$.

3. Bob chooses a random large integer y, which he will keep private, and computes $Y = g^y \bmod n$.

4. Alice sends X to Bob, and Bob sends Y to Alice (without giving away x or y).

5. Alice computes $k = Y^x \bmod n$ and Bob computes $k' = X^y \bmod n$.

   Two conditions that improve efficiency of this protocol are:

   - The modulus n should be a prime, and large; at least 512 bits long, and
   - g should be a primitive root $\pmod{n}$.

If a third person, say Carol, were to be included, the process would just depend on adding one more value for computation, with g and n holding the same properties as before [Sch94]:

1. Alice chooses a random large integer x and computes $X = g^x \bmod n$.
2. Bob chooses a random large integer y and computes $Y = g^y \bmod n$.
3. Carol chooses a random large integer z and computes $Z = g^z \bmod n$.
4. Alice sends X to Bob, Bob sends Y to Carol, and Carol sends Z to Alice.
5. Alice computes $\widehat{Z} = Z^x \bmod n$.
6. Bob computes $\widehat{X} = X^y \bmod n$.
7. Carol computes $\widehat{Y} = Y^z \bmod n$.
8. Alice sends $\widehat{Z}$ to Bob, Bob sends $\widehat{X}$ to Carol, and Carol sends $\widehat{Y}$ to Alice. (Since there is an extra person, an extra round of computations needs to be done so that each person is able to get a copy of the key.)
9. Alice computes $k = \widehat{Y}^x \bmod n$.
10. Bob computes $k = \widehat{Z}^y \bmod n$.
11. Carol computes $k = \widehat{X}^z \bmod n$.

The obtained key, k, the mutual secret key between all 3 parties, is also equivalent to $g^{xyz} \bmod n$, and no eavesdropper can obtain or compute k, since the shared key is derived from a combination of each party's secret number. After obtaining k, Alice, Bob, and Carol can start encrypting and decrypting using k. Unless k is a very small number, it will be difficult for Eve to decipher any messages that are transferred between Alice, Bob, and Carol.

One question that might arise over this key exchange process is: How do Alice and Bob know what to do if they have never met? It must be assumed that Alice and Bob know the steps to deriving the key. Another question might be: How do they agree on an encryption process (since there are hundreds), without Eve finding out? It also

must be assumed that both parties know beforehand what method for encryption and decryption will be used after obtaining k.

After this breakthrough in cryptology, Diffie still had other things in mind; the thought of an asymmetric cipher was perplexing. The idea was to suppose Alice had a key that is solely used for encryption, but to decrypt, a different key is required. Diffie thought of this cipher as a form of computer encryption, so both keys would be two different numbers such that one would be private, known only to Alice, and the other would be public. Alice's private key is the decryption key, while her public key, to which everyone has access, is her encryption key. Alice publishes her encryption key so that everyone has access to it; if someone uses the encryption key to encrypt a message and send it to Alice, Alice can use her private key to decrypt the message.

The advantage of this system is that there is no time spent performing calculations to derive a mutual secret key. Two keys are required for each person who wishes to participate in the exchange of messages. After much effort to translate this idea into a workable cryptographic system, neither Diffie, Hellman, nor Ralph Merkle (a UC Berkeley graduate student that had become part of the Diffie-Hellman team), could derive an appropriate mathematical function. The race to find an asymmetric cipher for this idea was won by another trio of cryptology researchers: Ronald Rivest, Adi Shamir, and Leonard Adleman [Sin99].

4.4 The RSA Algorithm

Ronald Rivest was the first of the trio to stumble upon Diffie and Hellman's concept of asymmetric ciphers. Rivest persuaded two fellow MIT researchers, Leonard Adleman and Adi Shamir, that there might be some interesting mathematics in the problem [Sin99]. After spending months coming up with a *one-way function* (a function in which it is easy to go in one direction, but difficult to reverse the procedure), that satisfied the concept of an asymmetric cipher, Rivest made a breakthrough and presented it to his two colleagues. Shamir and Adleman could not find a flaw in Rivest's work, thus they felt their research was complete. (This does not necessarily mean that there is not a flaw in the RSA system; it has yet to be proven as foolproof.)

Typically, a theorem or algorithm that has multiple founders is named by the founders in alphabetical order. However, Adleman insisted that his name not be included

in the discovery the three had reached. Rivest disregarded this, including Adleman anyway, and dubbed the cipher RSA (Rivest, Shamir, Adleman) [Sin99].

Before introducing the RSA cryptosystem itself, we must first examine a few of the building blocks upon which RSA heavily relies. Since the RSA algorithm is a direct application of *Euler's theorem*, we will show the necessary definitions and steps for deriving this theorem [Beu94].

For a natural number n, we define $\phi(n)$ to be the number of positive integers less than n that are relatively prime (sharing no common factor besides 1) to n. As an example, if $n = 6$, then all the positive integers smaller than n are $1, 2, 3, 4, 5$. Since 1 and 5 are the only numbers in this group relatively prime to 6, $\phi(6) = 2$. Using this definition, we can validate the following claims:

1. If p denotes a prime number, then $\phi(p) = p - 1$.
   This claim follows immediately since $1, 2, 3, ..., p - 1$ are relatively prime to p.

2. If p and q are two distinct primes, then $\phi(pq) = (p - 1)(q - 1)$.

   We know there is a total of $pq - 1$ positive integers smaller than pq. We count how many of them are not relatively prime to pq. So, we consider the multiples of p less than pq: $p, 2p, 3p, ..., (q-1)p$, and the multiples of q less than pq: $q, 2q, 3q, ..., (p-1)q$. Since these are the only integers between 1 and $pq - 1$ that are *not* relatively prime to pq, it follows that

$$\begin{aligned}\phi(pq) &= pq - 1 - (q - 1) - (p - 1) \\ &= pq - q - p + 1 \\ &= (p - 1)(q - 1).\end{aligned}$$

Thus, we are able to formulate Euler's theorem.

Theorem 4.2. ***Euler's Theorem***

Let m and n be two relatively prime positive integers. Then $m^{\phi(n)} \pmod{n} \equiv 1$, where n is the product of two distinct primes. As we consider the second claim mentioned above, we conclude that $m^{(p-1)(q-1)} \pmod{pq} \equiv 1$, where m is relatively prime to both p and q, which are both distinct prime numbers.

Euler's theorem is a specific case from the following theorem, where $k = 1$:

Theorem 4.3. *Let p and q be distinct primes.*
(1) If a is relatively prime to pq, then

$$a^{k(p-1)(q-1)} \equiv 1 \pmod{pq},$$

where k is any integer.
(2) For any integer a,

$$a^{k(p-1)(q-1)+1} \equiv a \pmod{pq},$$

where k is any positive integer.

Another concept that is very helpful when applying RSA is the *Euclidean algorithm*. This algorithm makes it easy to produce the private and public key of any participant. Euclid described this algorithm in his book, *Elements* (Book 7, Proposition 1 & 2), written around 300 B.C. However, historians believe the algorithm existed years earlier [Sch94].

The objective of the Euclidean algorithm is to compute the greatest common divisor of two given positive integers a and b, where $a \geq b$. With two reasonably small numbers, we can determine the greatest common divisor by factoring. However, to save time and tedious work, this algorithm makes it easier to calculate the greatest common divisor of two very large numbers.

The algorithm works by taking the larger number a, and dividing it by the smaller number b, which does not take too much effort. We then divide b by the remainder from the previous step, and the process repeats, until no remainder is left.

The following definition of Euclid's algorithm comes from Donald Knuth's text [Knu97]:

Definition 4.4. *(Euclid's Algorithm).*
Given two positive integers m and n, find their greatest common divisor, that is, the largest positive integer that evenly divides both m and n.

1. *Find Remainder: Divide m by n and let r be the remainder. (We will have $0 \leq r < n$.)*
2. *Is $r = 0$?: If $r = 0$, the algorithm terminates; n is the greatest common divisor.*
3. *Reduce: Set $m \leftarrow n, n \leftarrow r$, and go back to Step 1.*

The proof for this algorithm can be found in [Knu97].

Let us consider an example from [Beu94], where we compute the greatest common divisor of a and b, where $a = 792$ and $b = 75$:

Example 4.4.1. Applying the Euclidean Algorithm

$$\begin{aligned} 792 &= 10 \cdot 75 + 42 \\ 75 &= 1 \cdot 42 + 33 \\ 42 &= 1 \cdot 33 + 9 \\ 33 &= 3 \cdot 9 + 6 \\ 9 &= 1 \cdot 6 + 3 \\ 6 &= 2 \cdot 3 + 0. \end{aligned}$$

By the Euclidean algorithm, 3 is the greatest common divisor of 792 and 75.

Since we have described the two concepts that are necessary for applying the RSA algorithm, we may proceed by defining the steps of the RSA cryptosystem.

The RSA Cryptosystem [Mao04]:

Key Setup

To set up Alice's key, Alice preforms the following steps.

1. Choose two large, random prime numbers p and q such that $|p| \approx |q|$.
2. Compute $n = pq$.
3. Compute $\phi(n) = (p-1)(q-1)$.
4. Choose a random integer $e < \phi(n)$ such that $\gcd(e, \phi(n)) = 1$, and compute an integer d such that $ed \equiv 1 \pmod{\phi(n)}$.
5. Publish (n, e) as the public key, safely destroy p, q, and $\phi(n)$, and keep d as the private key.

Encryption

To send a confidential message $m < n$ to Alice, the sender Bob creates the ciphertext c as follows: $c \leftarrow m^e \bmod n$.

Decryption

To decrypt the ciphertext c, Alice computes: $m \leftarrow c^d \bmod n$.

The question arises: How do we know that the m we get after decryption will be the same m that was encrypted? We present the proof of why RSA works using Theorem (4.3) [Bar02]:

Proof. In this proof, we will have m represent the product of the two primes, and x be the message Bob wants to send to Alice. To prepare for receiving encrypted messages, Alice selects two prime numbers p and q and calculates

$$m = pq \quad \text{and} \quad n = (p-1)(q-1).$$

Next, she selects another number e that is relatively prime to n. She then uses the Eucldean algorithm to find d such that $e \cdot d \equiv 1 \pmod{n}$. Alice broadcasts e and m. Now, if Bob wants to send x to Alice, with x in the range 0 to $m-1$, he calculates the corresponding ciphertext as

$$y = x^e \bmod m.$$

Bob sends y to Alice. Upon receiving y, Alice calculates

$$y^d \bmod m.$$

Now, by the definition of y, this gives

$$\begin{aligned} y^d &\equiv (x^e \bmod m)^d \\ &\equiv (x^e)^d \\ &\equiv x^{ed} \pmod{m}. \end{aligned}$$

Since $d \equiv e^{-1} \pmod{n}$, by the definition of multiplicative inverse modulo n we know that $d \cdot e = 1 + kn = 1 + k(p-1)(q-1)$ for some integer k. So by Theorem (4.3),

$$y^d \equiv x^{ed} \equiv x^{1+k(p-1)(q-1)} \equiv x \pmod{n}.$$

That is, Alice recovers Bob's original message.

□

The following is a straightforward example using RSA that Wenbo Mao provides in his text *Modern Cryptology: Theory & Practice*:

Example 4.4.2.

Let Alice set $n = 7 \cdot 13 = 91$ and $e = 5$. Then $\phi(n) = 6 \cdot 12 = 72$. After applying the Euclidean algorithm, Alice derives $5 \cdot 29 \equiv 1 \pmod{72}$. Therefore, 29 will be Alice's key for decryption. She publishes $(n, e) = (91, 5)$ as her public key.

Now, let Bob encrypt a plaintext $m = 3$. So Bob encrypts m by computing

$$c = 3^5 = 243 \equiv 61 \pmod{91}.$$

Therefore the resulting ciphertext message is 61. To decrypt this ciphertext, Alice computes

$$61^{29} \bmod 91 = 3,$$

thus retrieving the original plaintext, 3.

RSA gets its security from the difficulty of factoring large numbers whose factors alone consist of at least 100 to 200 digits [Sch94]. To ensure maximum security in our cryptosystem, we consider two conditions [Sch94]:

- A common modulus should not be used.
- The exponents chosen should not be small.

There are many interpretations of the RSA algorithm that other cryptologists or cryptology enthusiasts have created, including the RSA key exchange, the Rabin Cryptosystem, and the ElGamal Cryptosystem (which is more of an application of the Diffie-Hellman Key exchange). There are also many current applications that strongly rely on the RSA algorithm, mainly internet security.

4.5 An Insecure Public-key Cryptosystem: The Knapsack Cryptosystem

As the Diffie-Hellman algorithm was just starting to be recognized for its innovation in cryptology, Ralph Merkle had been working on another public key approach [Lev01]. The system Merkle had in mind was built around a mathematical problem known as a *knapsack problem*. In general, one solves the knapsack problem by finding a way to optimally fill the knapsack. One looks for a way to select the "best" items to put in the knapsack in such a way that the knapsack becomes neither overfilled nor underfilled.

Merkle had envisioned this system by considering a *trapdoor one-way function* where the right person could easily obtain the solution, but it would remain exceedingly difficult for other parties to do so. Merkle had worked with Marty Hellman to devise such a system. To ensure that this cryptosystem would be secure, Merkle posted a challenge on his office door offering $100 to the first person to break it. The first person to break the "easier" version of the challenge, a *single-iteration knapsack scheme*, was one of the RSA founders, Leonard Adleman. Merkle then decided to offer $1000 to decode a multiple-iteration knapsack. In just after two years, Merkle wrote a check to researcher Ernie Brickell from Sandia National Laboratory [Lev01].

The knapsack problem may be stated as follows [Bur11]: Given a knapsack of volume V and n items of various volumes $a_1, a_2, ..., a_n$, we want to see if there exists a subset of these items that will completely fill the knapsack. An alternative formulation: For positive integers $a_1, a_2, ..., a_n$, and a sum V, solve the equation

$$V = a_1x_1 + a_2x_2 + ... + a_nx_n,$$

where $x_i = 0$ or 1 for $i = 1, 2, ..., n$. The number of possible solutions depends on the choice of the sequence $a_1, a_2, ..., a_n$ and the integer V. For example [Bur11], the knapsack problem

$$22 = 3x_1 + 7x_2 + 9x_3 + 11x_4 + 20x_5$$

is not solvable, but the problem

$$27 = 3x_1 + 7x_2 + 9x_3 + 11x_4 + 20x_5$$

has two distinct solutions:

$$x_2 = x_3 = x_4 = 1;\ x_1 = x_5 = 0$$

and

$$x_2 = x_5 = 1;\ x_1 = x_3 = x_4 = 0.$$

These are straightforward examples, but finding a solution to a randomly chosen knapsack problem may be much more difficult. We may choose to conduct an exhaustive direct search by testing all 2^n possibilities for $x_1, x_2, ..., x_n$, but this method becomes impractical for n greater than 100. However, what if the sequence of integers $a_1, a_2, ..., a_n$ has a special property? A property that makes the knapsack problem much easier to solve is to use a *superincreasing sequence* $a_1, a_2, ..., a_n$. Such a sequence is defined by the property that

$$a_i > a_1 + a_2 + ... + a_{i-1}, \qquad i = 2, 3, ..., n.$$

Knapsack problems based on superincreasing sequences are uniquely solvable, if there exists a solution at all. We observe this in the following example [Bur11]:

Example 4.5.1.
Solving the superincreasing knapsack problem:

$$28 = 3x_1 + 5x_2 + 11x_3 + 20x_4 + 41x_5.$$

We apply the following steps starting with the largest coefficient in this equation: 41.

1. Since $41 > 28$, it cannot be part of our subset sum; hence $x_5 = 0$.

2. We then consider the next largest coefficient: 20. Now $20 < 28$, and the sum of preceding coefficients is $3 + 5 + 11 < 28$, so these 3 items alone cannot fill the knapsack; therefore 20 must be included in the sum, and so $x_4 = 1$.

3. Since $x_4 = 1$ and $x_5 = 0$, we may substitute these values into the equation to obtain:

$$\begin{aligned} 28 &= 3x_1 + 5x_2 + 11x_3 + 20(1) + 41(0) \\ 28 &= 3x_1 + 5x_2 + 11x_3 + 20 \\ 8 &= 3x_1 + 5x_2 + 11x_3. \end{aligned}$$

Repeating Step 1: Since $11 > 8$, $x_3 = 0$, thus reducing our equation to $8 = 3x_1 + 5x_2$, which has an obvious unique solution: $x_1 = x_2 = 1$. This identifies the subset of $\{3, 5, 11, 20, 41\}$ having the desired sum:

$$28 = 3 + 5 + 20.$$

A generalization of this process may be defined as follows [Bur11]. Suppose that we wish to solve the knapsack problem

$$V = a_1x_1 + a_2x_2 + ... + a_nx_n,$$

where $a_1, a_2, ..., a_n$ is a superincreasing sequence of integers. Assume that V can be obtained by using some subset of the sequence, so, in particular, V is not larger than the sum $a_1 + a_2 + ... + a_n$. Working from right to left in the sequence, we begin by letting $x_n = 1$ if $V \geq a_n$, and $x_n = 0$ if $v < a_n$. Then obtain $x_{n-1}, x_{n-2}, ..., x_1$, in turn, by choosing

$$x_i = \begin{cases} 1 & \text{if } V - (a_{i+1}x_{i+1} + ... + a_nx_n) \geq a_i, \\ 0 & \text{if } V - (a_{i+1}x_{i+1} + ... + a_nx_n) < a_i. \end{cases}$$

Note that any solution obtained from this procedure must be unique.

The application of knapsacks in Merkle and Hellman's cryptosystem functions as follows:

Setting up the public key for a sender to use

1. A user starts by choosing a superincreasing sequence $a_1, a_2, ..., a_n$.

2. The user selects a modulus $m > 2a_n$ and a multiplier a, with $0 < a < m$ and $\gcd(a, m) = 1$. This property ensures that the congruence $ax \equiv 1 \pmod{m}$ has a unique solution, say $x = c \bmod m$.

3. The user forms the sequence of integers $b_1, b_2, ..., b_n$ defined by

$$b_i \equiv aa_i \pmod{m}, \quad i = 1, 2, ..., n,$$

   where $0 < b_i < m$. Carrying out this last transformation destroys the superincreasing property we saw in a_i.

4. The user keeps secret the original sequence $a_1, a_2, ..., a_n$, and the numbers m and a, but publicizes $b_1, b_2, ..., b_n$ in a public directory.

A sender using the public key b_i to encrypt a message and send to the user

1. A sender must use the b_i's as the encryption key in order to send the user a message. The sender begins by converting their plaintext message into a string M of 0's and 1's using the binary equivalent of letters (see Fig. 4.1).

2. The string is then split into blocks of n binary digits, with the last block being filled with 1's at the end, if necessary.

3. The public encrypting sequence $b_1, b_2, ..., b_n$ is next used to transform a given plaintext block, say $x_1, x_2, ..., x_n$, into the sum

$$S = b_1x_1 + b_2x_2 + ... + b_nx_n. \tag{4.1}$$

   The number S is then sent to the user over a communication channel, which is presumed to be insecure.

The user decrypts S

Recall that c is a number such that $ac \equiv 1 \pmod{m}$. Knowing c and m, the user derives

$$S' = cS \bmod m. \tag{4.2}$$

Using (4.1) to expand the above congruence, we obtain

$$\begin{aligned} S' &= cb_1x_1 + cb_2x_2 + ... + cb_nx_n \bmod m \\ &= caa_1x_1 + caa_2x_2 + ... + caa_nx_n \bmod m. \end{aligned}$$

Now $ca \equiv 1 \pmod{m}$, therefore the last congruence simplifies to

$$S' = a_1x_1 + a_2x_2 + ... + a_nx_n \bmod m.$$

Because $m > 2a_n > a_1 + a_2 + ... + a_n$, we obtain $a_1x_1 + a_2x_2 + ... + a_nx_n < m$; and in light of the condition $0 \leq S' < m$, the equality

$$S' = a_1x_1 + a_2x_2 + ... + a_nx_n$$

must hold. Thus, the solution to this superincreasing knapsack problem leads us to the solution of the more difficult problem (4.1), and the plaintext block $x_1, x_2, ..., x_n$ of n digits is thereby recovered from S. Let us consider a small-scale example with $n = 5$.

Example 4.5.2.
Recall the important conditions for a and m: $m > 2a_n$; $0 < a < m$; and $\gcd(a, m) = 1$. Since $n = 5$ for this example, user A may select the superincreasing sequence $\{a_i\} = \{3, 5, 11, 20, 41\}$, and choose the multiplier $a = 44$ and the modulus $m = 85$. Each a_i is then multiplied by 44 and reduced modulo 85, yielding the public key sequence $\{b_i\} = \{47, 50, 59, 30, 19\}$. Sender B wants to send the message *UNITE* to User A. (For this example, we will convert these letters into a smaller binary representation than ASCII code format for the sake of conciseness.) Sender B converts his plaintext into the following string of 0's and 1's:

$$M = 10100 \quad 01101 \quad 01000 \quad 10011 \quad 00100$$

The string is then divided into blocks of digits, in the current case, blocks of length 5 (since each binary form of the plaintext letters is of length 5), and multiplied with the binary digits that are part of the binary representations of the plaintext letters. Using the public key to encrypt, the sender transforms the successive blocks (containing our plaintext in binary form) into:

$$\begin{aligned}
106 &= 47 \cdot 1 + 50 \cdot 0 + 59 \cdot 1 + 30 \cdot 0 + 19 \cdot 0 \\
128 &= 47 \cdot 0 + 50 \cdot 1 + 59 \cdot 1 + 30 \cdot 0 + 19 \cdot 1 \\
50 &= 47 \cdot 0 + 50 \cdot 1 + 59 \cdot 0 + 30 \cdot 0 + 19 \cdot 0 \\
96 &= 47 \cdot 1 + 50 \cdot 0 + 59 \cdot 0 + 30 \cdot 1 + 19 \cdot 1 \\
59 &= 47 \cdot 0 + 50 \cdot 0 + 59 \cdot 1 + 30 \cdot 0 + 19 \cdot 0
\end{aligned}$$

Thus the transmitted ciphertext is

$$106 \quad 128 \quad 50 \quad 96 \quad 59.$$

User A is able to decrypt the sequence by first solving the congruence

$$44x \equiv 1 \pmod{85},$$

yielding $x \equiv 29 \pmod{85}$. Then each ciphertext number is multiplied by 29 and reduced modulo 85, to produce a superincreasing knapsack problem. For instance, 50 is converted to 5, because $50 \cdot 29 \equiv 5 \pmod{85}$. The corresponding knapsack problem is

$$5 = 3x_1 + 5x_2 + 11x_3 + 20x_4 + 41x_5,$$

which has the unique solution $x_2 = 1, x_1 = x_3 = x_4 = x_5 = 0$. User A repeats this process by multiplying all ciphertext numbers by 29 and reducing modulo 85 to obtain a superincreasing knapsack problem for each item in the ciphertext.

One might already feel that this cryptosystem is a bit insecure, for the only possible combinations of x_i's for the third and fifth number in our ciphertext are 01000 and 00100, thus revealing that the letters must be I and E. We see that it is easy for an outside party to decrypt a letter if one of the public b_i's appears as a number in the transmitted ciphertext. An outside party might also be able to decrypt more letters by exhausting all possible combinations by considering the properties of adding even and odd integers. For example, let us look at

$$128 = 47 \cdot 0 + 50 \cdot 1 + 59 \cdot 1 + 30 \cdot 0 + 19 \cdot 1,$$

which is known only to Sender B. An outside party that is trying to see the message being sent to User A would see only

$$128 = 47x_1 + 50x_2 + 59x_3 + 30x_4 + 19x_5,$$

and may try to decrypt x_i by a brute force attack as follows. Let us represent an even number by E, and an odd number by O. Since 128 is an even number, we know that the possible combinations of even and odd numbers that sum to 128, an even number, are

$$\begin{aligned}
E &= E & E &= E+E+E+E \\
E &= E+E & E &= O+O+O+O \\
E &= O+O & E &= E+E+E+E+E \\
E &= E+E+E & E &= O+O+E+E+E \\
E &= O+O+E & E &= O+O+O+O+E. \\
E &= O+O+E+E
\end{aligned}$$

We eliminate some combinations since we see that we have 3 odd coefficients and 2 even coefficients in our equation. Thus, the only possible combinations left to consider are

$$\begin{aligned} 128 &= E + E \\ &= O + O \\ &= O + O + E \\ &= O + O + E + E. \end{aligned}$$

The first combination will not work since the sum of the even coefficients is 80. Thus, by exhausting all remaining combinations, we obtain the correct combination:

$$128 = 50 + 59 + 19.$$

Thus, the x_i sequence for this equation must be 01101. This technique of exhausting all possible combinations of 0's and 1's for all x_i enables an outside party to attack the ciphertext with ease! If n is relatively large, then a computer might be able to perform this or any other method of exhaustion faster than the time it takes for one to select a superincreasing sequence. A more secure example where $n = 10$ can be found in [Bur11].

Many variations of the Merkle-Hellman scheme have been created; however, most have been proven to be insecure and thus not very useful for current means of cryptology.

Chapter 5

Conclusion

Throughout this paper, we learned of different ciphers, methods of encryption and decryption, and a brief history of each concept. We began by introducing one of the earliest known ciphers, the Caesar cipher. Both the Caesar cipher and the more complicated affine cipher can be deciphered by applying frequency analysis. Frequency analysis, as we learned, is an important technique that can be used in some way to decipher many ciphers, mono and polyalphabetic. Polyalphabetic ciphers were created in order to provide better security than the various, and once popular, monoalphabetic ciphers. We discussed a few polyalphabetic ciphers and techniques to decipher them, but mainly considered the famous Vigenère cipher. Chapter 3 also included a discussion of William Friedman and his many successes and contributions to cryptology. We ended this chapter with other significant classical ciphers that one might find interesting.

Chapter 4 introduced cryptology used in the ever growing era of computers. Instead of making up a key and having our key be vulnerable to attacks, shift registers allow pseudorandom keys to be created and incorporated into the cipher. However, there are a few ways one can work backwards to derive the shift register. (Section 4.1 presents only one of these ways.) In Section 4.2 we discussed user authentication, an application of cryptology that is used to verify a user. This application of cryptology (that is the basis of many protocols and ciphers) has allowed small-time cryptologists to think outside of the box and consider the vast possibilities of cryptology.

We concluded this paper with the most significant breakthrough in cryptology: public-key cryptosystems. Whitfield Diffie's contributions to public-key cryptography

were fortuitous and revolutionary. Steven Levy writes in his book *Crypto* [Lev01], "For the first time, it became possible to conceive of all sorts of official transactions–contracts, receipts, and the like–to be performed over computer networks, with no need for one's physical presence." Cryptology was no longer limited to the government; Diffie had made it possible for anyone to be protected with encryption while performing commercial transactions. Aside from the Diffie-Hellman key exchange, Diffie was able to formulate the concept of the RSA algorithm and inspire the founders to seek and discover a mathematical representation of Diffie's idea. The RSA algorithm opened up more possibilities for computer security and for cryptology iteself.

The last public-key cryptosystem we discussed was one of many insecure cryptosystems created: the knapsack cryptosystem. Studying this cryptosystem leaves many unanswered questions for cryptology enthusiasts alike: Could a definite secure cryptosystem be created based on this concept? Might other factors be put into consideration when deriving a public key? We were able to present one of the many methods of attack on a ciphertext encrypted under the knapsack cryptosystem.

In response to these new breakthroughs, colleges have been able to implement modern and public-key cryptology into their curriculum. However, cryptology is usually not a required course, and only available as an elective in some schools. Learning it on one's own is possible, but may require some patience and many sources for various explanations. Learning the many forms of cryptology, and how it has evolved through time, shows how fast we are advancing in technology and in our problem solving techniques. Cryptology is a part of mathematical and world history, and has given humanity some control over what is rightfully ours: our secrets.

Bibliography

[Bar02] Thomas H. Barr. *Invitation to Cryptology.* Prentice Hall, Upper Saddle River, New Jersey, 2002.

[Beu94] Albrecht Beutelspacher. *Cryptology.* The Mathematical Association of America, Washington DC, 1994.

[Bur11] David M Burton. *Elementary Number Theory.* McGraw-Hill, New York, NY, 2011.

[CKM94] Don Coppersmith, Hugo Krawczyk, and Yishay Mansour. *Advances in Cryptology-Crypto '93.* Springer Berlin Heidelberg, USA, 1994.

[Kah67] David Kahn. *The Codebreakers: The Story of Secret Writing.* Macmillan, New York, 1967.

[Knu97] Donald Knuth. *The Art of Computer Programming: Fundamental Algorithms 3rd Edition.* Addison-Wesley Professional, USA, 1997.

[Lev01] Steven Levy. *Crypto: how the code rebels beat the government saving privacy in the digital age.* Penguin Group, New York, 2001.

[Lew00] Robert E. Lewand. *Cryptological Mathematics.* The Mathematical Association of America, USA, 2000.

[Mao04] Wenbo Mao. *Modern Cryptology: Theory & Practice.* Hewlett-Packard Company, Prentice Hall PTR, Upper Saddle River, New Jersey, 2004.

[Sch94] Bruce Schneier. *Applied Cryptography.* John Wiley & Sons, Inc., New York, 1994.

[Sin99] Simon Singh. *The Code Book: The Science of Secrecy From Ancient Egypt to Quantum Cryptography.* Anchor Books, New York, 1999.

[Sta10] Alexander Stanoyevitch. *Introduction to Cryptography with Mathematical Foundations and Computer Implementations.* Chapman and Hall, London, 2010.