Information Security: Principles and Practice, Second Edition
by Mark Stamp

Part I

Crypto

Information Security: Principles and Practice, Second Edition
by Mark Stamp

Chapter 2

Crypto Basics

MXDXBVTZWVMXNSPBQXLIMSCCSGXSCJXBOVQXCJZMOJZCVC
TVWJCZAAXZBCSSCJXBQCJZCOJZCNSPOXBXSBTVWJC
JZDXGXXMOZQMSCSCJXBOVQXCJZMOJZCNSPJZHGXXMOSPLH
JZDXZAAXZBXHCSCJXTCSGXSCJXBOVQX
— plaintext from Lewis Carroll, *Alice in Wonderland*

The solution is by no means so difficult as you might be led to imagine from the first hasty inspection of the characters. These characters, as any one might readily guess, form a cipher—that is to say, they convey a meaning...
— Edgar Allan Poe, *The Gold Bug*

2.1 Introduction

In this chapter we'll discuss some of the basic elements of cryptography. This discussion will lay the foundation for the remaining crypto chapters which, in turn, underpin much of the material throughout the book. We'll avoid mathematical rigor as much as possible. Nevertheless, there is enough detail here so that you will not only understand the "what" but you will also have some appreciation for the "why."

After this introductory chapter, the remaining crypto chapters focus on:

- Symmetric key cryptography
- Public key cryptography
- Hash functions
- Advanced cryptanalysis

A handful of special topics are also covered.

2.2 How to Speak Crypto

The basic terminology of crypto includes the following.

- *Cryptology* — the art and science of making and breaking "secret codes."
- *Cryptography* — the making of "secret codes."
- *Cryptanalysis* — the breaking of "secret codes."
- *Crypto* — a synonym for any or all of the above (and more), where the precise meaning should be clear from context.

A *cipher* or *cryptosystem* is used to *encrypt* data. The original unencrypted data is known as *plaintext*, and the result of encryption is *ciphertext*. We *decrypt* the ciphertext to recover the original plaintext. A *key* is used to configure a cryptosystem for encryption and decryption.

In a *symmetric* cipher, the same key is used to encrypt and to decrypt, as illustrated by the black box cryptosystem in Figure 2.1.[1] There is also a concept of *public key* cryptography where the encryption and decryption keys are different. Since different keys are used, it's possible to make the encryption key public—thus the name public key.[2] In public key crypto, the encryption key is, appropriately, known as the *public key*, whereas the decryption key, which must remain secret, is the *private key*. In symmetric key crypto, the key is known as a *symmetric key*. We'll avoid the ambiguous term secret key.

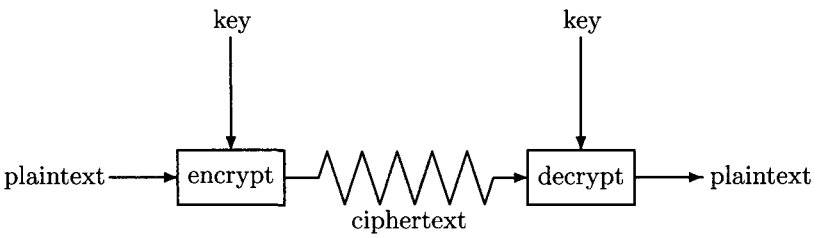

Figure 2.1: Crypto as a Black Box

For an ideal cipher, it is infeasible to recover the plaintext from the ciphertext without the key. That is, even if the attacker, Trudy, has complete knowledge of the algorithms used and lots of other information (to be made more precise later), she can't recover the plaintext without the key. That's the goal, although reality sometimes differs.

[1] This is the only black box you'll find in this book!

[2] Public key crypto is also known as asymmetric crypto, in reference to the fact that the encryption and decryption keys are different.

A fundamental tenet of cryptography is that the inner workings of a cryptosystem are completely known to the attacker, Trudy, and the only secret is a key. This is known as *Kerckhoffs' Principle*, which, believe it or not, was named after a guy named Kerckhoffs. In the year 1883, Kerckhoffs, a Dutch linguist and cryptographer, laid out six principles of cipher design and use [164]. The principle that now bears his name states that a cipher "must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience" [165], that is, the design of the cipher is not secret.

What is the point of Kerckhoffs' Principle? After all, it must certainly be more difficult for Trudy to attack a cryptosystem if she doesn't know how the cipher works. So, why would we want to make Trudy's life easier? There are (at least) a couple of problems with relying on a secret design for your security. For one, the details of "secret" cryptosystems seldom, if ever, remain secret for long. Reverse engineering can be used to recover algorithms from software, and even algorithms embedded in tamper-resistant hardware are sometimes subject to reverse engineering attacks and exposure. And, even more worrisome is the fact that secret crypto-algorithms have a long history of failing to be secure once the algorithms have been exposed to public scrutiny—see [29] for a relatively recent example where Microsoft violated Kerckhoffs' Principle.

Cryptographers will not deem a crypto-algorithm worthy of use until it has withstood extensive public analysis by many cryptographers over an extended period of time. The bottom line is that any cryptosystem that does not satisfy Kerckhoffs' Principle is suspect. In other words, ciphers are presumed guilty until "proven" innocent.

Kerckhoffs' Principle is often extended to cover various aspects of security well beyond cryptography. In other contexts, this basic principle is usually taken to mean that the security design itself is open to public scrutiny. The belief is that "more eyeballs" are more likely to expose more security flaws and therefore ultimately result in a system that is more secure. Although Kerckhoffs' Principle (in both its narrow crypto form and in a broader context) seems to be universally accepted in principle, there are many real-world temptations to violate this fundamental tenet, almost invariably with disastrous consequences. Throughout this book we'll see several examples of security failures that were directly caused by a failure to heed the venerable Mr. Kerckhoffs.

In the next section, we look briefly at a few classic cryptosystems. Although the history of crypto is a fascinating topic [159], the purpose of this material is to provide an elementary introduction to some of the crucial concepts that arise in modern cryptography. In other words, pay attention since we will see all of these concepts again in the next couple of chapters and in many cases, in later chapters as well.

2.3 Classic Crypto

In this section, we examine four classic ciphers, each of which illustrates a feature that is relevant to modern cryptosystems. First on our agenda is the simple substitution, which is one of the oldest cipher systems—dating back at least 2,000 years—and one that is good for illustrating some basic attacks. We then turn our attention to a type of double transposition cipher, which includes important concepts that are used in modern ciphers. We also discuss classic codebooks, since many modern ciphers can be viewed as the "electronic" equivalent of codebooks. Finally, we consider the so-called one-time pad, a practical cipher that is provably secure. No other cipher in this book (or in common use) is provably secure.

2.3.1 Simple Substitution Cipher

First, we consider a particularly simple implementation of a simple substitution cipher. In the simplest case, the message is encrypted by substituting the letter of the alphabet n places ahead of the current letter. For example, with $n = 3$, the substitution—which acts as the key—is

```
plaintext:  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
ciphertext: 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
```

where we've followed the convention that the plaintext is lowercase, and the ciphertext is uppercase. In this example, the key could be given succinctly as "3" since the amount of the shift is, in effect, the key.

Using the key 3, we can encrypt the plaintext message

```
fourscoreandsevenyearsago                    (2.1)
```

by looking up each plaintext letter in the table above and then substituting the corresponding letter in the ciphertext row, or by simply replacing each letter by the letter that is three positions ahead of it in the alphabet. For the particular plaintext in (2.1), the resulting ciphertext is

```
IRXUVFRUHDAGVHYHABHDUVDIR.
```

To decrypt this simple substitution, we look up the ciphertext letter in the ciphertext row and replace it with the corresponding letter in the plaintext row, or we can shift each ciphertext letter backward by three. The simple substitution with a shift of three is known as the Caesar's cipher.[3]

[3]Historians generally agree that the Caesar's cipher was named after the Roman dictator, not the salad.

There is nothing magical about a shift by three—any shift will do. If we limit the simple substitution to shifts of the alphabet, then the possible keys are $n \in \{0, 1, 2, ..., 25\}$. Suppose Trudy intercepts the ciphertext message

CSYEVIXIVQMREXIH

and she suspect that it was encrypted with a simple substitution cipher using a shift by n. Then she can try each of the 26 possible keys, "decrypting" the message with each putative key and checking whether the resulting putative plaintext makes sense. If the message really was encrypted via a shift by n, Trudy can expect to find the true plaintext—and thereby recover the key—after 13 tries, on average.

This brute force attack is something that Trudy can always attempt. Provided that Trudy has enough time and resources, she will eventually stumble across the correct key and break the message. This most elementary of all crypto attacks is known as an *exhaustive key search.* Since this attack is always an option, it's necessary (although far from sufficient) that the number of possible keys be too large for Trudy to simply try them all in any reasonable amount of time.

How large of a keyspace is large enough? Suppose Trudy has a fast computer (or group of computers) that's able to test 2^{40} keys each second.[4] Then a keyspace of size 2^{56} can be exhausted in 2^{16} seconds, or about 18 hours, whereas a keyspace of size 2^{64} would take more than half a year for an exhaustive key search, and a keyspace of size 2^{128} would require more than nine quintillion years. For modern symmetric ciphers, the key is typically 128 bits or more, giving a keyspace of size 2^{128} or more.

Now, back to the simple substitution cipher. If we only allow shifts of the alphabet, then the number of possible keys is far too small, since Trudy can do an exhaustive key search very quickly. Is there any way that we can increase the number of keys? In fact, there is no need not to limit the simple substitution to a shifting by n, since any permutation of the 26 letters will serve as a key. For example, the following permutation, which is not a shift of the alphabet, gives us a key for a simple substitution cipher:

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

In general, a simple substitution cipher can employ any permutation of the alphabet as a key, which implies that there are $26! \approx 2^{88}$ possible keys. With

[4] In 1998 the Electronic Frontier Foundation (EFF) built a special-purpose key cracking machine for attacking the Data Encryption Standard (DES, which we'll discuss in the next chapter). This machine, which cost $220,000, consisted of about 43,200 processors, each of which ran at 40 MHz and, overall, it was capable of testing about 2.5 million keys per second [156]. Extrapolating this to a state-of-the-art PC with a single 4 GHz processor, Trudy could test fewer than 2^{30} keys per second on one such machine. So, if she had access to 1000 such machines, she could test about 2^{40} keys per second.

Trudy's superfast computer that tests 2^{40} keys per second, trying all possible keys for the simple substitution would take more than 8900 millennia. Of course, she would expect to find the correct key half that time, or just 4450 millennia. Since 2^{88} keys is far more than Trudy can try in any reasonable amount of time, this cipher passes the crucial first requirement of any practical cipher, namely, the keyspace is big enough so that an exhaustive key search is infeasible. Does this mean that a simple substitution cipher is secure? The answer is a resounding no, as the attack described in the next section clearly illustrates.

2.3.2 Cryptanalysis of a Simple Substitution

Suppose Trudy intercepts the following ciphertext, which she suspects was produced by a simple substitution cipher, where the key could be any permutation of the alphabet:

```
PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOXBTFXQWA
XBVCXQWAXFQJVWLEQNTOZQGGQLFXQWAKVWLXQWAEBIPBFXFQVXGTVJV
WLBTPQWAEBFPBFHCVLXBQUFEVWLXGDPEQVPQGVPPBFTIXPFHXZHVFAG
FOTHFEFBQUFTDHZBQPOTHXTYFTODXQHFTDPTOGHFQPBQWAQJJTODXQH
FOQPWTBDHHIXQVAPBFZQHCFWPFHPBFIPBQWKFABVYYDZBOTHPBQPQJT
QOTOGHFQAPBFEQJHDXXQVAVXEBQPEFZBVFOJIWFFACFCCFHQWAUVWFL
QHGFXVAFXQHFUFHILTTAVWAFFAWTEVOITDHFHFQAITIXPFHXAFQHEFZ
QWGFLVWPTOFFA
```
(2.2)

Since it's too much work for Trudy to try all 2^{88} possible keys, can she be more clever? Assuming the plaintext is English, Trudy can make use of the English letter frequency counts in Figure 2.2 together with the frequency counts for the ciphertext in (2.2), which appear in Figure 2.3.

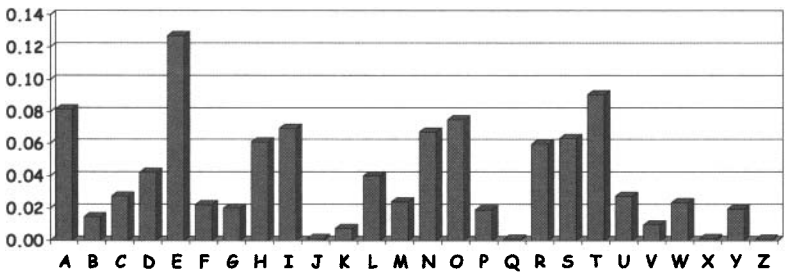

Figure 2.2: English Letter Frequency Counts

From the ciphertext frequency counts in Figure 2.3, we see that "F" is the most common letter in the encrypted message and, according to Figure 2.2, "E" is the most common letter in the English language. Trudy therefore

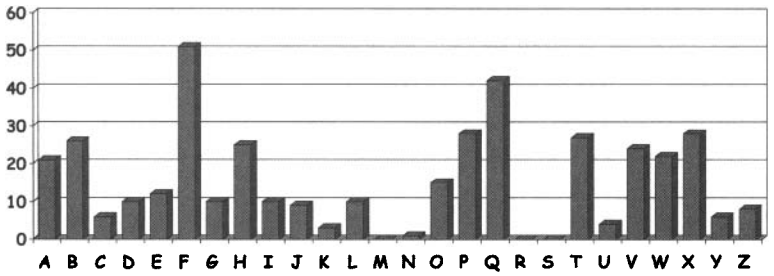

Figure 2.3: Ciphertext Frequency Counts

surmises that it's likely that "F" has been substituted for "E." Continuing in this manner, Trudy can try likely substitutions until she recognizes words, at which point she can be confident in her guesses.

Initially, the easiest word to determine might be the first word, since Trudy doesn't know where inter-word spaces belong in the text. Since the third plaintext letter appears to be "e," and given the high frequency counts of the first two letter, Trudy might reasonably guess (correctly, as it turns out) that the first word of the plaintext is "the." Making these substitutions into the remaining ciphertext, she will be able to guess more letters and the puzzle will begin to unravel. Trudy will likely make some missteps along the way, but with sensible use of the statistical information available, she will find the plaintext in considerably less time than 4450 millennia.

This attack on the simple substitution shows that a large keyspace is not sufficient to ensure security. This attack also shows that cipher designers must guard against clever attacks. But how can we protect against all such attacks, since new attacks are developed all the time? The answer is that we can't and, as a result, a cipher must be subjected to extensive analysis by skilled cryptographers before we can trust it—the more skilled cryptographers who have tried to break a cipher and failed, the more confidence we have in the system.

2.3.3 Definition of Secure

There are several reasonable definitions of a secure cipher. Ideally, we would like to have a rigorous mathematical proof that there is no feasible attack on a system, but such ciphers are few and far between and provably secure ciphers are impractical for most uses.

Lacking a proof that a cipher is secure, we could require that the best-known attack on the system is impractical, in the sense of being computationally infeasible. While this would seem to be the most crucial property, we'll use a slightly different definition. We say that a cryptosystem is *secure*

if the best-known attack requires as much work as an exhaustive key search. In other words, no shortcut attack is known.

Note that by our definition, a secure cipher with a small number of keys could be easier to break than an insecure one with a large number of keys. While this may seem counterintuitive, there is a method to the madness. The rationale for our definition is that a cipher can never offer more security than an exhaustive key search, so the key size could be considered its "advertised" level of security. If a shortcut attack is known, the algorithm fails to provide its advertised level of security, as indicated by the key length. In short, a shortcut attack indicates that the cipher has a design flaw.

Note also that in practice, we must select a cipher that is secure (in the sense of our definition) and has a large enough key space so that an exhaustive key search is impractical. Both factors are necessary when choosing a cipher to protect sensitive data.

2.3.4 Double Transposition Cipher

In this section we discuss another classic cipher that illustrates some important basic concepts. The double transposition presented in this section is a weaker form of the usual double transposition cipher. We use this form of the cipher since it provides a slightly simpler means of illustrating all of the points that we want to make.

To encrypt with a double transposition cipher, we first write the plaintext into an array of a given size and then permute the rows and columns according to specified permutations. For example, suppose we write the plaintext `attackatdawn` into a 3×4 array:

$$\begin{bmatrix} \mathtt{a} & \mathtt{t} & \mathtt{t} & \mathtt{a} \\ \mathtt{c} & \mathtt{k} & \mathtt{a} & \mathtt{t} \\ \mathtt{d} & \mathtt{a} & \mathtt{w} & \mathtt{n} \end{bmatrix}.$$

Now if we transpose (or permute) the rows according to $(1,2,3) \rightarrow (3,2,1)$ and then transpose the columns according to $(1,2,3,4) \rightarrow (4,2,1,3)$, we obtain

$$\begin{bmatrix} \mathtt{a} & \mathtt{t} & \mathtt{t} & \mathtt{a} \\ \mathtt{c} & \mathtt{k} & \mathtt{a} & \mathtt{t} \\ \mathtt{d} & \mathtt{a} & \mathtt{w} & \mathtt{n} \end{bmatrix} \longrightarrow \begin{bmatrix} \mathtt{d} & \mathtt{a} & \mathtt{w} & \mathtt{n} \\ \mathtt{c} & \mathtt{k} & \mathtt{a} & \mathtt{t} \\ \mathtt{a} & \mathtt{t} & \mathtt{t} & \mathtt{a} \end{bmatrix} \longrightarrow \begin{bmatrix} \mathtt{n} & \mathtt{a} & \mathtt{d} & \mathtt{w} \\ \mathtt{t} & \mathtt{k} & \mathtt{c} & \mathtt{a} \\ \mathtt{a} & \mathtt{t} & \mathtt{a} & \mathtt{t} \end{bmatrix}.$$

The ciphertext is then read from the final array:

$$\mathtt{NADWTKCAATAT} \tag{2.3}$$

For the double transposition, the key consists of the size of the matrix and the row and column permutations. Anyone who knows the key can simply put

the ciphertext into the appropriate sized matrix and undo the permutations to recover the plaintext For example, to decrypt (2.3), the ciphertext is first put into a 3×4 array. Then the columns are numbered as $(4, 2, 1, 3)$ and rearranged to $(1, 2, 3, 4)$, and the rows are numbered $(3, 2, 1)$ and rearranged into $(1, 2, 3)$,

$$\begin{bmatrix} \mathtt{N} & \mathtt{A} & \mathtt{D} & \mathtt{W} \\ \mathtt{T} & \mathtt{K} & \mathtt{C} & \mathtt{A} \\ \mathtt{A} & \mathtt{T} & \mathtt{A} & \mathtt{T} \end{bmatrix} \longrightarrow \begin{bmatrix} \mathtt{D} & \mathtt{A} & \mathtt{W} & \mathtt{N} \\ \mathtt{C} & \mathtt{K} & \mathtt{A} & \mathtt{T} \\ \mathtt{A} & \mathtt{T} & \mathtt{T} & \mathtt{A} \end{bmatrix} \longrightarrow \begin{bmatrix} \mathtt{A} & \mathtt{T} & \mathtt{T} & \mathtt{A} \\ \mathtt{C} & \mathtt{K} & \mathtt{A} & \mathtt{T} \\ \mathtt{D} & \mathtt{A} & \mathtt{W} & \mathtt{N} \end{bmatrix}$$

and we see that we have recovered the plaintext, namely, `attackatdawn`.

The bad news is that, unlike a simple substitution, the double transposition does nothing to disguise the letters that appear in the message. The good news is that the double transposition appears to thwart an attack that relies on the statistical information contained in the plaintext, since the plaintext statistics are disbursed throughout the ciphertext.

Even this simplified version of the double transposition is not a trivial cipher to break. The idea of smearing plaintext information through the ciphertext is so useful that it is employed by modern block ciphers, as we will see in the next chapter.

2.3.5 One-Time Pad

The one-time pad, which is also known as the Vernam cipher, is a provably secure cryptosystem. Historically it has been used in various times and places, but it's not practical for most situations. However, it does nicely illustrate some important concepts that we'll see again later.

For simplicity, let's consider an alphabet with only eight letters. Our alphabet and the corresponding binary representation of letters appear in Table 2.1. It's important to note that the mapping between letters and bits is not secret. This mapping serves a similar purpose as, say, the ASCII code, which is not much of a secret either.

Table 2.1: Abbreviated Alphabet

letter	`e`	`h`	`i`	`k`	`l`	`r`	`s`	`t`
binary	000	001	010	011	100	101	110	111

Suppose that Alice, who recently got a job as a spy, wants to use a one-time pad to encrypt the plaintext message

`heilhitler`.

She first consults Table 2.1 to convert the plaintext letters to the bit string

001 000 010 100 001 010 111 100 000 101.

The one-time pad key consists of a randomly selected string of bits that is the same length as the message. The key is then XORed with the plaintext to yield the ciphertext. For the mathematically inclined, a fancier way to say this is that we add the plaintext and key bits modulo 2.

We denote the XOR of bit x with bit y as $x \oplus y$. Since $x \oplus y \oplus y = x$, decryption is accomplished by XOR-ing the same key with the ciphertext. Modern symmetric ciphers utilize this magical property of the XOR in various ways, as we'll see in the next chapter.

Now suppose that Alice has the key

111 101 110 101 111 100 000 101 110 000

which is of the proper length to encrypt her message above. Then to encrypt, Alice computes the ciphertext as

	h	e	i	l	h	i	t	l	e	r
plaintext:	001	000	010	100	001	010	111	100	000	101
key:	111	101	110	101	111	100	000	101	110	000
ciphertext:	110	101	100	001	110	110	111	001	110	101
	s	r	l	h	s	s	t	h	s	r

Converting these ciphertext bits back into letters, the ciphertext message to be transmitted is `srlhssthsr`.

When her fellow spy, Bob, receives Alice's message, he decrypts it using the same shared key and thereby recovers the plaintext:

	s	r	l	h	s	s	t	h	s	r
ciphertext:	110	101	100	001	110	110	111	001	110	101
key:	111	101	110	101	111	100	000	101	110	000
plaintext:	001	000	010	100	001	010	111	100	000	101
	h	e	i	l	h	i	t	l	e	r

Let's consider a couple of scenarios. First, suppose that Alice has an enemy, Charlie, within her spy organization. Charlie claims that the actual key used to encrypt Alice's message is

101 111 000 101 111 100 000 101 110 000.

Bob decrypts the ciphertext using the key given to him by Charlie and obtains

	s	r	l	h	s	s	t	h	s	r
ciphertext:	110	101	100	001	110	110	111	001	110	101
"key":	101	111	000	101	111	100	000	101	110	000
"plaintext":	011	010	100	100	001	010	111	100	000	101
	k	i	l	l	h	i	t	l	e	r

Bob, who doesn't really understand crypto, orders that Alice be brought in for questioning.

Now let's consider a different scenario. Suppose that Alice is captured by her enemies, who have also intercepted the ciphertext. The captors are eager to read the message and Alice is "encouraged" to provide the key for this super-secret message. Alice claims that she is actually a double agent and to prove it she provides the "key"

111 101 000 011 101 110 001 011 101 101.

When Alice's captors "decrypt" the ciphertext using this "key," they find

	s	r	l	h	s	s	t	h	s	r
ciphertext:	110	101	100	001	110	110	111	001	110	101
"key":	111	101	000	011	101	110	001	011	101	101
"plaintext":	001	000	100	010	011	000	110	010	011	000
	h	e	l	i	k	e	s	i	k	e

Alice's captors, who are not very knowledgeable about crypto, congratulate Alice for her patriotism and release her.

While not a proof, these examples do indicate why the one-time pad is provably secure. The bottom line is that if the key is chosen at random, and used only once, then an attacker who sees the ciphertext has no information about the message itself (other than its length, which could be padded). That is, given the ciphertext, any "plaintext" of the same length can be generated by a suitable choice of "key," and all possible plaintexts are equally likely. So the ciphertext provides no meaningful information at all about the plaintext. From a cryptographer's point of view, it doesn't get any better than that.

Of course, we are assuming that the one-time pad cipher is used correctly. The key (or pad) must be chosen at random, used only once, and must be known only to the Alice and Bob.

Since we can't do better than provable security, why don't we always use the one-time pad? Unfortunately, the cipher is impractical for most applications. Why is this the case? The crucial problem is that the pad is the same length as the message and since the pad is the key, it must be securely shared with the intended recipient before the ciphertext can be decrypted. If we can securely transmit the pad, why not simply transmit the plaintext by the same means and do away with the encryption?

Below, we'll see an historical example where it actually did make sense to use a one-time pad—in spite of its limitations. However, for modern high data-rate systems, a one-time pad cipher would be totally impractical.

While we're at it, why is it that the one-time pad can only be used once? Suppose we have two plaintext messages P_1 and P_2 and we encrypted these as as $C_1 = P_1 \oplus K$ and $C_2 = P_2 \oplus K$, that is, we have two messages encrypted

with the same "one-time" pad K. In the cryptanalysis business, this is known as a *depth*. With one-time pad ciphertexts in depth, we see that

$$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$$

and the key has disappeared from the problem. In this case, the ciphertext does yield some information about the underlying plaintext. Another way to see this is consider an exhaustive key search. If the pad is only used once, then the attacker has no way to know whether the guessed key is correct or not. But if two messages are in depth, for the correct key, both putative plaintexts must make sense. This provides the attacker with a means to distinguish the correct key from incorrect guesses. The problem only gets worse (or better, from Trudy's perspective) the more times the key is reused.

Let's consider an example of one-time pad encryptions that are in depth. Using the same bit encoding as in Table 2.1, suppose we have

$$P_1 = \texttt{like} = 100\,010\,011\,000 \quad \text{and} \quad P_2 = \texttt{kite} = 011\,010\,111\,000$$

and both are encrypted with the same key $K = 110\ 011\ 101\ 111$. Then

	l	i	k	e
P_1:	100	010	011	000
K:	110	011	101	111
C_1:	010	001	110	111
	i	h	s	t

and

	k	i	t	e
P_2:	011	010	111	000
K:	110	011	101	111
C_2:	101	001	010	111
	r	h	i	t

If Trudy the cryptanalyst knows that the messages are in depth, she immediately sees that the second and fourth letters of P_1 and P_2 are the same, since the corresponding ciphertext letters are identical. But far more devastating is the fact that Trudy can now guess a putative message P_1 and check her results using P_2. Suppose that Trudy (who only has C_1 and C_2) suspects that $P_1 = \texttt{kill} = 011\,010\,100\,100$. Then she can find the corresponding putative key:

	k	i	l	l
putative P_1:	011	010	100	100
C_1:	010	001	110	111
putative K:	001	011	010	011

and she can then use this K to "decrypt" C_2 and obtain

C_2:	101	001	010	111
putative K:	001	011	010	011
putative P_2:	100	010	000	100
	l	i	e	l

Since this K does not yield a sensible decryption for P_2, Trudy can safely assume that her guess for P_1 was incorrect. When Trudy eventually guesses $P_1 =$ `like` she will obtain the correct key K and decrypt to find $P_2 =$ `kite`, thereby confirming the correctness of the key and, therefore, the correctness of both decryptions.

2.3.6 Project VENONA

The so-called VENONA project [315] provides an interesting example of a real-world use of the one-time pad. In the 1930s and 1940s, spies from the Soviet Union who entered the United States brought with them one-time pad keys. When it was time to report back to their handlers in Moscow, these spies used their one-time pads to encrypt their messages, which could then be safely sent back to Moscow. These spies were extremely successful, and their messages dealt with the most sensitive U.S. government secrets of the time. In particular, the development of the first atomic bomb was a focus of much of the espionage. The Rosenbergs, Alger Hiss, and many other well-known traitors—and many who were never identified—figure prominently in VENONA messages.

The Soviet spies were well trained and never reused the key, yet many of the intercepted ciphertext messages were eventually decrypted by American cryptanalysts. How can that be, given that the one-time pad is provably secure? In fact, there was a flaw in the method used to generate the pads, so that, in effect, long stretches of the keys were repeated. As a result, many messages were in depth, which enabled the successful cryptanalysis of much VENONA traffic.

Part of one interesting VENONA decrypt is given in Table 2.2. This message refers to David Greenglass and his wife Ruth. LIBERAL is Julius Rosenberg who, along with his wife Ethyl, was eventually executed for his role in nuclear espionage.[5] The Soviet codename for the atomic bomb was, appropriately, ENORMOUS. For any World War II-era history buff, the VENONA decrypts at [315] make for fascinating reading.

[5]David Greenglass served ten years of a fifteen year sentence for his part in the crime. He later claimed that he lied in crucial testimony about Ethyl Rosenberg's level of involvement—testimony that was probably decisive in her being sentenced to death.

Table 2.2: VENONA Decrypt of Message of September 21, 1944

```
[C% Ruth] learned that her husband [v] was called up by the army
but he was not sent to the front.  He is a mechanical engineer
and is now working at the ENORMOUS [ENORMOZ] [vi] plant in
SANTA FE, New Mexico.

[45 groups unrecoverable]

detain VOLOK [vii] who is working in a plant on ENORMOUS. He is a
FELLOWCOUNTRYMAN [ZEMLYaK] [viii].  Yesterday he learned that
they had dismissed him from his work.  His active work in
progressive organizations in the past was cause of his dismissal.

In the FELLOWCOUNTRYMAN line LIBERAL is in touch with CHESTER [ix].
They meet once a month for the payment of dues.  CHESTER is
interested in whether we are satisfied with the collaboration and
whether there are not any misunderstandings.  He does not inquire
about specific items of work [KONKRETNAYa RABOTA].  In as much
as CHESTER knows about the role of LIBERAL's group we beg consent
to ask C. through LIBERAL about leads from among people who are
working on ENOURMOUS and in other technical fields.
```

2.3.7 Codebook Cipher

A classic codebook cipher is, literally, a dictionary-like book containing (plaintext) words and their corresponding (ciphertext) codewords. To encrypt a given word, the cipher clerk would simply look up the word in the codebook and replace it with the corresponding codeword. Decryption, using the inverse codebook, was equally straightforward. Table 2.3 contains an excerpt from a famous codebook used by Germany during World War I.

For example, to use the codebook in Table 2.3 to encrypt the German word **Februar**, the entire word would be replaced with the 5-digit codeword 13605. This codebook was used for encryption, while the corresponding inverse codebook, arranged with the 5-digit codewords in numerical order, was used for decryption. A codebook is a form of a substitution cipher, but the substitutions are far from simple, since substitutions are for entire words, or in some cases, entire phrases.

The codebook illustrated in Table 2.3 was used to encrypt the famous Zimmermann telegram. At the height of World War I in 1917, the German Foreign Minister, Arthur Zimmermann, sent an encrypted telegram to the German ambassador in Mexico City. The ciphertext message, which appears in Figure 2.4 [227], was intercepted by the British. At the time, the British and French were at war with Germany, but the U.S. was neutral [307].

Table 2.3: Excerpt from a German Codebook

Plaintext	Ciphertext
Februar	13605
fest	13732
finanzielle	13850
folgender	13918
Frieden	17142
Friedenschluss	17149
⋮	⋮

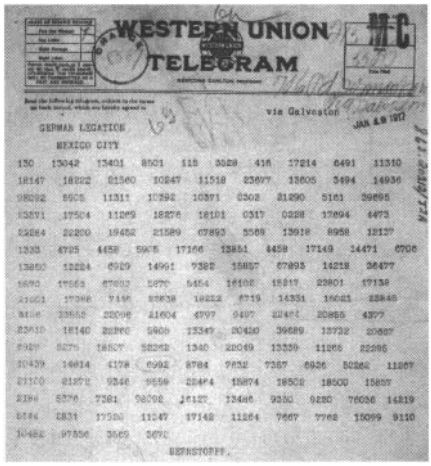

Figure 2.4: The Zimmermann Telegram

The Russians had recovered a damaged version of the German codebook, and the partial codebook had been passed on to the British. Through painstaking analyses, the British were able to fill in the gaps in the codebook so that by the time they obtained the Zimmermann telegram, they could decrypt it [83]. The telegram stated that the German government was planning to begin unrestricted submarine warfare and had concluded that this would likely lead to war with the United States. As a result, Zimmermann told his ambassador that Germany should try to recruit Mexico as an ally to fight against the United States. The incentive for Mexico was that it would "reconquer the lost territory in Texas, New Mexico and Arizona." When the Zimmermann telegram was released in the U.S., public opinion turned against Germany and, after the sinking of the Lusitania, the U.S. declared war.

The British were initially hesitant to release the Zimmermann telegram since they feared that the Germans would realize that their cipher was broken and, presumably, stop using it. However, after decrypting the Zimmermann telegram, the British took a closer look at other intercepted messages that had been sent at about the same time. To their amazement, they found that a variant of the incendiary telegram had been sent unencrypted.[6] The version of the Zimmermann telegram that the British subsequently released closely matched the unencrypted version of the telegram. As the British hoped, the Germans concluded that their codebook had not been compromised and continued to use it for sensitive messages throughout the war.

The security of a classic codebook cipher depends primarily on the physical security of the book itself. That is, the book must be protected from capture by the enemy. In addition, statistical attacks analogous to those used to break a simple substitution cipher apply to codebooks, although the amount of data required is much larger. The reason that a statistical attack on a codebook is more difficult is due to the fact that the size of the "alphabet" is much larger, and consequently far more data must be collected before the statistical information can rise above the noise.

As late as World War II, codebooks were in widespread use. Cryptographers realized that these ciphers were subject to statistical attack, so codebooks needed to be periodically replaced with new codebooks. Since this was an expensive and risky process, techniques were developed to extend the life of a codebook. To accomplish this, a so-called *additive* was generally used.

Suppose that for a particular codebook cipher, the codewords are all 5-digit numbers. Then the corresponding additive book would consist of a long list of randomly generated 5-digit numbers. After a plaintext message had been converted to a series of 5-digit codewords, a starting point in the additive book would be selected and beginning from that point, the sequence of 5-digit additives would be added to the codewords to create the ciphertext. To decrypt, the same additive sequence would be subtracted from the ciphertext before looking up the codeword in the codebook. Note that the additive book—as well as the codebook itself—is required to encrypt or decrypt a message.

Often, the starting point in the additive book was selected at random by the sender and sent in the clear (or in a slightly obfuscated form) at the start of the transmission. This additive information was part of the *message indicator*, or MI. The MI included any non-secret information needed by the intended recipient to decrypt the message.

If the additive material was only used once, the resulting cipher would be equivalent to a one-time pad and therefore, provably secure. However, in

[6] Apparently, the message had not initially attracted attention because it was not encrypted. The lesson here is that, ironically, encryption with a weak cipher may be worse than no encryption at all. We have more to say about this issue in Chapter 10.

practice, the additive was reused many times and, therefore, any messages sent with overlapping additives would have their codewords encrypted with the same key, where the key consists of the codebook and the specific additive sequence. Therefore, any messages with overlapping additive sequences could be used to gather the statistical information needed to attack the underlying codebook. In effect, the additive book dramatically increased the amount of ciphertext required to mount a statistical attack on the codebook, which is precisely the effect the cryptographers had hoped to achieve.

Modern block ciphers use complex algorithms to generate ciphertext from plaintext (and vice versa), but at a higher level, a block cipher can be viewed as a codebook, where each key determines a distinct codebook. That is, a modern block cipher consists of an enormous number of distinct codebooks, with the codebooks indexed by the key. The concept of an additive also lives on, in the form of an initialization vector, or IV, which is often used with block ciphers (and sometimes with stream ciphers as well). Block ciphers are discussed in detail in the next chapter.

2.3.8 Ciphers of the Election of 1876

The U.S. presidential election of 1876 was a virtual dead heat. At the time, the Civil War was still fresh in people's minds, Radical Reconstruction was ongoing in the former Confederacy, and the nation was still bitterly divided.

The contestants in the election were Republican Rutherford B. Hayes and Democrat Samuel J. Tilden. Tilden had obtained a slight plurality of the popular vote, but it is the Electoral College that determines the winner of the presidency. In the Electoral College, each state sends a delegation and for almost every state, the entire delegation is supposed to vote for the candidate who received the largest number of votes in that particular state.[7]

In 1876, the electoral college delegations of four states[8] were in dispute, and these held the balance. A commission of 15 members was appointed to determine which state delegations were legitimate, and thus determine the presidency. The commission decided that all four states should go to Hayes and he became president of the United States. Tilden's supporters immediately charged that Hayes' people had bribed officials to turn the vote in his favor, but no evidence was forthcoming.

Some months after the election, reporters discovered a large number of encrypted messages that had been sent from Tilden's supporters to officials in the disputed states. One of the ciphers used was a partial codebook together

[7]However, there is no legal requirement for an Electoral College delegate to vote for a particular candidate, and on occasion a "faithless elector" will vote contrary to the popular vote in his or her state.

[8]Foreshadowing the election of 2000, one of these four disputed states was, believe it or not, Florida.

Table 2.4: Election of 1876 Codebook

Plaintext	Ciphertext
Greenbacks	Copenhagen
Hayes	Greece
votes	Rochester
Tilden	Russia
telegram	Warsaw
⋮	⋮

with a transposition on the words. The codebook was only applied to important words and the transposition was a fixed permutation for all messages of a given length. The allowed message lengths were 10, 15, 20, 25, and 30 words, with all messages padded to one of these lengths. A snippet of the codebook appears in Table 2.4.

The permutation used for a message of 10 words was

$$9, 3, 6, 1, 10, 5, 2, 7, 4, 8.$$

One actual ciphertext message was

```
Warsaw they read all unchanged last are idiots can't situation
```

which was decrypted by undoing the permutation and substituting `telegram` for `Warsaw` to obtain

```
Can't read last telegram.
Situation unchanged.
They are all idiots.
```

The cryptanalysis of this weak cipher was relatively easy to accomplish [124]. Since a permutation of a given length was used repeatedly, many messages of particular length were in depth—with respect to the permutation as well as the codebook. A cryptanalyst could therefore compare all messages of the same length, making it relatively easy to discover the fixed permutation, even without knowledge of the partial codebook. Of course, the analyst first had to be clever enough to consider the possibility that all messages of a given length were using the same permutation, but, with this insight, the permutations were easily recovered. The codebook was then deduced from context and also with the aid of some unencrypted messages that provided context for the ciphertext messages.

And what did these decrypted messages reveal? The reporters who broke the messages were amused to discover that Tilden's supporters had tried to

bribe officials in the disputed states. The irony here—or not, depending on your perspective—is that Tilden's people were guilty of precisely the same crime of which they had accused Hayes.

By any measure, this cipher was poorly designed and weak. One lesson is that the overuse of a key can be an exploitable flaw. In this case, each time a permutation was reused, it gave the cryptanalyst more information that could be collated to recover the permutation. In modern cipher systems, we try to limit the use of a key so that we do not allow a cryptanalyst to accumulate too much information and to limit the damage if a particular key is exposed.

2.4 Modern Crypto History

Don't let yesterday take up too much of today.
— Abraham Lincoln

Throughout the 20th century, cryptography played an important role in major world events. Late in the 20th century, cryptography became a critical technology for commercial and business communications as well, and it remains so today.

The Zimmermann telegram is one of the first examples from the last century of the role that cryptanalysis has had in political and military affairs. In this section, we mention a few other historical highlights from the past century. For more on the history of cryptography, the best source is Kahn's book [159].

In 1929, Secretary of State Henry L. Stimson ended the U.S. government's official cryptanalytic activity, justifying his actions with the immortal line, "Gentlemen do not read each other's mail" [291]. This would prove to be a costly mistake in the run-up to the attack on Pearl Harbor.

Prior to the Japanese attack of December 7, 1941, the United States had restarted its cryptanalytic programs. The successes of allied cryptanalysts during the World War II era were remarkable, and this period is often seen as the golden age of cryptanalysis. Virtually all significant Axis cryptosystems were broken by the Allies and the value of the intelligence obtained from these systems is difficult to overestimate.

In the Pacific theatre, the so-called Purple cipher was used for high level Japanese government communication. This cipher was broken by American cryptanalysts before the attack on Pearl Harbor, but the intelligence gained (code named MAGIC) provided no clear indication of the impending attack [82]. The Japanese Imperial Navy used a cipher known as JN-25, which was also broken by the Americans. The intelligence from JN-25 was almost

certainly decisive in the extended battle of Coral Sea and Midway, where an inferior American force was able to to halt the advance of the Japanese in the Pacific for the first time. The Japanese Navy was never able to recover from the losses inflicted during this crucial battle.

In Europe, the German Enigma cipher (code named ULTRA) was a major source of intelligence for the Allies during the war [104, 118]. It is often claimed that the ULTRA intelligence was so valuable that Churchill decided not to inform the British city of Coventry of an impending attack by the German Luftwaffe, since the primary source of information on the attack came from Enigma decrypts [69]. Churchill was supposedly concerned that a warning might tip off the Germans that their cipher had been broken. That this did not occur has been well documented. Nevertheless, it was a challenge to utilize valuable ULTRA intelligence without giving away the fact that the Enigma had been broken [42].

The Enigma was initially broken by Polish cryptanalysts. After the fall of Poland, these cryptanalysts escaped to France, but shortly thereafter France fell to the Nazis. The Polish cryptanalysts eventually made their way to England, where they provided their knowledge to British cryptanalysts.[9] A British team that included the computing pioneer, Alan Turing, developed improved attacks on the Enigma [104].

A picture of the Enigma appears in Figure 2.5. Additional details on the inner workings of the Enigma are given in the problems at the end of this chapter and a cryptanalytic attack is presented in Chapter 6.

Figure 2.5: An Enigma Cipher (Courtesy of T. B. Perera and the Enigma Museum)

[9]Remarkably, the Polish cryptanalysts were not allowed to continue their work on the Enigma in Britian.

In the post-World War II era, cryptography slowly moved from a black art into the realm of science. The publication of Claude Shannon's seminal 1949 paper, *Information Theory of Secrecy Systems* [267], marks the turning point. Shannon proved that the one-time pad is secure and he also offered two fundamental cipher design principles: *confusion* and *diffusion*. These two principles have guided symmetric cipher design ever since.

In Shannon's use, confusion is, roughly speaking, defined as obscuring the relationship between the plaintext and ciphertext. On the other hand, diffusion is the idea of spreading the plaintext statistics through the ciphertext. A simple substitution cipher and a one-time pad employ only confusion, whereas a double transposition is a diffusion-only cipher. Since the one-time pad is provably secure, evidently confusion alone is enough, while it appears that diffusion alone is not.

These two concepts—confusion and diffusion—are as relevant today as they were on the day that they were originally published. In subsequent chapters, it will become clear that these concepts remain crucial to modern block cipher design.

Until recently, cryptography was primarily the domain of the government and military. That changed dramatically in the 1970s, due in large part to the computer revolution which led to the need to protect large amounts of electronic data. By the mid-1970s, even the U.S. government realized that there was a legitimate commercial need for secure cryptography. Furthermore, it was clear that the commercial products of the day were severely lacking. So, the National Bureau of Standards, or NBS,[10] issued a request for cryptographic algorithms. The idea was that NBS would select an algorithm that would then become an official U.S. government standard. The ultimate result of this ill-conceived process was a cipher known as the Data Encryption Standard, or DES.

It's impossible to overemphasize the role that DES has played in the modern crypto history. We'll have much more to say about DES in the next chapter.

Post-DES, academic interest in cryptography grew rapidly. Public key cryptography was discovered (or, more precisely, rediscovered) shortly after the arrival of DES. By the 1980s there were annual CRYPTO conferences, which are a consistent source of high-quality work in the field. In the 1990s the Clipper Chip and the development of a replacement for the aging DES were two of the many crypto highlights.

Governments continue to fund major organizations that work in crypto and related fields. However, it's clear that the crypto genie has escaped from its classified bottle, never to be put back.

[10]NBS has since been rechristened as the National Institute of Standards and Technology, or NIST, perhaps in an effort to recycle three-letter acronyms and thereby delay their eventual exhaustion by government agencies.

2.5 A Taxonomy of Cryptography

In the next three chapters, we'll focus on three broad categories of ciphers: *symmetric* ciphers, *public key* cryptosystems, and *hash functions*. Here, we give a very brief overview of these different categories.

Each of the classic ciphers discussed above is a symmetric cipher. Modern symmetric ciphers can be subdivided into *stream ciphers* and *block ciphers*. Stream ciphers generalize the one-time pad approach, sacrificing provable security for a key that is manageable. Block ciphers are, in a sense, the generalization of classic codebooks. In a block cipher, the key determines the codebook, and as long as the key remains fixed, the same codebook is used. Conversely, when the key changes, a different codebook is selected.

While stream ciphers dominated in the post-World War II era, today block ciphers are the kings of the symmetric crypto world—with a few notable exceptions. Generally speaking, block ciphers are easier to optimize for software implementations, while stream ciphers are usually most efficient in hardware.

As the name suggests, in public key crypto, encryption keys can be made public. For each public key, there is a corresponding decryption key that is known as a private key. Not surprisingly, the private key is not public—it must remain private.

If you post your public key on the Internet, anyone with an Internet connection can encrypt a message for you, without any prior arrangement regarding the key. This is in stark contrast to a symmetric cipher, where the participants must agree on a key in advance. Prior to the adoption of public key crypto, secure delivery of symmetric keys was the Achilles heel of modern cryptography. A spectacular case of a failed symmetric key distribution system can be seen in the exploits of the Walker family spy ring. The Walker family sold cryptographic keys used by the U.S. military to the Soviet Union for nearly two decades before being discovered [81, 96]. Public key cryptography does not completely eliminate the key distribution problem, since the private key must be in the hands of the appropriate user, and no one else.

Public key cryptography has another somewhat surprising and extremely useful feature, for which there is no parallel in the symmetric key world. Suppose a message is "encrypted" with the private key instead of the public key. Since the public key is public, anyone can decrypt this message. At first glance such encryption might seem pointless. However, it can serve as a digital form of a handwritten signature—anyone can verify the signature, but only the signer could have created the signature. As with all of these topics, we'll have much more to say about *digital signatures* in a later chapter.

Anything we can do with a symmetric cipher we can also accomplish with a public key cryptosystem. Public key crypto also enables us to do things that cannot be accomplished with a symmetric cipher. So why not use public

key crypto for everything? The primary reason is efficiency—symmetric key crypto is orders of magnitude faster than public key. As a result, symmetric crypto is used to encrypt the vast majority of data today. Yet public key crypto has several critical roles to play in modern information security.

The third major crypto category we'll consider is cryptographic hash functions.[11] These functions take an input of any size and produce an output of a fixed size. In addition, hash functions must satisfy some very stringent requirements. For example, if the input changes in one or more bits, the output should change in about half of its bits. For another, it must be computationally infeasible to find *any* two inputs that hash to the same output. It may not be obvious that such a function is useful—or that such functions actually exist—but we'll see that they do exist and that they turn out to be extremely useful for a surprisingly wide array of problems.

2.6 A Taxonomy of Cryptanalysis

The goal of cryptanalysis is to recover the plaintext, the key, or both. By Kerckhoffs' Principle, we assume that Trudy, the cryptanalyst, has complete knowledge of the inner workings of the algorithm. Another basic assumption is that Trudy has access to the ciphertext—otherwise, why would we bother to encrypt? If Trudy only knows the algorithms and the ciphertext, then she must conduct a *ciphertext only* attack. This is the most disadvantageous possible scenario from Trudy's perspective.

Trudy's chances of success might improve if she has access to *known plaintext*. That is, Trudy might know some of the plaintext and observe the corresponding ciphertext. These matched plaintext-ciphertext pairs might provide information about the key. Of course, if all of the plaintext were known, there would be little point in recovering the key. But it's often the case that Trudy has access to (or can guess) some of the plaintext. For example, many kinds of data include stereotypical headers (email being a good example). If such data is encrypted, the attacker can likely guess some of the plaintext that corresponds to some of the ciphertext.

Surprisingly often, Trudy can actually choose the plaintext to be encrypted and see the corresponding ciphertext. Not surprisingly, this goes by the name of *chosen plaintext* attack. How is it possible for Trudy to choose the plaintext? In later chapters, we'll see that some security protocols encrypt anything that is sent and return the corresponding ciphertext. It's also possible that Trudy could have limited access to a cryptosystem, allowing her to encrypt plaintext of her choice. For example, Alice might forget to log out of her computer when she takes her lunch break. Trudy could then encrypt

[11]Not to be confused with hash functions that you may have seen in other computing contexts.

some selected messages before Alice returns. This type of "lunchtime attack" takes many forms.

Potentially more advantageous for the attacker is an *adaptively chosen plaintext* attack. In this scenario, Trudy chooses the plaintext, views the resulting ciphertext, and chooses the next plaintext based on the observed ciphertext. In some cases, this can make Trudy's job significantly easier.

Related key attacks are also significant in some applications. The idea here is to look for a weakness in the system when the keys are related in some special way.

There are other types of attacks that cryptographers occasionally worry about—mostly when they feel the need to publish another academic paper. In any case, a cipher can only be considered secure if no successful shortcut attack is known.

Finally, there is one particular attack scenario that applies to public key cryptography, but not the symmetric key case. Suppose Trudy intercepts a ciphertext that was encrypted with Alice's public key. If Trudy suspects that the plaintext message was either "yes" or "no," then she can encrypt both of these putative plaintexts with Alice's public key. If either matches the ciphertext, then the message has been broken. This is known as a *forward search*. Although a forward search is not applicable against a symmetric cipher, we'll see that this approach can be used to attack hash functions in some applications.

We've previously seen that the size of the keyspace must be large enough to prevent an attacker from trying all possible keys. The forward search attack implies that in public key crypto, we must also ensure that the size of the plaintext message space is large enough so that the attacker cannot simply encrypt all possible plaintext messages. In practice, this is easy to achieve, as we'll see in Chapter 4.

2.7 Summary

In this chapter we covered several classic cryptosystems, including the simple substitution, the double transposition, codebooks, and the one-time pad. Each of these illustrates some important points that we'll return to again in later chapters. We also discussed some elementary aspects of cryptography and cryptanalysis.

In the next chapter we'll turn our attention to modern symmetric key ciphers. Subsequent chapters cover public key cryptography, hash functions, and cryptanalysis. Cryptography will appear again in later parts of the book. In particular, cryptography is a crucial ingredient in security protocols. Contrary to some authors' misguided efforts, the fact is that there's no avoiding cryptography in information security.

2.8 Problems

1. In the field of information security, Kerckhoffs' Principle is like motherhood and apple pie, all rolled up into one.

   a. Define Kerckhoffs' Principle in the context of cryptography.

   b. Give a real-world example where Kerckhoffs' Principle has been violated. Did this cause any security problems?

   c. Kerckhoffs' Principle is sometimes applied more broadly than its strict cryptographic definition. Give a definition of Kerckhoffs' Principle that applies more generally.

2. Edgar Allan Poe's 1843 short story, "The Gold Bug," features a cryptanalytic attack.

   a. What type of cipher is broken and how?

   b. What happens as a result of this cryptanalytic success?

3. Given that the Caesar's cipher was used, find the plaintext that corresponds to the following ciphertext:

   ```
   VSRQJHEREVTXDUHSDQWU.
   ```

4. Find the plaintext and the key, given the ciphertext

   ```
   CSYEVIXIVQMREXIH.
   ```

   Hint: The key is a shift of the alphabet.

5. Suppose that we have a computer that can test 2^{40} keys each second.

   a. What is the expected time (in years) to find a key by exhaustive search if the keyspace is of size 2^{88}?

   b. What is the expected time (in years) to find a key by exhaustive search if the keyspace is of size 2^{112}?

   c. What is the expected time (in years) to find a key by exhaustive search if the keyspace is of size 2^{256}?

6. The weak ciphers used during the election of 1876 employed a fixed permutation of the words for a given length sentence. To see that this is weak, find the permutation of $(1, 2, 3, \ldots, 10)$ that was used to produce the scrambled sentences below, where "San Francisco" is treated as a single word. Note that the same permutation was used for all three sentences.

first try try if you and don't again at succeed
only you you you as believe old are are as
winter was in the I summer ever San Francisco coldest spent

7. The weak ciphers of the election of 1876 used a partial codebook and a permutation of the words. Modify this approach so that it is more secure.

8. This problem deals with the concepts of confusion and diffusion

   a. Define the terms confusion and diffusion as used in cryptography.

   b. Which classic cipher discussed in this chapter employs only confusion?

   c. Which classic cipher discussed in this chapter employs only diffusion?

   d. Which cipher discussed in this chapter employs both confusion and diffusion?

9. Recover the plaintext and key for the simple substitution example that appears in (2.2) on page 24.

10. Determine the plaintext and key for the ciphertext that appears in the *Alice in Wonderland* quote at the beginning of this chapter. Hint: The message was encrypted with a simple substitution cipher and the plaintext has no spaces or punctuation.

11. Decrypt the following message that was encrypted using a simple substitution cipher:

```
GBSXUCGSZQGKGSQPKQKGLSKASPCGBGBKGUKGCEUKUZKGGBSQEICA
CGKGCEUERWKLKUPKQQGCIICUAEUVSHQKGCEUPCGBCGQOEVSHUNSU
GKUZCGQSNLSHEHIEEDCUOGEPKHZGBSNKCUGSUKUASERLSKASCUGB
SLKACRCACUZSSZEUSBEXHKRGSHWKLKUSQSKCHQTXKZHEUQBKZAEN
NSUASZFENFCUOCUEKBXGBSWKLKUSQSKNFKQQKZEHGEGBSXUCGSZQ
GKGSQKUZBCQAEIISKOXSZSICVSHSZGEGBSQSAHSGKHMERQGKGSKR
EHNKIHSLIMGEKHSASUGKNSHCAKUNSQQKOSPBCISGBCQHSLIMQGKG
SZGBKGCGQSSNSZXQSISQQGEAEUGCUXSGBSSJCQGCUOZCLIENKGCA
USOEGCKGCEUQCGAEUGKCUSZUEGBHSKGEHBCUGERPKHEHKHNSZKGGKAD
```

12. Write a program to help an analyst decrypt a simple substitution cipher. Your program should take the ciphertext as input, compute letter frequency counts, and display these for the analyst. The program should then allow the analyst to guess a key and display the results of the corresponding "decryption" with the putative key.

13. Extend the program described in Problem 12 so that it initially tries to decrypt the message. One sensible way to proceed is to use the computed letter frequencies and the known frequencies of English for an initial guess at the key. Then from the resulting putative decryption, count the number of dictionary words that appear and use this as a score. Next, for each letter in the key, try swapping it with the letter that is adjacent (with respect to frequency counts) and recompute the score. If the score improves, update the key; if not, don't change the putative key. Iterate this process until the score does not improve for an entire pass through the alphabet. At this point you will give your putative decryption to the analyst. To aid the analyst in the manual phase, your program must maintain all of the functionality of the program in Problem 12.

14. Encrypt the message

    we are all together

    using a double transposition cipher (of the type described in the text) with 4 rows and 4 columns, using the row permutation

    $$(1, 2, 3, 4) \longrightarrow (2, 4, 1, 3)$$

    and the column permutation

    $$(1, 2, 3, 4) \longrightarrow (3, 1, 2, 4).$$

15. Decrypt the ciphertext

    ```
    IAUTMOCSMNIMREBOTNELSTRHEREOAEVMWIH
    TSEEATMAEOHWHSYCEELTTEOHMUOUFEHTRFT
    ```

    This message was encrypted with a double transposition (of the type discussed in the text) using a matrix of 7 rows and 10 columns. Hint: The first word is "there."

16. Outline an automated attack on a double transposition cipher (of the type discussed in the text), assuming that the size of the matrix is known.

17. A double transposition cipher can be made much stronger by using the following approach. First, the plaintext is put into an $n \times m$ array, as described in the text. Next, permute the columns, and then write out the intermediate ciphertext column by column. That is, column 1 gives the first n ciphertext letters, column 2 gives the next n, and so on. Then repeat the process, that is, put the intermediate ciphertext

into an $n \times m$ array, permute the columns, and write out the ciphertext column by column. Use this approach, with a 3×4 array, and permutations $(2, 3, 1, 4)$ and $(4, 2, 1, 3)$ to encrypt the plaintext `attackatdawn`.

18. Using the letter encodings in Table 2.1, the following two ciphertext messages were encrypted with the same one-time pad:

    KHHLTK and KTHLLE.

    Find all possible plaintexts for each message and the corresponding one-time pad.

19. Using the letter encodings in Table 2.1, the following ciphertext message was encrypted with a one-time pad:

    KITLKE.

    a. If the plaintext is "thrill," what is the key?

    b. If the plaintext is "tiller," what is the key?

20. Suppose that you have a message consisting of 1024 bits. Design a method that will extend a key that is 64 bits long into a string of 1024 bits, so that the resulting 1024 bits can be XORed with the message, just like a one-time pad. Is the resulting cipher as secure as a one-time pad? Is it possible for any such cipher to be as secure as a one-time pad?

21. Design a codebook cipher that can encrypt any block of bits, not just specific words. Your cipher should include many possible codebooks, with a key used to determine which codebook will be employed to encrypt (or decrypt) a particular message. Discuss some possible attacks on your cipher.

22. Suppose that the following is an excerpt from the decryption codebook for a classic codebook cipher.

123	once
199	or
202	maybe
221	twice
233	time
332	upon
451	a

    Decrypt the following ciphertext:

    242, 554, 650, 464, 532, 749, 567

assuming that the following additive sequence was used to encrypt the message:

119, 222, 199, 231, 333, 547, 346

23. An affine cipher is a type of simple substitution where each letter is encrypted according to the rule $c = (a \cdot p + b) \bmod 26$ (see the Appendix for a discussion of mod). Here, p, c, a, and b are each numbers in the range 0 to 25, where p represents the plaintext letter, c the ciphertext letter, and a and b are constants. For the plaintext and ciphertext, 0 corresponds to "a," 1 corresponds to "b," and so on. Consider the ciphertext `QJKES REOGH GXXRE OXEO`, which was generated using an affine cipher. Determine the constants a and b and decipher the message. Hint: Plaintext "t" encrypts to ciphertext "H" and plaintext "o" encrypts to ciphertext "E."

24. A Vigenère cipher uses a sequence of "shift-by-n" simple substitutions, where the shifts are indexed using a keyword, with "A" representing a shift-by-0, "B" representing a shift-by-1, etc. For example, if the keyword is "DOG," then the first letter is encrypted using a simple substitution with a shift-by-3, the second letter is encrypted using a shift-by-14, the third letter is encrypted using a shift-by-6, and the pattern is repeated—the fourth letter is encrypted using a shift-by-3, the fifth letter is encrypted using a shift-by-14, and so on. Cryptanalyze the following ciphertext, i.e., determine the plaintext and the key. This particular message was encrypted using a Vigenère cipher with a 3-letter English keyword:

```
CTMYR DOIBS RESRR RIJYR EBYLD IYMLC CYQXS RRMLQ FSDXF
OWFKT CYJRR IQZSM X
```

25. Suppose that on the planet Binary, the written language uses an alphabet that contains only two letters X and Y. Also, suppose that in the Binarian language, the letter X occurs 75% of the time, while Y occurs 25% of the time. Finally, assume that you have two messages in the Binary language, and the messages are of equal length.

    a. If you compare the corresponding letters of the two messages, what fraction of the time will the letters match?

    b. Suppose that one of the two messages is encrypted with a simple substitution, where X is encrypted as Y and Y is encrypted as X. If you now compare the corresponding letters of the two messages—one encrypted and one not—what fraction of the time will the letters match?

c. Suppose that both of the messages are encrypted with a simple substitution, where X is encrypted as Y and Y is encrypted as X. If you now compare the corresponding letters of the two messages—both of which are encrypted—what fraction of the time will the letters match?

d. Suppose instead that you are given two randomly generated "messages" that use only the two letters X and Y. If you compare the corresponding letters of the two messages, what fraction of the time will the letters match?

e. What is the index of coincidence (IC)? Hint: See, for example, [148].

f. How can the index of coincidence be used to determine the length of the keyword in a Vigenère cipher (see Problem 24 for the definition of a Vigenère cipher)?

26. In the this chapter, we discussed a forward search attack.

a. Explain how to conduct a forward search attack.

b. How can you prevent a forward search attack against a public key cryptosystem?

c. Why can't a forward search attack be used to break a symmetric cipher?

27. Consider a "one-way" function h. Then, given the value $y = h(x)$, it is computationally infeasible to find x directly from y.

a. Suppose that Alice computes $y = h(x)$, where x is Alice's salary, in dollars. If Trudy obtains y, how can she determine Alice's salary x? Hint: Adapt the forward search attack to this problem.

b. Why does your attack not violate the one-way property of h?

c. How could Alice prevent this attack ? We assume that Trudy has access to the output of the function h, Trudy knows that the input includes Alice's salary, and Trudy knows the format of the input. Also, no keys are available, so Alice cannot encrypt the output value.

28. Suppose that a particular cipher uses a 40-bit key, and the cipher is secure (i.e., there is no known shortcut attack).

a. How much work, on average, is an exhaustive search attack?

b. Outline an attack, assuming that known plaintext is available.

c. How would you attack this cipher in the ciphertext-only case?

29. Suppose that Alice encrypted a message with a secure cipher that uses a 40-bit key. Trudy knows the ciphertext and Trudy knows the algorithm, but she does not know the plaintext or the key. Trudy plans to do an exhaustive search attack, that is, she will try each possible key until she finds the correct key.

    a. How many keys, on average, must Trudy try before she finds the correct one?

    b. How will Trudy know when she has found the correct key? Note that there are too many solutions for Trudy to manually examine each one—she must have some automated approach to determining whether a putative key is correct or not.

    c. How much work is your automated test in part b?

    d. How many false alarms do you expect from your test in part b? That is, how often will an incorrect key produce a putative decrypt that will pass your test?