Chapter 3

A Cryptography Primer

Scott R. Ellis
kCura Corporation, Chicago, IL, United States

"Cryptography," as a word, literally means the "study of hidden writing." It comes from the Greek κρυπτός, "hidden, secret"; and from γράφειν, *graphein*, "writing," or -λογία, *-logia*, "study."[1] In practice, it is so much more than that. The zeros and ones of compiled software binary, something that frequently requires encryption, can hardly be considered "writing." Were a new word for cryptography to be invented today, it would probably be "secret communications." It follows that, rather than point to the first altered writing as the origins of cryptography, we must look to the origins of communication and to the first known alterations of it in any form. Historically, then, you might say that cryptography is a built-in defense mechanism, as a property of language. As you will see in this chapter, ultimately this dependency is also the final, greatest weakness of any cryptographic system, even the perceivably unbreakable Advanced Encryption Standard (AES) system. From unique, cultural body language to language itself, to our every means of communication, it is in our nature to want to prevent others who would do us harm from intercepting private communications (which could be about them!). Perhaps nothing so perfectly illustrates this fact as the art of cryptography. It is, in its purpose, an art form entirely devoted to the methods whereby we can prevent information from falling into the hands of those who would use it against us: our enemies.

Since the beginning of sentient language, cryptography has been a part of communication. It is as old as language itself. In fact, one could make the argument that the desire and ability to encrypt communication, to alter a missive in such a way so that only the intended recipient may understand it, is an innate ability hard-wired into the human genome. Aside from the necessity to communicate, it could well be what led to the development of language itself. Over time, languages and dialects evolved, as we can see with Spanish, French, Portuguese, and Italian, all of which derived from Latin. People who speak French have a great deal of trouble understanding people who speak Spanish, and vice versa. The profusion of Latin cognates in these languages is undisputed, but generally speaking, the two languages are so far removed that they are not dialects but rather separate languages. But why is this? Certain abilities, such as walking, are built into our nervous systems; other abilities, such as language, are not. From Pig Latin to whispering circles to word jumbles, to languages so foreign that only the native speakers understand them, to diverse languages and finally modern cryptography, it is in our nature to keep our communications secret.

So why is language not hard-wired into our nervous system, as it is with bees, which are born knowing how to tell another bee how far away a flower is, as well as the quantity of pollen and whether there is danger present? Why do we humans not all speak the same language? The reason is undoubtedly because unlike bees, humans understand that knowledge is power, and knowledge is communicated via spoken and written words. Plus, we were not born with giant stingers with which to sting people we do not like. With the development of evolving languages innate in our genetic wiring, the inception of cryptography was inevitable.

In essence, computer-based cryptography is the art of creating a form of communication that embraces the following precepts:

- It can be readily understood by the intended recipients.
- It cannot be understood by unintended recipients.
- It can be adapted and changed easily with relatively small modifications, such as a changed passphrase or word.

1. H. Liddell, R. Scott, Greek-English Lexicon, Oxford University Press, 1984.

All artificially created lexicons, such as the Pig Latin of children, pictograph codes, gang-speak, and corporate lingo, and even the names of music albums, such as *Four Flicks*, are manners of cryptography in which real text, sometimes not so ciphered, is hidden in what appears to be plaintext. They are attempts at hidden communications.

1. WHAT IS CRYPTOGRAPHY? WHAT IS ENCRYPTION?

Ask any ancient Egyptian and he will undoubtedly define "cryptography" as the practice of burying the dead so that they cannot be found again. The Egyptians were good at it; thousands of years later, new crypts are still being discovered. The Greek root *krypt* literally means "a hidden place," and as such it is an appropriate base for any term involving cryptology. According to the *Online Etymology Dictionary*, *crypto*- as a prefix, meaning "concealed, secret," has been used since 1760, and from the Greek *graphikos*, "of or for writing, belonging to drawing, picturesque." Together, *crypto* + *graphy* would then mean "hiding place for ideas, sounds, pictures, or words." *Graph*, technically from its Greek root, is "the art of writing." "Encryption," in contrast, merely means the act of carrying out some aspect of cryptography. "Cryptology," with its *-ology* ending, is the study of cryptography. Encryption is subsumed by cryptography.

How Is Cryptography Done?

For most information technology (IT) occupations, knowledge of cryptography is a small part of a broader skill set and is generally limited to relevant applications. The argument could be made that this is why the Internet is so extraordinarily plagued with security breaches. The majority of IT administrators, software programmers, and hardware developers are barely cognizant of the power of true cryptography. Overburdened with battling the plague that they inherited, they cannot afford to devote the time or resources needed to implement a truly secure strategy. The reason, as we shall see, is that as good as cryptographers can be, for every cryptographer there is a decryptographer working just as diligently to decipher a new encryption algorithm.

Traditionally, cryptography has consisted of any means possible whereby communications may be encrypted and transmitted. This could be as simple as using a language with which the opposition is not familiar. Who has not been in a place where everyone around them was speaking a language they did not understand? There are thousands of languages in the world; nobody can know them all. As was shown in World War II, when the Allied forces used Navajo as a means of communicating freely, some languages are so obscure that an entire nation may not contain one person who speaks it! All true cryptography is composed of three parts: a cipher, an original message, and the resultant encryption. The *cipher* is the method of encryption used. Original messages are referred to as *plaintext* or as *clear text*. A message that is transmitted without encryption is said to be sent "in the clear." The resultant message is called a *ciphertext* or *cryptogram*. This part of the chapter begins with a simple review of cryptography procedures and carries them through; each section builds on the next to illustrate the principles of cryptography.

2. FAMOUS CRYPTOGRAPHIC DEVICES

The past few hundred years of technical development and advances have brought greater and greater means to decrypt, encode, and transmit information. With the advent of the most modern warfare techniques and the increase in communication and ease of reception, the need for encryption has never been more urgent.

World War II publicized and popularized cryptography in modern culture. The Allied forces' ability to capture, decrypt, and intercept Axis communications is said to have hastened the end of the war by several years. Next, we take a quick look at some famous cryptographic devices from that era.

The Lorenz Cipher

The Lorenz cipher machine was an industrial-strength ciphering machine used in teleprinter circuits by the Germans during World War II. Not to be confused with its smaller cousin, the Enigma machine, the Lorenz cipher could possibly best be compared to a virtual private network tunnel for a telegraph line, only it was not sending Morse code, it was using a code like a sort of American Standard Code for Information Interchange (ASCII) format. A granddaddy of sorts, called the Baudot code, was used to send alphanumeric communications across telegraph lines. Each character was represented by a series of 5 bits.

The Lorenz cipher is often confused with the famous Enigma, but unlike the Enigma (which was a portable field unit), the Lorenz cipher could receive typed messages, encrypt them, and send them to another distant Lorenz cipher, which would then decrypt the signal. It used a pseudorandom cipher XOR'd (an encryption algorithm) with plaintext. The machine would be inserted inline as an attachment to a Lorenz teleprinter. Fig. 3.1 is a rendered drawing from a photograph of a Lorenz cipher machine.

Enigma

The Enigma machine was a field unit used in World War II by German field agents to encrypt and decrypt messages

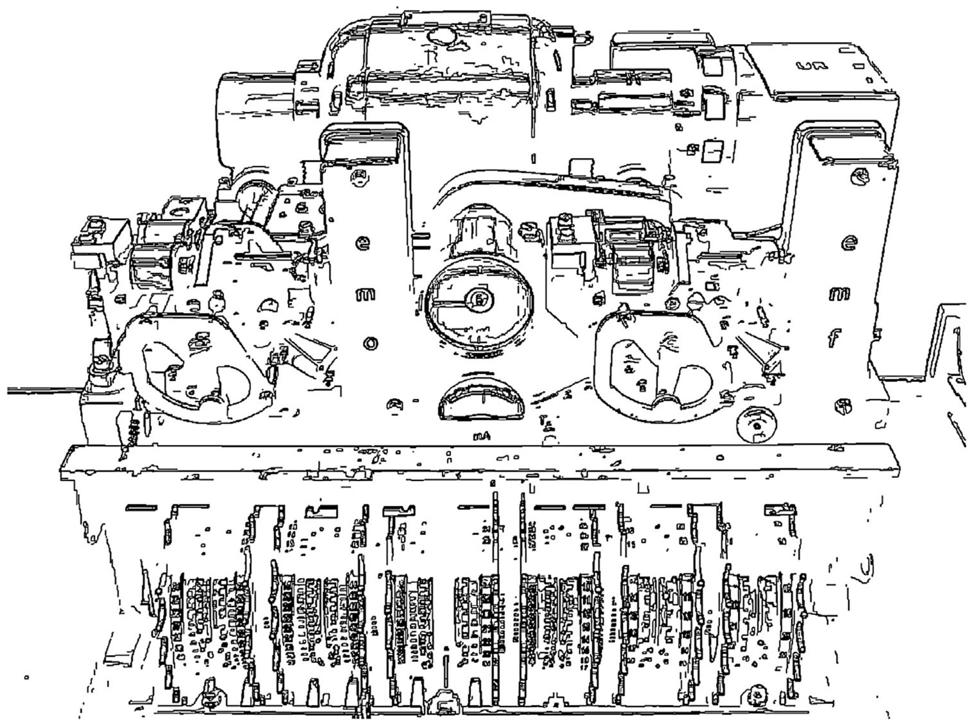


FIGURE 3.1 The Lorenz machine was set inline with a teletype to produce encrypted telegraphic signals.

and communications. Similar to the Feistel function of the 1970s, the Enigma machine was one of the first mechanized methods of encrypting text using an iterative cipher. It employed a series of rotors that, with some electricity, a light bulb, and a reflector, allowed the operator to either encrypt or decrypt a message. The original position of the rotors, set with each encryption and based on a prearranged pattern that in turn was based on the calendar, allowed the machine to be used even if it was compromised.

When the Enigma was in use, with each subsequent key press, the rotors would change in alignment from their set positions in such a way that a different letter was produced each time. With a message in hand, the operator would enter each character into the machine by pressing a typewriter-like key. The rotors would align and a letter would then illuminate, telling the operator what the letter *really* was. Likewise, when enciphering, the operator would press the key and the illuminated letter would be the cipher text. The continually changing internal flow of electricity that caused the rotors to change was not random, but it created a polyalphabetic cipher that could be different each time it was used.

3. CIPHERS

Cryptography is built on one overarching premise: the need for a cipher that can be used reliably and portably to encrypt text so that through any means of cryptanalysis (differential, deductive, algebraic, or the like) the ciphertext cannot be undone with available technology. Throughout the centuries, there have been many attempts to create simple ciphers that can achieve this goal. With the exception of the one-time pad, which is not particularly portable, success has been limited. Let us look at a few of these methods.

The Substitution Cipher

In this method, each letter of the message is replaced with a single character. Table 3.1 shows an example of a substitution cipher. Because some letters appear more often and certain words are used more than others, some ciphers are extremely easy to decrypt and can be deciphered at a glance by more practiced cryptologists.

Simply by understanding probability and employing some applied statistics, certain metadata about a language can be derived and used to decrypt any simple one-for-one substitution cipher. Decryption methods often rely on understanding the context of the *ciphertext*. What was encrypted: business communication? Spreadsheets? Technical data? Coordinates? For example, using a hex editor and an access database to conduct some statistics, we can use the information in Table 3.2 to gain highly specialized knowledge about the data in Chapter 40, “Cyber Forensics,” by Scott R. Ellis, in this book. A long chapter at nearly 25,000 words, it provides a sufficiently large statistical pool to draw some meaningful analyses.

Table 3.3 gives additional data about the occurrence of specific words in Chapter 40. Note that because it is a technical text, words such as “computer,” “files,” “email,”

TABLE 3.1 **Simple Substitution Cipher**

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

Letters are numbered by their order in the alphabet, to provide a numeric reference key. To encrypt a message, the letters are replaced, or substituted, by the numbers. This is a particularly easy cipher to reverse.

TABLE 3.2 Statistical Data of Interest in Encryption

Character Analysis	Count
Number of distinct alphanumeric combinations	1958
Distinct characters	68
Number of four-letter words	984
Number of five-letter words	1375

An analysis of a selection of a manuscript (in this case, the preedited version of Chapter 40 of this book) can provide insight into the reasons why good ciphers need to be developed.

TABLE 3.3 Five-Letter Word Recurrences in Chapter 40

Words Field	Number of Recurrences
files	125
drive	75
there	67
email	46
these	43
other	42
about	41
where	36
would	33
every	31
court	30
their	30
first	28
using	28
which	24
could	22
table	22
after	21
image	21
don't	19
tools	19
being	18
entry	18

A glimpse of the leading five-letter words found in the preedited manuscript. Once unique letter groupings have been identified, substitution, often by trial and error, can result in a meaningful reconstruction that allows the entire cipher to be revealed.

and "drive" emerge as leaders. Analysis of these leaders can reveal individual and paired alpha frequencies. Being armed with knowledge about the type of communication can be beneficial in decrypting it.

Further information about types of data being encrypted includes word counts by the length of words. Table 3.4 contains such a list for Chapter 40. This information can be used to begin to piece together useful and meaningful short sentences, which can provide cues to longer and more

TABLE 3.4 Leaders by Word Length in the Preedited Manuscript for Chapter 40

Words Field	Number of Duplications	Word Length
XOriginalArrivalTime:	2	21
interpretations	2	15
XOriginatingIP:	2	15
electronically	4	14
investigations	5	14
interpretation	6	14
reconstructing	3	14
irreproducible	2	14
professionally	2	14
inexperienced	2	13
demonstrative	2	13
XAnalysisOut:	8	13
steganography	7	13
understanding	8	13
certification	2	13
circumstances	8	13
unrecoverable	4	13
investigation	15	13
automatically	2	13
admissibility	2	13
XProcessedBy:	2	13
administrator	4	13
determination	3	13
investigative	3	13
practitioners	2	13
preponderance	2	13
intentionally	2	13
consideration	2	13
interestingly	2	13

The context of the clear text can make the cipher less secure. After all, there are only a finite number of words. Few of them are long.

complex structures. It is exactly this sort of activity that good cryptography attempts to defeat.

If it was encrypted using a simple substitution cipher, a good start to deciphering Chapter 40 could be made using the information we have gathered. As a learning exercise, game, or logic puzzle, substitution ciphers are useful. Some substitution ciphers that are more elaborate can be just as difficult to crack. Ultimately, though, the weakness behind a substitution cipher is that the ciphertext remains a one-to-one, directly corresponding substitution; ultimately, anyone with a pen and paper and a large enough sample of the ciphertext can defeat it. Through use of a computer, deciphering a simple substitution cipher becomes child's play.

The Shift Cipher

Also known as the Caesar cipher, the shift cipher is one that anyone can readily understand and remember for decoding. It is a form of the substitution cipher. By shifting the alphabet a few positions in either direction, a simple sentence can become unreadable to casual inspection. Example 3.1 is an example of such a shift.[2]

Interestingly, for cryptogram word games, spaces are always included. Often puzzles use numbers instead of letters for the substitution. Removing the spaces in this particular example can make the ciphertext somewhat more secure. The possibility for multiple solutions becomes an issue; any number of words might fit the pattern.

Today many software tools are available to decode most cryptograms quickly and easily (at least, those not written in a dead language). You can have some fun with these tools; for example, the name Scott Ellis, when decrypted, turns into Still Books. The name of a friend of the author decrypts to "His Sinless." It is apparent, then, that smaller-sample simple substitution ciphers can have more than one solution.

Much has been written and stated about frequency analysis; it is considered the "end-all and be-all" with respect to cipher decryption. Frequency analysis is not to be confused with cipher breaking, which is a modern attack against the actual cryptographic algorithms themselves. However, to think simply plugging of in some numbers generated from a Google search is naïve. The frequency chart in Table 3.5 is commonplace on the Web.

It is beyond the scope of this chapter to delve into the accuracy of the table, but suffice it to say that our own analysis of Chapter 40's 118,000 characters, a technical text, yielded a much different result (Table 3.6). Perhaps the significantly larger sample and the fact that it is a technical text make the results different after the top two. In addition, where computers are concerned, an actual frequency analysis would take into consideration all ASCII characters, as shown in Table 3.6.

TABLE 3.5 "In a Random Sampling of 1000 Letters," This Pattern Emerges

Letter	Frequency
E	130
T	93
N	78
R	77
I	74
O	74
A	73
S	63
D	44
H	35
L	35
C	30
F	28
P	27
U	27
M	25
Y	19
G	16
W	16
V	13
B	9
X	5
K	3
Q	3
J	2
Z	1
Total	**1000**

EXAMPLE 3.1 A Sample Cryptogram. Try This Out: Gv Vw, Dtwvg?

Hint: Caesar said it, and it is in Latin.

Frequency analysis is not difficult; once of all the letters of a text are pulled into a database program, it is straightforward to count all the duplicate values. The snippet of code in Example 3.2 demonstrates one way in which text can be transformed into a single column and imported into a database.

The cryptograms that use formatting (every word becomes the same length) are considerably more difficult

2. Et tu, Brute?

TABLE 3.6 **Using MS Access to Perform Frequency Analysis of Chapter 40 in This Book**

Chapter 40 Letters	Frequency
e	14,467
t	10,945
a	9,239
i	8,385
o	7,962
s	7,681
n	7,342
r	6,872
h	4,882
l	4,646
d	4,104
c	4,066
u	2,941
m	2,929
f	2,759
p	2,402
y	2,155
g	1,902
w	1,881
b	1,622
v	1,391
.	1,334
,	1,110
k	698
0	490
x	490
q	166
7	160
*	149
5	147
)	147
(	146
j	145
3	142
6	140
Æ	134
ò	134
ô	129
ö	129
4	119
z	116
Total	**116,798**

Continued

Characters with fewer repetitions than z were excluded from the return. Character frequency analysis of different types of communications yields slightly different results.

EXAMPLE 3.2 **How Text Can Be Transformed Into a Single Column and Imported Into a Database**

```
Sub Letters2column ()
Dim bytText () As Byte
Dim bytNew() As Byte
Dim IngCount As Long
With ActiveDocument.Content
bytText = .Text
ReDim bytNew((((UBound(bytText()) + 1) * 2) - 5))
For IngCount = 0 To (UBound(bytText()) - 2) Step two
bytNew((lngCount * 2)) = bytText(lngCount)
bytNew(((lngCount * 2) + 2)) = 13
Next IngCount
.Text = bytNew()
End With
End Sub
```

for basic online decryption programs to crack. They must take into consideration spacing and word lengths when considering whether a string matches a word. It stands to reason, then, that the formulation of the cipher (in which a substitution that is based partially on frequency similarities and with a whole lot of obfuscation, so that when messages are decrypted, they have ambiguous or multiple meanings) would be desirable for simple ciphers. However, this would be true only for very short and very obscure messages that could be code words to decrypt other messages or could simply be sent to misdirect the opponent. The amount of ciphertext needed to break a cipher successfully is called the *unicity distance*. Ciphers with small unicity distances are weaker than those with large ones.

Ultimately, substitution ciphers are vulnerable to either word-pattern analysis, letter-frequency analysis, or some combination of both. Where numerical information is encrypted, tools such as Benford's law can be used to elicit patterns of numbers that *should* be occurring. Forensic techniques incorporate such tools to uncover accounting fraud. Thus, although this particular cipher is a child's game, it is useful in that it is an underlying principle of cryptography and should be well understood before continuing. The primary purpose of discussing it here is as an introduction to ciphers.

Further topics of interest and places to find information involving substitution ciphers are the chi-square statistic, Edgar Allan Poe, Sherlock Holmes, Benford's law, Google, and Wikipedia. For example, an Internet search for Edgar Allan Poe + cryptography will lead you to articles detailing how Poe's interest in the subject and his use of it in stories such as "The Gold-Bug" served to popularize and raise awareness of cryptography in the general public.

The Polyalphabetic Cipher

The preceding clearly demonstrated that although the substitution cipher is fun and easy, it is also vulnerable and weak. It is especially susceptible to frequency analysis. Given a large enough sample, a cipher can easily be broken by mapping the frequency of the letters in the ciphertext to the frequency of letters in the language or dialect of the ciphertext (if it is known). To make ciphers more difficult to crack, Blaise de Vigenère, from the 16th-century court of Henry III of France, proposed a polyalphabetic substitution. In this cipher, instead of a one-to-one relationship, there is a one-to-many. A single letter can have multiple substitutes. The Vigenère solution was the first known cipher to use a keyword.

It works like this: First, a *tableau* is developed, as in Table 3.7. This tableau is a series of shift ciphers. In fact, because there can be only 26 additive shift ciphers, it is all of them.

In Table 3.7, a table combined with a keyword is used to create the cipher. For example, if we choose the keyword *rockerrooks*, overlay it over the plaintext, and cross-index it to Table 3.7, we can produce the ciphertext. In this example, the top row is used to look up the plaintext and the leftmost column is used to reference the keyword.

For example, we lay the word *rockerrooks* over the sentence "Ask not what your country can do for you." Line 1 is the keyword, line 2 is the plaintext, and line 3 is the ciphertext:

> Keyword: ROC KER ROOK SROC KERROOK SRO CK ERR OOK
> Plaintext: ASK NOT WHAT YOUR COUNTRY CAN DO FOR YOU
> Ciphertext: RGM XSK NVOD QFIT MSLEHFI URB FY JFI MCE

The similarity of this tableau to a mathematical table like the one shown in Table 3.8 becomes apparent. Just think letters instead of numbers, and it becomes clear how this works. The top row is used to "look up" a letter from the plaintext, the leftmost column is used to locate the overlaying keyword letter, and where the column and the row intersect is the ciphertext.

In fact, this similarity is the weakness of the cipher. Through some creative "factoring," the length of the keyword can be determined. Because the tableau is, in practice, a series of shift ciphers, the length of the keyword determines how many ciphers are used. With only six distinct letters the keyword *rockerrook* uses only six ciphers. Regardless, for nearly 300 years many people believed the cipher to be unbreakable.

The Kasiski—Kerckhoff Method

Now let us look at Kerckhoff's principle: "Only secrecy of the key provides security." (This principle is not to be confused with Kirchhoff's law, a totally different man and rule.) In the 19th century, Auguste Kerckhoff stated that essentially, a system should still be secure, even when everyone knows everything about the system (except the password). Basically, his thought was that if more than one person knows something, it is no longer a secret. Throughout modern cryptography, the inner workings of cryptographic techniques have been well-known and published. Creating a portable, secure, unbreakable code is easy if nobody knows how it works. The problem lies in the fact that we people just cannot keep a secret!

In 1863, Friedrich Kasiski, a Prussian major, proposed a method to crack the Vigenère cipher.[3] Briefly, his method required the cryptographer to deduce the length of the keyword used and then dissect the cryptogram into a corresponding number of ciphers. This is accomplished simply by examining the distance between repeated strings in the ciphertext. Each cipher would then be solved independently. The method required a suitable number of bigrams to be located. A *bigram* is a portion of the ciphertext two characters long, which repeats in a discernible pattern. In Example 3.3, a repetition has been deliberately made simple with a short keyword (*toto*) and engineered by crafting a harmonic between the keyword and the plaintext.

This might seem an oversimplification, but it effectively demonstrates the weakness of the polyalphabetic cipher. Similarly, the polyalphanumeric ciphers, such as the Gronsfeld cipher, are even weaker because they use 26 letters and 10 digits. This one also happens to decrypt to "On of when on of," but a larger sample with such a weak keyword would easily be cracked by even the least intelligent Web-based cryptogram solvers. The harmonic is created by the overlaying keyword with the underlying text; when the bigrams "line up" and repeat themselves,

3. D. Kahn, The Codebreakers—The Story of Secret Writing, Scribner, New York, NY, 1996. (ISBN:0,684,831,309)

TABLE 3.7 Vigenère's Tableau Arranging All Shift Ciphers Into a Single Table

Letter	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

Vigenère's tableau then implements a keyword to create a more complex cipher than the simple substitution or shift ciphers. The number of spurious keys, or bogus decryptions, that result from attempting to decrypt a polyalphabetic encryption, is greater than those created during the decryption of a single shift cipher.

TABLE 3.8 Multiplication Table Is the Inspiration for the Vigenère Tableau

Multiplier	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

EXAMPLE 3.3 A Repetitious, Weak Keyword Combines With Plaintext to Produce an Easily Deciphered Ciphertext

I	to	to	toto	to	to	toto	to
Plaintext	it	is	what	it	is,	Isn't	it?
Ciphertext	BH	BG	PVTH	BH	BG	BGGH	BH

the highest frequency will be the length of the password. The distance between the two occurrences will be the length of the password. In Example 3.3, we see BH and BG repeating, and then we see BG repeating at a tight interval of 2, which tells us the password might be two characters long and based on two shift ciphers that, when decrypted side by side, will make a real word. Not all bigrams will be indicators of this, so some care must be taken. As can be seen, BH repeats with an interval of 8, but the password is not eight digits long (however, it is a factor of 8!). By locating the distance of all of the repeating bigrams and factoring them, we can deduce the length of the keyword.

4. MODERN CRYPTOGRAPHY

Some of cryptography's greatest stars emerged in World War II. For the first time during modern warfare, vast resources were devoted to enciphering and deciphering communications. Both sides made groundbreaking advances in cryptography. Understanding the need for massive calculations (for the time: more is probably happening in the random-access memory of this author's personal computer over a period of 5 min than happened in all of the war), both sides developed new machinery, predecessors to modern solid-state computers, that could be coordinated to perform the calculations and procedures needed to crack enemy ciphers.

The Vernam Cipher (Stream Cipher)

Gilbert Sandford Vernam (1890–1960) invented the stream cipher in 1917; a patent was issued on July 22, 1919. Vernam worked for Bell Labs, and his patent described a cipher in which a prepared key, on a paper tape, combined with plaintext to produce a transmitted ciphertext message. He did not use the term "*d'art*" "XOR," but he implemented the same logic at the relay layer. The credit for automating cryptography goes to Vernam, who introduced the Baudot system, which is the Morse code of the teletype, to cryptography. In it, each character is represented by five units, or pulses. With the expectation that a set number of "pulses" would be transmitted over a given period of time, the pulse, or absence of it, creates a system of zeros and ones that flesh out a binary system. Vernam was the first to suggest that a prepunched tape (cipher) could *combine* with the plaintext and yield difficult to crack ciphertext. The same tape would then be used to decrypt the ciphertext. Through testing and development, it became apparent that two tapes could be used and offset against one another to produce many different ciphers. Later, methods were derived to employ a single stream of random numbers to create an unbreakable cipher. Physical problems barred this from gaining wide implementation; the logistics of managing or transmitting the random cipher, and then knowing which message to which it applied, were simply insurmountable in wartime, when messaging increased dramatically. Regardless, Vernam's accomplishment of employing a method of automation to encryption cannot be underestimated. He developed a way in which, using a series of magnets and relays, the cipher and plaintext pulses could be combined electrically.[4] Fig. 3.2 shows a page from the actual patent papers, Patent No. 1,310,719.[5]

In effect, the Vernam stream cipher and "one-time pad" ciphers are similar; in fact, Vernam later coinvented it. The primary difference is that the "one-time pad" cipher dictates that a truly random stream cipher be used for the encryption. The stream cipher had no such requirement and used a different method of relay logic to combine a pseudorandom stream of bits with the plaintext bits. (The XOR process is discussed in more detail in the section on XOR ciphering.) In practice today, the Vernam cipher is any stream cipher in which pseudorandom or random text is combined with plaintext to produce cipher text that is the same length as the cipher. RC4 is a modern example of a Vernam cipher.

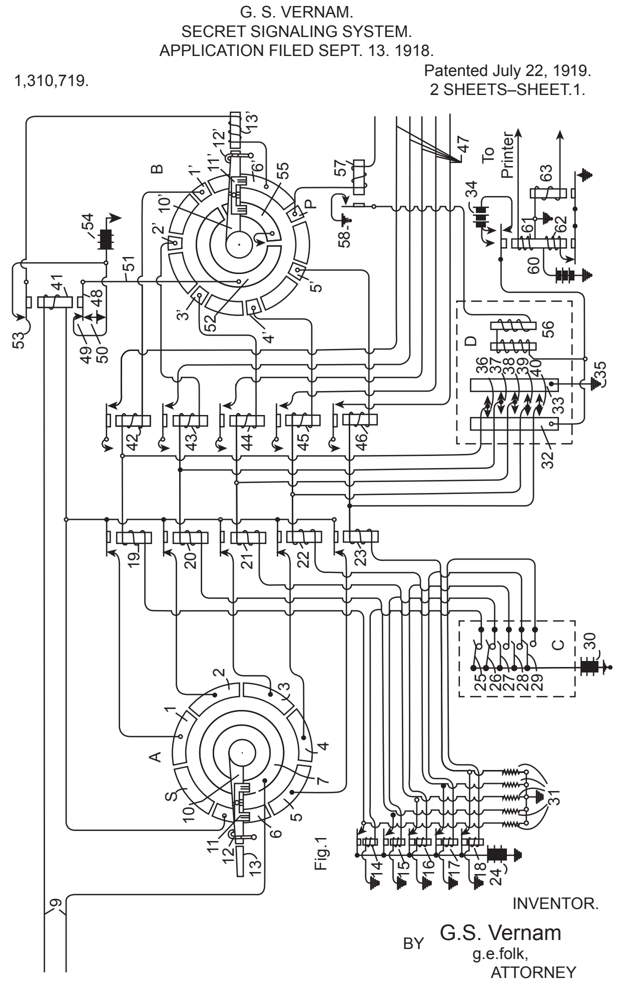


FIGURE 3.2 G. S. Vernam's Secret Signaling System introduced bit-by-bit enciphering using XOR technology to the world of cryptography for the first time.

4. D. Kahn, The Codebreakers—The Story of Secret Writing (394—403), (Scribner, 1996).
5. U.S. Patent 1,310,719.

The One-Time Pad

The "one-time pad" cipher, attributed to Joseph Mauborgne, is perhaps one of the most secure forms of cryptography. It is difficult to break if used properly, and if the key stream is perfectly random, the ciphertext gives away absolutely no details about the plaintext, which renders it unbreakable. As the name suggests, it uses a single random key that is the same length as the entire message, and it uses the key only once. The word "pad" is derived from the fact that the key is distributed on pads of paper, with each sheet torn off and destroyed as it is used.

There are several weaknesses to this cipher. We begin to see that the more secure the encryption is, the more it will rely on other means of key transmission. The more a key has to be moved around, the more likely it is that someone who should not have it will have it. The following weaknesses are apparent in this "bulletproof" style of cryptography:

- The key length has to equal plaintext length.
- It is susceptible to key interception; the key must be transmitted to the recipient, and the key is as long as the message.
- It is cumbersome, because it doubles the traffic on the line.
- The cipher must be perfectly random.
- One-time use is absolutely essential. As soon as two separate messages are available, the messages can be decrypted. Example 3.4 demonstrates this.

Since most people do not use binary, the author takes the liberty in Example 3.4 of using decimal numbers modulus 26 to represent the XOR that would take place in a bitstream encryption (see the section on the XOR cipher) that uses the method of the one-time pad.

A numeric value is assigned to each letter, as seen in Table 3.9. By assigning a numeric value to each letter, adding the plaintext value to the ciphertext value, modulus 26, yields a pseudo-XOR, or a wraparound Caesar shift that has a different shift for each letter in the entire message.

As this example demonstrates, by using the same cipher twice, a dimension is introduced that allows for the introduction of frequency analysis. By placing the two streams side by side, we can identify letters that are the same. In a large enough sample, in which the ciphertext is sufficiently randomized, frequency analysis of the aligned values will begin to crack the cipher wide open because we know that they are streaming in a logical order: the order in which they were written. One of the chief advantages of 21st-century cryptography is that the "eggs" are scrambled and descrambled during decryption based on the key, which in fact you do not want people to know. If the same cipher is used repeatedly, multiple inferences can be made, and eventually the entire key can be deconstructed. Because plaintext 1 and plaintext 2 are so similar, this sample yields

EXAMPLE 3.4 Using the Random Cipher, a Modulus Shift Instead of an XOR, and Plaintext to Produce Ciphertext

Plaintext 1

t h i s w i l l b e s o e a s y t o b r e a k i t w i l l b e f u n n y

20 8 9 19 23 9 12 12 2 5 19 15 5 1 19 25 20 15 2 18 5 1 11 9 20 23 9 12 12 2 5 6 21 14 14 25

Cipher 1

q e r t y u i o p a s d f g h j k l z x c v b n m q a z w s x e r f v t

17 5 18 20 25 21 9 15 16 1 19 4 6 7 8 10 11 12 26 24 3 22 2 14 13 17 1 26 23 19 24 5 18 6 22 20

Ciphertext 1

11 13 1 13 22 4 21 1 18 6 12 19 11 8 1 9 5 1 2 16 8 23 13 23 7 14 10 12 9 21 3 11 13 20 10 19

k m a m v d u a r f l s k h a i e a b p h w m w g n j l w u c k m t j s

Plaintext 2

T h i s w i l l n o t b e e a s y t o b r e a k o r b e t o o f u n n y

20 8 9 19 23 9 12 12 14 15 20 2 5 5 1 19 25 20 15 2 18 5 1 11 15 18 2 5 20 15 15 6 21 14 14 25

Ciphertext 2, also using Cipher 1

11 13 1 13 22 4 21 1 4 16 13 6 11 12 9 3 10 6 15 0 21 1 3 25 2 9 3 5 17 8 13 11 13 20 10 19

k m a m v d u a e p m f k l i f j f o z u a c y b i c e q h m k m t j s

Some Statistical Tests for Cryptographic Applications by Adrian Fleissig

In many applications, it is often important to determine whether a sequence is random. For example, a random sequence provides little or no information in cryptographic analysis. When estimating economic and financial models, it is important for the residuals from the estimated model to be random. Various statistical tests can be used to evaluate whether a sequence is actually random. For a truly random sequence, it is assumed that each element is generated independently of any prior and/or future elements. A statistical test is used to compute the probability that the observed sequence is random compared with a truly random sequence. The procedures have test statistics that are used to evaluate the null hypothesis, which typically assumes that the observed sequence is random. The alternative hypothesis is that the sequence is nonrandom. Thus, failing to accept the null hypothesis, at some critical level selected by the researcher, suggests that the sequence may be nonrandom.

There are many statistical tests to evaluate for randomness in a sequence, including frequency tests, runs tests, discrete Fourier transforms, and serial tests. The test statistics often have chi-square or standard normal distributions that are used to evaluate the hypothesis. Whereas no test is superior overall to the others, a frequency or runs test is a good starting point to examine for nonrandomness in a sequence. As an example, a frequency or runs test typically evaluates whether the number of zeros and ones in a sequence are about the same, as would be the case if the sequence were truly random.

It is important to examine the results carefully. For example, the researcher may incorrectly fail to accept the null hypothesis that the sequence is random, and therefore may make a type I error. Incorrectly accepting the null of randomness when the sequence is actually nonrandom results in committing a type II error. The reliability of the results depends on having a sufficiently large number of elements in a sequence. In addition, it is important to perform alternative tests to evaluate whether a sequence is random.

the following harmonics (in bold and boxed), as shown in Example 3.5.

Cracking Ciphers

One method of teasing out the frequency patterns is by applying some sort of mathematical formula to test a hypothesis against reality. The chi-square test is perhaps one of the most commonly used; it allows someone to use what is called *inferential statistics* to draw certain inferences about the data by testing them against known statistical distributions.

Using the chi-square test against an encrypted text would allow certain inferences to be made, but only where the contents, or the type of contents (random or of an expected distribution), of the text were known. For example, someone may use a program that encrypts files. By creating the null hypothesis that the text is completely random and by reversing the encryption steps, a block cipher may emerge as the null hypothesis is disproved through the chi-square test. This would be done by reversing the encryption method and XORing against the bytes with a block created from the known text. At the point where the nonencrypted text matches the positioning of the encrypted text, chi-square would reveal that the output is not random and the block cipher would be revealed.

$$\text{Chi-squared} = \ldots(\text{observed-expected})2/(\text{expected})$$

What would be observed would be the actual 0:1 ratio produced by XORing the data streams together, and what would be expected would be the randomness of zeros and ones (50:50) in a body of pseudorandom text.

Independent of having a portion of the text, a large body of encrypted text could be reverse-encrypted using a block size of all zeros. In this manner it may be possible to

TABLE 3.9 A Simple Key Is Created So That Random Characters and Regular Characters May Be Combined With a Modulus Function

Key																									
a	b	e	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
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

Without the original cipher, this key is meaningless intelligence. It is used here in a similar capacity as an XOR, which is also a function that everyone knows how to perform.

EXAMPLE 3.5 Where Plaintext 1 and Plaintext 2 Are so Similar, This Sample Yields the Following Harmonics (In Bold and Boxed)

Side by side, the two ciphertexts show a high level of harmonics. This indicates that two different ciphertexts actually have the same cipher. Where letters are different, because XOR is a known process and our encryption technique is also publicly known, it is a simple matter to say that *r* = 18, e = 5 (Table 3.9), and thus construct an algorithm that can tease apart the cipher and ciphertext to produce plaintext.

```
k m a m v d u a r f l s k h a i e a b p h w m w g n j l w u c k m t j s (ciphertext 1)
k m a m v d u a e p m f k l i f j f o z u a c y b i c e q h m k m t j s (ciphertext 2)
```

tease out a block cipher by searching for nonrandom block-sized strings. Modern encryption techniques generate many block cipher permutations that are layered against previous iterations ($n - 1$) of permutated blocks. The feasibility of running such decryption techniques would require a heavy-duty programmer and a statistician, an incredible amount of processing power, and in-depth knowledge of the encryption algorithm used. An unpublished algorithm would render such testing worthless.

The methods and procedures employed in breaking encryption algorithms are used throughout society in many applications where a null hypothesis needs to be tested. Forensic consultants use pattern matching and similar decryption techniques to combat fraud on a daily basis. Adrian Fleissig, a seasoned economist, uses many statistical tests to examine corporate data (see the sidebar, "Some Statistical Tests for Cryptographic Applications").[6]

The XOR Cipher and Logical Operands

In practice, the XOR cipher is not so much a cipher as it is a mechanism whereby ciphertext is produced. "Random binary stream cipher" would be a better term. The terms "XOR," "logical disjunction," and "inclusive" may be used interchangeably. Most people are familiar with the logical functions of speech, which are words such as "and," "or," "nor," and "not." A girl can tell her brother, "Mother is either upstairs or at the neighbor's," which means she could be in either state, but you have no way of knowing which one it is. The mother could be in either place, and you cannot infer from the statement the greater likelihood of either. The outcome is undecided.

Alternatively, if a salesman offers a customer either a blue car or a red car, the customer knows that he can have red or he can have blue. Both statements are true. Blue cars and red cars exist simultaneously in the world. A person can own both a blue car and a red car. But Mother will never be in more than one place at a time. Purportedly, there is a widespread belief that no author has produced an example of an English *or* sentence that appears to be false because both of its inputs are true.[7] Quantum physics takes considerable exception to this statement (which explains quantum physicists) at the quantum-mechanical level. In the Schrödinger cat experiment, the sentence "The cat is alive or dead" or the statement "The photon is a particle and a wave until you look at it, then it is a particle or a wave, depending on how you observed it" both create a quandary for logical operations, and there are

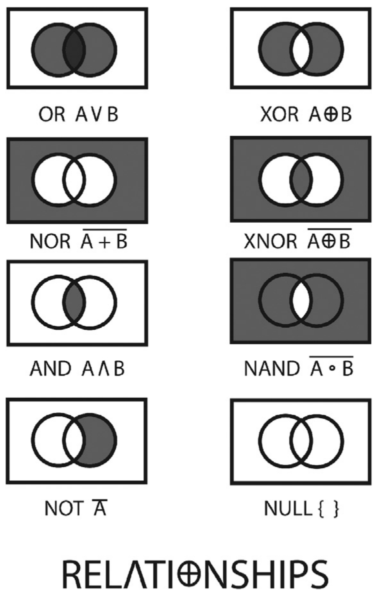


FIGURE 3.3 In each Venn diagram, the possible outcome of two inputs is decided.

no Venn diagrams or words that depend on time or quantum properties of the physical universe. Regardless of this exception, when speaking of things in the world in a more rigorously descriptive fashion (in the macroscopically nonphenomenological sense), greater accuracy is needed.

To create a greater sense of accuracy in discussions of logic, the operands as listed in Fig. 3.3 were created. When attempting to understand this chart, the best thing to do is to assign a word to the A and B values and think of each Venn diagram as a universe of documents, perhaps in a document database or just on a computer being searched. If A stands for the word "tree" and B for "frog," each letter simply takes on a significant and distinct meaning.

In computing, it is traditional that a value of 0 is false and a value of 1 is true. Thus, an XOR operation is the determination of whether two possibilities can be combined to produce a value of true or false, based on whether both operations are true, both are false, or one of the values is true.

$$1 \text{ XOR } 1 = 0$$
$$0 \text{ XOR } 0 = 0$$
$$1 \text{ XOR } 0 = 1$$
$$0 \text{ XOR } 1 = 1$$

In an XOR operation, if the two inputs are different, the resultant is TRUE, or 1. If the two inputs are the same, the resultant value is FALSE, or 0.

In Example 3.6, the first string represents the plaintext and the second line represents the cipher. The third line represents the ciphertext. If, and only exactly if, just one of

6. Adrian Fleissig is the Senior Economist of Counsel for RGL Forensics, 2006–present. He is also a full professor, California State University Fullerton (CSUF) with a joint Ph.D. in Economics and Statistics from North Carolina State University in 1993.

7. Barrett, Stenner, The myth of the exclusive 'or,' Mind 80 (317) (1971) 116–121. [First names or initials needed for authors].

EXAMPLE 3.6 Lines 1 and 2 Are Combined With an XOR Operand to Produce Line 3

Line 1, plaintext: 1 0 0 1 1 1 0 1 0 1 1 0 1 1 1 1
Line 2, random cipher "": 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 1
Line 3, XOR ciphertext: 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0

the items has a value of TRUE, the results of the XOR operation will be true.

Without the cipher, and if the cipher is truly random, decoding the string becomes impossible. However, as in the one-time pad, if the same cipher is used, then (1) the cryptography becomes vulnerable to a known text attack, and (2) it becomes vulnerable to statistical analysis. Example 3.7 demonstrates this by showing exactly where the statistical aberration can be culled in the stream. If we know they both used the same cipher, can anyone solve for Plaintext A and Plaintext B?

EXAMPLE 3.7 Where the Statistical Aberration Can Be Culled in the Stream

To reconstruct the cipher if the plaintext is known, PlaintextA can be XOR'd to ciphertextB to produce cipherA! Clearly, in a situation where plaintext may be captured, using the same cipher key twice could completely expose the message. By using statistical analysis, the unique possibilities for PlaintextA and PlaintextB will emerge; *unique possibilities* means that for ciphertext = *x*, where the cipher is truly random, this should be at about 50% of the sample. Additions of ciphertext *n* + 1 will increase the possibilities for unique combinations because, after all, these binary streams must be converted to text and the set of binary stream possibilities that will combine into ASCII characters is relatively small. Using basic programming skills, you can develop algorithms that will sort through these data quickly and easily to produce a deciphered result. An intelligent person with some time on her hands could sort it out on paper or on an Excel spreadsheet. When the choice is "The red house down the street from the green house is where we will meet" or a bunch of garbage, it begins to become apparent how to decode the cipher.

CipherA and PlaintextA are XOR'd to produce ciphertextA:

PlaintextA: 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
cipherA: 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
ciphertextA: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

PlaintextB and cipherA are XOR'd to produce ciphertextB:

ciphertextB: 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
cipherA: 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
PlaintextB: 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
|<—— Column 1 ——>||<——Column 2 —— |
Note: Compare ciphertextA to ciphertextB!

Block Ciphers

Block ciphers work in a way similar to polyalphabetic ciphers, with the exception that a block cipher pairs together two algorithms for the creation of ciphertext and its decryption. It is also somewhat similar in that, whereas the polyalphabetic cipher uses a repeating key, the block cipher uses a permutating yet repeating cipher block. Each algorithm uses two inputs: a key and a "block" of bits, each of a set size. Each output block is the same size as the input block, the block being transformed by the key. The key, which is algorithm based, is able to select the permutation of its bijective mapping from 2*n*, where *n* is equal to the number of bits in the *input* block. Often when 128-bit encryption is discussed, it is referring to the size of the *input* block. Typical encryption methods involve use of XOR chaining or some similar operation (Fig. 3.4).

Block ciphers have been widely used since 1976 in many encryption standards. As such, for a long time cracking these ciphers became the top priority of cipher crackers everywhere. Block ciphers provide the backbone algorithmic technology behind most modern-era ciphers.

5. THE COMPUTER AGE

Many people consider January 1, 1970, to be the dawn of the computer age. That is when Palo Alto Research Center (PARC) in California introduced modern computing; the graphical user interface (no more command line and punch cards), networking on an Ethernet, and object-oriented programming have all been attributed to PARC. The 1970s also featured the UNIX clock, Alan Shepard on the moon, the US Bicentennial, the civil rights movement, women's liberation, Robert Heinlein's sci-fi classic, *Stranger in a Strange Land*, the birth of my wife, and, most important to this chapter, modern cryptography. The late 1960s and early 1970s changed the face of the modern world at breakneck speed. Modern warfare reached tentative heights with radio-guided missiles, and warfare needed a new hero. And then there was the Data Encryption Standard (DES); in a sense, DES was the turning point for cryptography, in that for the first time it fully leveraged the power of modern computing in its algorithms. The sky appeared to be the limit, but, unfortunately for those who wanted to keep their information secure, decryption techniques were not far behind.

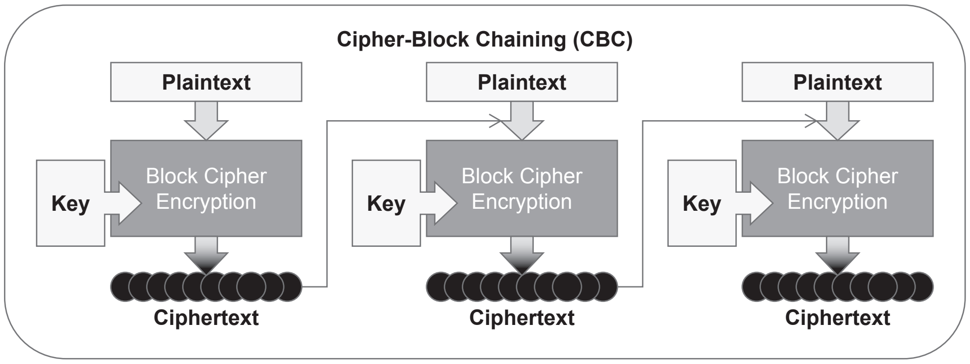


FIGURE 3.4 XOR chaining, or cipher-block chaining, is a method in which the next block of plaintext to be encrypted is XOR'd with the previous block of ciphertext before being encrypted.

Data Encryption Standard

In the mid-1970s, the US Government issued a public specification, through its National Bureau of Standards (NBS), called the DES. This could perhaps be considered the dawn of modern cryptography because it was likely the first block cipher, or at least its first widespread implementation. However, the 1970s were a relatively untrusting time. "Big Brother" loomed right around the corner (as per George Orwell's *1984*), and most people did not understand or necessarily trust DES. Issued under the NBS, now called the National Institute of Standards and Technology (NIST), hand in hand with the National Security Agency (NSA), DES led to tremendous interest in the reliability of the standard among academia's ivory towers. A shortened key length and the implementation of substitution boxes, or "S-boxes," in the algorithm led many to think that the NSA had deliberately weakened the algorithms and left a security "back door" of sorts.

The use of S-boxes in the standard was not generally understood until the design was published in 1994 by Don Coppersmith. The S-boxes, it turned out, had been deliberately designed to prevent a sort of cryptanalysis attack called *differential cryptanalysis*, as was discovered by IBM researchers in the early 1970s; the NSA had asked IBM to keep quiet about it. In 1990 the method was "re"-discovered independently, and when used against DES, the usefulness of the S-boxes became readily apparent.

Theory of Operation

DES used a 64-bit block cipher combined with a mode of operation based on cipher-block chaining (CBC) called the *Feistel function*. This consisted of an initial expansion permutation followed by 16 rounds of XOR key mixing via subkeys and a key schedule, substitution (S-boxes), and permutation.[8] In this strategy, a block is increased from 32 to 48 bits (expansion permutation). Then the 48-bit block is divided in half. The first half is XORs, with parts of the key according to a key schedule. These are called subkeys. Fig. 3.5 shows this concept in a simplified format.

The resulting cipher is then XOR'd with the half of the cipher that was not used in step 1. The two halves switch sides. Substitution boxes reduce the 48 bits down to 32 bits via a nonlinear function, and then a permutation, according to a permutation table, takes place. Then the entire process is repeated 16 times, except in the last step the two halves are not flipped. Finally, this diffusive strategy produced via substitution, permutation, and key schedules creates an effective ciphertext. Because a fixed-length cipher, a block cipher, is used, the permutations and the S-box introduce enough confusion that the cipher cannot be deduced through brute force methods without extensive computing power.

With the increase in size of hard drives and computer memory, the need for disk space and bandwidth still demands that a block-cipher algorithm be portable. DES, Triple DES, and the AES all provide or have provided solutions that are secure and practical.

Implementation

Despite the controversy at the time, DES was implemented. It became the encryption standard of choice until the late 1990s, when it was broken, when Deep Crack and distributed.net broke a DES key in 22 h 15 min. Later that year, a new form of DES called Triple DES, which encrypted the plaintext in three iterations, was published. It remained in effect until 2002, when it was superseded by AES.

Rivest, Shamir, and Adleman

The release of DES included the creation and release of Ron Rivest, Adi Shamir, and Leonard Adleman's encryption algorithm [Rivest, Shamir, and Adleman (RSA)]. Rivest, Shamir, and Adleman, based at the Massachusetts Institute of Technology, publicly described the algorithm in

8. A. Sorkin, Lucifer: a cryptographic algorithm, Cryptologia 8 (1) (1984) 22–35.

Feistel Structure of DES (Simplified)

1. Plaintext	0	0	0	0	0	0	0	0
2. Expanded plaintext A)*	0	0	0	0	0	0	>	>
3. Cipher	1	0	1	0	0	1		
4. Ciphertext A	1	0	1	0	0	1		

Expanded Plaintext B:

>>	0	0	1	1	1	1

Key Schedule
1. 101000
2. 100101

.

.

.

15. 010100
16. 101101

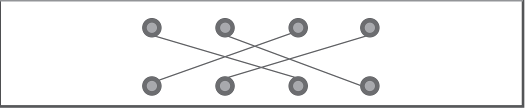
5. Expanded Plaintext B)	0	0	1	1	1	1
6. Cipher from Step 1:	1	0	1	0	0	0
7. Ciphertext B	1	0	0	1	1	1

8. Cipher (B,A)	1	0	0	1	1	1	1	0	1	0	0	0
9. Substitution (S–box)			0	0	0	1			1	0	0	1
10. Permutation P–Box			0	1	0	0			0	1	1	0

Key (16 parts) : 101000 010101 101011 . . . 010100 100101 101010 101101

S – Box**	0011	0100
10	0101	1001
11	0001	1100

Permutation Box (P–Box)



*A bijective function is applied to expand from 32 bits (represented here by 8 bits) to 48 bits. A function is bijective if inverse relation f^{-1} (x) is also a function. Reduction is achieved through the S–box operation.

** This S–box is reduced to include only the four bits that are in our cipher. Typical S–boxes are as large as needed and may be based partially on the key.

FIGURE 3.5 The Feistel function with a smaller key size. *DES*, Data Encryption Standard.

1977. RSA is the first encryption standard to introduce (to public knowledge) the new concept of digital signing. In 1997 it was revealed through declassification of papers that Clifford Cocks, a British mathematician working for the UK Government Communications Headquarters, had written a paper in 1973 describing this process. Assigned top secret status, the work had never previously seen the light of day. Because it was submitted in 1973, the method had been considered unattainable, because computing power at the time could not handle its methods.

Advanced Encryption Standard (or Rijndael)

AES represents one of the latest chapters in the history of cryptography. Once it became clear that neither DES nor its answer to its weaknesses, "Triple-DES," could carry encryption through to the 21st century, a decree went out from the NIST so that a new standard might be achieved. AES won out over the other standards for several reasons, and it is currently one of the most popular encryption standards. For people involved in any security work, its occurrence on the desktop is frequent. It also enjoys the free marketing and acceptance that it received when it was awarded the title of official cryptography standard in 2001.[9] This designation went into effect in May of the following year.

To a large degree, this part of the chapter is merely a rehashing/book report on the Federal Information Processing Standards (FIPS) 197 standard, because this appears to be one of the more authoritative guides available, short of the authors themselves. It provides a few original examples and some observations made by the author during his examination of the standard.

Similar to DES, AES encrypts plaintext in a series of rounds, involves the use of a key and block sizes, and leverages substitution and permutation boxes. It differs from DES in the following respects:

- It supports 128-bit block sizes.
- The key schedule is based on the S-box.

9. U.S. FIPS PUB 197 (FIPS 197), November 26, 2001.

- It expands the key, not the plaintext.
- It is not based on a Feistel cipher.
- It is extremely complex.

The AES algorithms are to symmetric ciphers what a bowl of spaghetti is to the shortest distance between two points. Through a series of networked XOR operations, key substitutions, temporary variable transformations, increments, iterations, expansions, value swapping, S-boxing, and the like, a strong encryption is created that, with modern computing, creates a cipher that in itself is impossible to break. Like all ciphers, though, AES is only as strong as its weakest link, which is the password routine. This weakness will be explored toward the end of this part of the chapter.

Overview

Simply put, it works like this: First, the idea is to confuse the real message and the encrypted message. Like other encryption methods, it uses the XOR to do this. AES requires 128, 192, or 256 bits to work; however, one must have a "key" with which to start. This might be a password or a string of random numbers stored on a card, or any input derived from an unchanging but unique thing, such as your retina. From there, the "key" needs to be both obfuscated and expanded to match the correct block size, and to be parceled up into the little packages, or blocks, that will be used in later operations of the encryption sequence. To accomplish this, a procedure called Password-Based Key Derivation Function (PBKDF2) is used.[10] Enciphering is then achieved by using an XOR and hashing the bits together repeatedly through a shift row offset. This effectively "shuffles the deck."

Next, introduce diffusion using a simple column transposition to rearrange the bits until they are no longer sensible as letters, and then hashing the bits using substitution (XOR). Furthermore, AES employs a key expansion and multiple rounds. For example, if you XOR a string and produce ciphertext, you have successfully obfuscated the message. If you want anyone to read it, just give them the key and they can reverse the XOR to get the correct text. The problem arises in portability: We want people to be able to decrypt our messages, but only if they have the pass key. This is where things get tricky, because we have to have a key that is long (128 or 256 bits), but that can be generated from a password that is reasonably short, so that we can remember it. Ultimately, this is the weakness of AES or actually the gateway into it: that is, the weak link.

10. RSA Laboratories Public-Key Cryptography Standards (PKCS) #5: Password-Based Cryptography Specification, Version 2.0. Network Working Group, B. Kaliski.

The FIPS 197 standard canonicalizes the Rijndael algorithm (Fig. 3.6) [3,4], which is a symmetric block cipher with the capability of processing blocks of 128 bits. To do this, it employs a multistep operation, enciphering the plaintext blocks using cipher keys with lengths of 128, 192, and 256 bits. The Rijndael algorithm possesses the ability to handle additional block sizes and key lengths, so although the capability exists, it is not part of the published FIPS 197 standard. This part of the chapter will cover the following topics:

1. some definitions that have not yet been covered
2. a brief discussion of the notation and conventions used
3. some mathematical properties of the algorithm
4. a brief discussion of how a password becomes a 128 (or longer) cipher key
5. a summary of the algorithm specification, including the key expansion, encryption, and decryption routines
6. implementation issues
7. a step-by-step example from FIPS 197

The Basics of Advanced Encryption Standard

The basic unit of encryption is the byte. If you read the beginning of this part of the chapter, you already know that the cipher key must be in the form of 128, 192, or 256 bits, which is 16, 24, or 32 bytes, respectively. All bit values are 0 or 1; NULL values are disallowed. This of course may spark the question, "How, then, do NULL values get encrypted in BIT columns in a database?" In Microsoft SQL Server, an additional "hidden" column called a NULLmap exists; if a value is NULL, it will be set to 1 otherwise, 0.[11]

AES encryption operates at the byte level, with each 4 bits represented (for convenience here) hexidecimally so that the following is true:

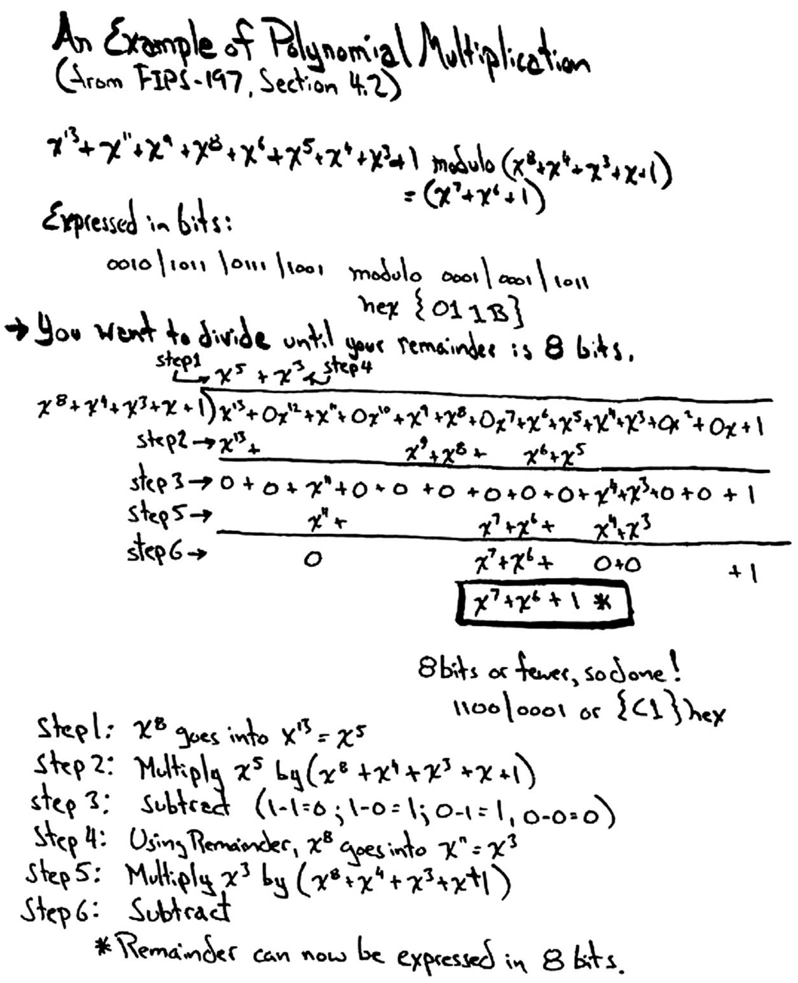
Binary value.hexidecimal value

For example, the value 1100 1101 would be represented as/xCD. XOR'd, with 0111 0110/x76, would result in 1011 1011, or/xBB. (Note how obfuscating it is that two completely different pairs can XOR to the same value.)

6. HOW ADVANCED ENCRYPTION STANDARD WORKS

The following describes each step of the cipher. It is a simplification, intended to provide a solid foundation for future study of the standard.

11. P.S. Randall, Misconceptions around Null Bitmap Size, 2012. http://www.sqlskills.com/BLOGS/PAUL/post/Misconceptions-around-null-bitmap-size.aspx.



An Example of Polynomial Multiplication
(from FIPS-197, Section 4.2)

$x^{13}+x^{11}+x^{9}+x^{8}+x^{6}+x^{5}+x^{4}+x^{3}+1$ modulo $(x^{8}+x^{4}+x^{3}+x+1)$
$= (x^{7}+x^{6}+1)$

Expressed in bits:

0010|1011 |0111 |1001 modulo 0001|0001|1011
hex {01 1B}

→ You want to divide until your remainder is 8 bits.

step1 → $x^{5}+x^{3}$ ← step4

$x^{8}+x^{4}+x^{3}+x+1\,)\,x^{13}+0x^{12}+x^{11}+0x^{10}+x^{9}+x^{8}+0x^{7}+x^{6}+x^{5}+x^{4}+x^{3}+0x^{2}+0x+1$

step2 → $x^{13}+ \quad x^{9}+x^{8}+ \quad x^{6}+x^{5}$

step3 → $0+0+x^{11}+0+0+0+0+0+0+x^{4}+x^{3}+0+0+1$

step5 → $x^{11}+ \quad x^{7}+x^{6}+ \quad x^{4}+x^{3}$

step6 → $0 \quad x^{7}+x^{6}+ \quad 0+0 \quad +1$

$\boxed{x^{7}+x^{6}+1 \ast}$

8bits or fewer, so done!
1100|0001 or {C1} hex

Step1: x^{8} goes into $x^{13} = x^{5}$
Step2: Multiply x^{5} by $(x^{8}+x^{4}+x^{3}+x+1)$
step3: Subtract (1-1=0; 1-0=1; 0-1=1, 0-0=0)
Step4: Using Remainder, x^{8} goes into $x^{11} = x^{3}$
Step5: Multiply x^{3} by $(x^{8}+x^{4}+x^{3}+x+1)$
Step6: Subtract
*Remainder can now be expressed in 8 bits.

FIGURE 3.6 Handwritten example of polynomial expansion using the Rijndael/Advanced Encryption Standard encryption algorithm.

Bytes

Programmatically, to encipher the plaintext bits, the AES procedure requires all of the bits to be loaded and arranged into a two-dimensional array called the State. The State has four rows in it, and each row contains 4 bytes. This is a total of 16 bytes, or 128 bits.

Note: You might ask, "128 bits is great, but how did we go from a 32-bit password typed by a user to a 128-bit cipher key?" This can be done in a number of ways, and an industrious engineer may certainly write his own method for it, but for the readers of this book, check out the PBKDF2. This is a key derivation function that is part of RSA Laboratories Public Key Cryptography Standards. It replaces an earlier standard, PBKDF1, which could only produce derived keys up to 160 bits long.

The bytes are arranged in columns, so that the first column, first row (let us call it A1) has, right "beneath" it, A2, which would b the second byte of the string to be encrypted. The actual FIPS standard has more dramatic notations for this, but essentially what is happening is that in the State, bytes are referred to by row and by column, and they are loaded top to bottom, left to right. Each column of 4 bytes in the State is referred to as Word. Then it starts to do some math.

Math

The AES standard employs the mathematical operations of both addition and multiplication. Addition using the XOR has already been covered heavily in this chapter. For examples see Table 3.6. This standard also relies heavily on prime, or irreducible, polynomials to allow for enciphering of the bits and to keep things nice and tidy in 128-bit buckets. It is important that for reversibility, all of the multiplication operations, where strings of bits are represented as polynomials which can then be manipulated, allowing for the shifting of bits, remain irreducible.

For example, multiply together the primes 3, 7, and 17, and the resulting number is easily calculated as 357. By factoring it, you can easily derive the exact three numbers used in the original equation. Now multiply together 2, 8, and 16, and you get 256. Unfortunately, if you try to invert the operation, with the requirement that you want only three factors, you can arrive at 4, 4, and 16. Perhaps this is fine if you are writing a data-scrambling application, but an encryption utility is only as good as its ability to invert the cipher and decrypt the string. The AES standard outlines the mathematical polynomials used in the multiplication operations, and it defines them as being irreducible. For example, the purpose of one of the polynomial expressions in AES is simply to rotate the word, so that [b0, b1, b2, b3] is transformed into [b1, b2, b3, b0]. An irreducible polynomial is used so that no matter what the input produces, the inverse operation performed against the output ciphertext yields the correct input. The cipher key becomes the *solution* for an extremely long equation, a solution that has such great uniqueness that it cannot be guessed easily or quickly.

Fig. 3.5 provides an example of the actual mathematics behind the first expansion of the FIPS 197 standard. In it, each power of x takes a bit position, numbered as follows: 7654 3210. So, x^7 turns on the bit in position 7, x^6 in position 6, and x^0 (i.e., 1) takes the zeroth position. Hence, $x^7 + x^6 + 1 = 1100\ 0001$. This is why the remainder has to have x to the power of 7 or less, so it can be expressed in 8 bits. According to the standard, "these bytes are interpreted as finite field elements using a polynomial representation." Frequently, to conserve space or just to make things look less ridiculously *binary*, a hexadecimal value may be used. Table 3.10 demonstrates the conversion of binary to base 16, aka "hexidecimal."

TABLE 3.10 Binary and Its Hexadecimal Equivalents

Binary	Hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	a
1011	b
1100	c
1101	d
1110	e
1111	f

In the Beginning

In the beginning, there are bits, and the bits have a physicality that is linear. That is, they are all lined up in one continuous string on the disk, unless the disk is fragmented, but that is another story. You should always keep your disks defragmented. Fragmentation is bad and will affect the performance of processing data for encryption. If, for example, you are encrypting thousands of files in a particular folder on the disk, and the files are all over the place, it will perform poorly. I digress. When a program that executes AES encryption gets its byte on your bits, the first thing it does is load them into a series of arrays called the State. This particular state is good because it will not take all of your money or tell you that you did not pay enough taxes last year. What it will do is provide a place where many different operations can be executed to encrypt your data better using a 128-bit cipher key. For the purpose of convenience, although AES can handle 192- and 256-bit encryption as well, the author simply refers to the 128-bit model. All operations are performed against this two-dimensional array of bytes called the State, which contains four rows of bytes; each row holds Nb bytes, where Nb is the block length (128, 192, 256) divided by 32.

The State array, s, has two indices. Denoted by the symbol s, each individual byte has two indices, with its row number r in the range $0 \leq r < 4$ and its column number c in the range $0 \leq c < Nb$. This allows an individual byte of the State to be referred to as either sr,c or s[r,c]. AES requires $Nb = 4$, so that $0 \leq c < 4$. In other words, if you think of an input, a state, and an output array as being the program product line, each array will be the same size. AES explodes the size of the output file.

Rounds

The number of rounds to be executed by the algorithm depends on the key size. $Nr = 10$ when $Nk = 4$, $Nr = 12$ when $Nk = 6$, and $Nr = 14$ when $Nk = 8$. AES, for encipherment, uses a "round" methodology, where each round consists of four steps:

1. byte substitution driven by a substitution table (S-box)
2. the shifting of rows in the State array by an offset
3. bit and byte shuffling within each column of the State
4. adding a round key to the State

These transformations (and their inverses) are explained in detail in Sections 5.1.1–5.1.4 and 5.3.1–5.3.4 of FIPS

197. Details and code samples can be found in the standard. This example is drawn directly from the standard and details the operations of the cipher itself. Essentially, the following functions are described in the standard and can be understood to be the steps taken in each encryption round. The number of rounds depends on the size of the encryption key:

1. SubBytes(state)
2. ShiftRows(state)
3. MixColumns(state)
4. AddRoundKey(state, w[round*Nb, (round + 1)*Nb-1])

By now, the reader of this text should realize that public standards such as FIPS 197 contain a wealth of information and that the chapters in this book can merely provide (it is hoped) the background needed to lend clarity to the material. This part of the chapter is, of course, no substitution for actually reading and adhering to the standard as published.

Finally, it is conceivable that with so complex a series of operations, a computer file and block could be combined in such a way as to produce all zeros. Theoretically, the AES cipher could be broken by solving massive quadratic equations that take into consideration every possible vector and solve 8000 quadratic equations with 1600 binary unknowns. This sort of an attack is called an *algebraic attack*, and, where traditional methods such as differential or differential cryptanalysis fail, it is suggested that the strength in AES lies in the current inability to solve supermultivariate quadratic equations with any sort of efficiency.

Reports that AES is not as strong as it should be are currently likely to be overstated and inaccurate, because anyone can present a paper that is dense and difficult to understand and claims to achieve the incredible. It is unlikely that at any time in the near or maybe not so near future (this author hedges his bets), AES will be broken using multivariate quadratic polynomials in thousands of dimensions. Mathematica is very likely one of the most powerful tools that can solve quadratic equations, and it is still many years away from being able to perform this feat.

Ultimately, AES's biggest drawback is that a user can trigger an encryption using a password of his or her choice. Unfortunately, most people choose passwords that are not strong; they want something they will *remember*. There are many IT techs who lost their passwords and rendered systems inalterable. There have also been many sinister communications that may pass from an employee to a future employer, or a competitor with whom he has become friendly, and has decided to pass secrets. Intellectual property tort is a real facet of litigation, and to this end, large consultancies that deal in computer forensics and e-discovery host rack upon rack of devices that are designed specifically to decrypt files that have been encrypted using AES encryption. They do this not by attacking the algorithm, but by attacking using brute force. *Hash tables*, or rainbow tables, are basically a list of all the known hash values of common (and not so common) passwords. For example, one might take every known phone number in the United States and create a table of all of their known hashes. One might also create one of every known child's name and parse that into the hash tables. For example, a phone number of 847-555-5555 might be combined with the name "Ethan" (who is known to live at a certain address, 233 TreeView), into a password of 233Ethan5555tree! (I added the exclamation point to be even more "secure"…). In fact, some of the largest consultancy firms that manage large litigations have constructed exactly this sort of database of rainbow tables, and they are generating more and more hashes each day. Programs that provide entire disk encryption are the bane of both law enforcement and litigation.

Brute force attacks are the only way to crack in when no key can (or will) be produced. Why do rainbow tables work? The spectrum of possible passwords that people may choose to use *because they can remember* them is much smaller than the total number of possible passwords that exist. By leveraging as much as 7 terabytes of rainbow tables against an encrypted body, the estimated success rate of cracking files, speculatively, could be as high as 60% to 70%. This is a horrible statistic for an encryption algorithm that is supposedly "unbreakable." So perhaps the one take-away from this writing is that there is still room for improvement; a truly unbreakable system still does not exist, and although the algorithm of AES cannot be successfully attacked through decomposition of the ciphertext, any system that fails to take into account *every* attack vector ultimately will be no stronger than its weakest link.

Finally, let us briefly look at the process used to select cryptographic mechanisms. This is similar to the process used to select any IT mechanism.

7. SELECTING CRYPTOGRAPHY: THE PROCESS

The cryptography selection process is documented in the system development life cycle (SDLC) model. An organization can use many SDLC models to develop an information system effectively. A traditional SDLC is a linear sequential model. This model assumes that the system will be delivered near the end of its development life cycle. Another SDLC model employs prototyping, which is often used to develop an understanding of system requirements without developing a final operational system. More complex models have been developed to address the evolving complexity of advanced and large information system designs. The SDLC model is embedded in any of the major system developmental approaches:

- waterfall: The phases are executed sequentially.
- spiral: The phases are executed sequentially, with feedback loops to previous phases.
- incremental development: Several partial deliverables are constructed, and each deliverable has incrementally more functionality. Builds are constructed in parallel, using available information from previous builds. The product is designed, implemented, integrated, and tested as a series of incremental builds.
- evolutionary: There is replanning at each phase in the life cycle, based on feedback. Each phase is divided into multiple project cycles with deliverable measurable results at the completion of each cycle.

An Agenda for Action for Selecting the Cryptographic Process Activities

The following high-level checklist questions should be addressed in determining the appropriate cryptographic mechanisms, policies, and procedures for a system (check all tasks completed):

_____**1.** How critical is the system to the organization's mission, and what is the impact level?

_____**2.** What are the performance requirements for cryptographic mechanisms (communications throughput and processing latency)?

_____**3.** What intersystem and intrasystem compatibility and interoperability requirements need to be met by the system (algorithm, key establishment, and cryptographic and communications protocols)?

_____**4.** What are the security/cryptographic objectives required by the system (content integrity protection, source authentication required, confidentiality, and availability)?

_____**5.** For what period of time will the information need to be protected?

_____**6.** What regulations and policies are applicable in determining what is to be protected?

_____**7.** Who selects the protection mechanisms that are to be implemented in the system?

_____**8.** Are the users knowledgeable about cryptography, and how much training will they receive?

_____**9.** What is the nature of the physical and procedural infrastructure for the protection of cryptographic material and information (storage, accounting and audit, and logistics support)?

_____**10.** What is the nature of the physical and procedural infrastructure for the protection of cryptographic material and information at the facilities of outside organizations with which cryptographically protected communications are required (facilities and procedures for protection of physical keying material)?

Security should be incorporated into all phases, from initiation to disposition, of an SDLC model. The goal of the selection process is to specify and implement cryptographic methods that address specific agency/organization needs.

Before selecting a cryptographic method, an organization should consider the operational environment, application requirements, types of services that can be provided by each type of cryptography, and cryptographic objectives that must be met when selecting applicable products. Based on the requirements, several cryptographic methods may be required. For example, both symmetric and asymmetric cryptography may be needed in one system, each performing different functions (symmetric encryption, and asymmetric digital signature and key establishment). In addition, high-level checklist questions should be addressed in determining the appropriate cryptographic mechanisms, policies, and procedures for a system (see checklist: An Agenda for Action for Selecting the Cryptographic Process Activities).

8. SUMMARY

Today's IT security environment consists of highly interactive and powerful computing devices and interconnected systems of systems across global networks in which organizations routinely interact with industry, private citizens, state and local governments, and the governments of other nations. Consequently, both private and public sectors depend on information systems to perform essential and mission-critical functions. In this environment of increasingly open and interconnected systems and networks, network and data security are essential for the optimum use of this IT. For example, systems that carry out electronic financial transactions and electronic commerce must protect against unauthorized access to confidential records and the unauthorized modification of data.

Thus, in keeping with the preceding, this chapter provided guidance to organizations regarding how to select cryptographic controls for protecting sensitive information. However, to provide additional information, products of other standards organizations (the American National Standards Institute and International Organization for Standardization) were briefly discussed.

This chapter was also intended for security individuals responsible for designing systems and for procuring, installing, and operating security products to meet identified security requirements. This chapter may be used by:

- a manager responsible for evaluating an existing system and determining whether cryptographic methods are necessary;
- program managers responsible for selecting and integrating cryptographic mechanisms into a system;
- a technical specialist requested to select one or more cryptographic methods/techniques to meet a specified requirement;

- a procurement specialist developing a solicitation for a system or network that will require cryptographic methods to perform security functionality.

In other words, this chapter provided those individuals with sufficient information that allowed them to make informed decisions about the cryptographic methods that met their specific needs to protect the confidentiality, authentication, and integrity of data that are transmitted and/or stored in a system or network. In addition, this primer also provided information about selecting cryptographic controls and implementing the controls in new or existing systems.

Finally, let us move on to the real interactive part of this chapter: review questions/exercises, hands-on projects, case projects, and the optional team case project. The answers and/or solutions by chapter can be found in the Online Instructor's Solutions Manual.

CHAPTER REVIEW QUESTIONS/ EXERCISES

True/False

1. True or False? For most information technology occupations, knowledge of cryptography is a large part of a broader skill set and is generally limited to relevant applications.
2. True or False? Cryptography is built on one overarching premise: the need for a cipher that can be used reliably and portably to encrypt text so that through any means of cryptanalysis (differential, deductive, algebraic, or the like) the ciphertext can be undone with any available technology.
3. True or False? In effect, the Vernam stream cipher and "one-time pad" ciphers are different; in fact, Vernam later coinvented it.
4. True or False? DES used a 64-bit block cipher combined with a mode of operation based on cipher-block chaining (CBC) called the *Feistel function.*
5. True or False? The cryptography selection process is documented in the system development life cycle (SDLC) model.

Multiple Choice

1. In essence, computer-based cryptography is the art of creating a form of communication that embraces the following precepts, except which two?
   A. Can be readily misunderstood by the intended recipients
   B. Cannot be understood by the unintended recipients
   C. Can be understood by the unintended recipients
   D. Can be readily understood by the intended recipients
   E. Can be adapted and changed easily with relatively small modifications, such as a changed pass phrase or word
2. What is known as the method of encryption?
   A. Plaintext
   B. Clear text
   C. Ciphertext
   D. Cryptogram
   E. Cipher
3. Decryption methods often rely on understanding the context of the:
   A. Cipher
   B. Ciphertext
   C. Shift cipher
   D. Cryptogram
   E. Cryptographic algorithms
4. The amount of ciphertext needed to break a cipher successfully is known as:
   A. Benford's law
   B. Chi-square statistic
   C. Polyalphabetic cipher
   D. Kerckhoff's principle
   E. Unicity distance
5. One method of teasing out the frequency patterns is through the application of some sort of mathematical formula to test a hypothesis against reality. What test is perhaps one of the most commonly used?
   A. Inferential statistics test
   B. Chi-square test
   C. Statistical test
   D. Random binary stream cipher test
   E. Block cipher test

EXERCISE

Problem

OpenSSL has a trick in it that mixes uninitialized memory with the randomness generated by the operating system's formal generator. The standard idea here is that it is good practice to mix different sources of randomness into your own source. Modern operating systems take several random things such as disk drive activity and net activity and mix the measurements into one pool, and then run it through a fast hash to filter it. Cryptoplumbing, on the other hand, of necessity involves lots of errors and fixes and patches. Bug-reporting channels are important, and apparently this was used. A security team found the bug with an analysis tool. It was duly reported up to OpenSSL, but the handover was muffed. The reason it was muffed was that it was not obvious what was going on. The reason it was not obvious is that code was too clever for its own good. It tripped up the analysis tool and the programmers, and the fix did not alert the OpenSSL programmers.

Complexity is always the enemy in security code. So, with the preceding in mind, as a risk manager, what would you do to fix the problem?

Hands-On Projects

Project

What is the basic method for using the output of a random bit generator?

Case Projects

Problem

How would an organization go about generating key pairs for asymmetric key algorithms?

Optional Team Case Project

Problem

How would an organization go about generating keys for symmetric key algorithms?