Chapter 2

Cryptographic Techniques

2.1 INTRODUCTION

This chapter introduces the basic concepts in **cryptography**. Although this word sounds fearful, we shall realize that it is very simple to understand. In fact, most terms in computer security have very straightforward meaning. Many terms, for no reason, sound complicated. Our aim will be to demystify all such terms in relation to cryptography in this chapter. After we are through with this chapter, we shall be ready to understand computer-based security solutions and issues that follow in later chapters.

Note Cryptography is the art of achieving security by encoding messages to make them non-readable.

In the early days, cryptography used to be performed by using manual techniques. The basic framework of performing cryptography has remained more or less the same, of course, with a lot of improvements in the actual implementation. More importantly, computers now perform these cryptographic functions/algorithms, thus making the process a lot faster and secure. This chapter, however, discusses the basic methods of achieving cryptography without referring to computers.

The basic concepts in cryptography are introduced first. We then proceed to discuss how we can make messages illegible, and thus, secure. This can be done in many ways. We discuss all these approaches in this chapter. Modern computer-based cryptography solutions have actually evolved based on these premises. This chapter touches upon all these cryptography algorithms. We also discuss the relative advantages and disadvantages of the various algorithms, as and when applicable.

Some cryptography algorithms are very trivial to understand, replicate, and therefore, crack. Some other cryptography algorithms are highly complicated, and therefore, difficult to crack. The rest are somewhere in the middle. A detailed discussion of these is highly essential in cementing our concepts that we shall keep referring to when we actually discuss computer-based cryptography solutions in later chapters.

2.2 PLAIN TEXT AND CIPHER TEXT

Any communication in the language that you and I speak—that is the human language, takes the form of **plain text** or **clear text**. That is, a message in plain text can be understood by anybody knowing the language as long as the message is not codified in any manner. For instance, when we speak with our family members, friends or colleagues, we use plain text because we do not want to hide anything from them. Suppose I say "Hi Anita", it is plain text because both Anita and I know its meaning and intention. More significantly, anybody in the same room would also get to hear these words, and would know that I am greeting Anita.

Notably, we also use plain text during electronic conversations. For instance, when we send an email to someone, we compose the email message using English (or these days, another) language. For instance, I can compose the email message as shown in Fig. 2.1.

```
Hi Amit

Hope you are doing fine. How about meeting at the train station this Friday at 5 pm?
Please let me know if it is ok with you.

Regards.

Atul
```

Fig. 2.1 *Example of a plain text message*

Now, not only Amit, but also any other person who reads this email would know what I have written. As before, this is simply because I am not using any codified language here. I have composed my email message using plain English. This is another example of plain text, albeit in written form.

Note Clear text or plain text signifies a message that can be understood by the sender, the recipient, and also by anyone else who gets an access to that message.

In normal life, we do not bother much about the fact that someone could be overhearing us. In most cases, that makes little difference to us because the person overhearing us can do little damage by using the overheard information. After all, we do not reveal many secrets in our day-to-day lives.

However, there are situations where we are concerned about the secrecy of our conversations. For instance, suppose that I am interested in knowing my bank account's balance and hence I call up my phone banker from my office. The phone banker would generally ask a secret question (e.g. What is your mother's maiden name?) whose answer only I know. This is to ascertain that someone else is not posing as me. Now, when I give the answer to the secret question (e.g. Meena), I generally speak in low voice, or better yet, initially call up from a phone that is isolated. This ensures that only the intended recipient (the phone banker) gets to know the correct answer.

On the same lines, suppose that my email to my friend Amit earlier shown in Fig. 2.1 is confidential for some reason. Therefore, I do not want anyone else to understand what I have written even if she is able to access the email by using some means, before it reaches Amit. How do I ensure this? This is exactly the problem that small children face. Many times, they want to communicate in such a manner that their little secrets are hidden from the elderly. What do they do in order to achieve this? Usually the simplest trick that they use is a code language. For instance, they replace each alphabet in their conversation with another alphabet. As an example, they replace each alphabet with the alphabet that is actually three alphabets down the order. So, each A will be replaced by D, B will be replaced by E, C will be replaced by F, and so on. To complete the cycle, each W will be replaced by Z, each X will be replaced by A, each Y will be replaced by B and each Z will be replaced by C. We can summarize this scheme as shown in Fig. 2.2. The first row shows the original alphabets, and the second row shows what each original alphabet will be replaced with.

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
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

Fig. 2.2 *A scheme for codifying messages by replacing each alphabet with an alphabet three places down the line*

Thus, using the scheme of replacing each alphabet with the one that is three places down the line, a message *I love you* shall become *L ORYH BRX* as shown in Fig. 2.3.

I		L	O	V	E		Y	O	U
L		O	R	Y	H		B	R	X

Fig. 2.3 *Codification using the alphabet replacement scheme*

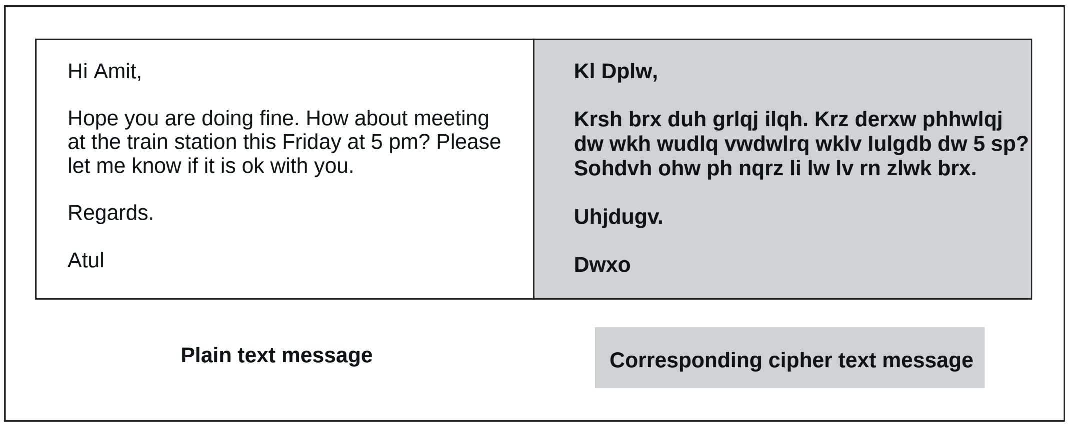
Of course, there can be many variants of such a scheme. It is not necessary to replace each alphabet with the one that is three places down the order. It can be the one that is four, five or more places down the order. The point is, however, that each alphabet in the original message can be replaced by another to hide the original contents of the message. The codified message is called as **cipher text**. Cipher means a code or a secret message.

Note When a plain text message is codified using any suitable scheme, the resulting message is called as cipher text.

Let us now write our original email message and the resulting cipher text by using the alphabet-replacing scheme, as shown in Fig. 2.4. This will clarify the idea further.

As shown in Fig. 2.5, there are two primary ways in which a plain text message can be codified to obtain the corresponding cipher text: **Substitution** and **Transposition**.

Let us discuss these two approaches now. Note that when the two approaches are used together, we call the technique as **product cipher.**



Hi Amit,

Hope you are doing fine. How about meeting at the train station this Friday at 5 pm? Please let me know if it is ok with you.

Regards.

Atul

Kl Dplw,

Krsh brx duh grlqj ilqh. Krz derxw phhwlqj dw wkh wudlq vwdwlrq wklv Iulgdb dw 5 sp? Sohdvh ohw ph nqrz li lw lv rn zlwk brx.

Uhjdugv.

Dwxo

Plain text message
Corresponding cipher text message

Fig. 2.4 *Example of a plain text message being transformed into cipher text*

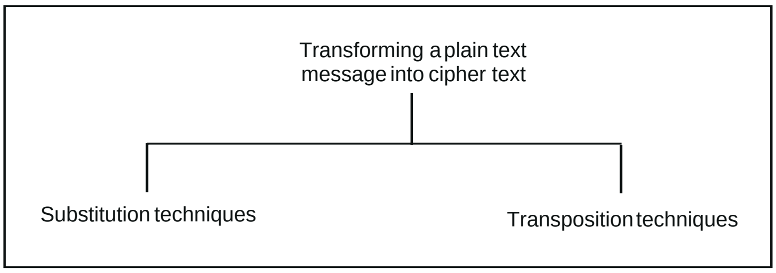


Fig. 2.5 *Techniques for transforming plain text to cipher text*

2.3 SUBSTITUTION TECHNIQUES

2.3.1 Caesar Cipher

The scheme illustrated in Fig. 2.2 was first proposed by Julius Caesar, and is termed as **Caesar Cipher**. It was the first example of substitution cipher. In the substitution cipher technique, the characters of a plain text message are replaced by other characters, numbers or symbols. Caesar Cipher is a special case of substitution techniques wherein each alphabet in a message is replaced by an alphabet three places down the line. For instance, using the Caesar Cipher, the plain text ATUL will become cipher text DWXO.

Note In the substitution cipher technique, the characters of a plain text message are replaced by other characters, numbers or symbols.

Clearly, the Caesar Cipher is a very weak scheme of hiding plain text messages. All that is required to break the Caesar Cipher is to do the reverse of the Caesar Cipher process—i.e. replace each alphabet in a cipher text message produced by Caesar Cipher with the alphabet that is three places up the line. Thus, to work backwards, take a cipher text produced by Caesar Cipher, and replace each A with X, B with Y, C with Z, D with A, E with B and so on. The simple algorithm required to break Caesar Cipher can be summarized as shown in Fig. 2.6.

1. Read each alphabet in the cipher text message, and search for it in the second row of the Fig. 2.2.
2. When a match is found, replace that alphabet in the cipher text message with the corresponding alphabet in the same column but the first row of the table (e.g. if the alphabet in cipher text is J, replace it with G).
3. Repeat the process for all alphabets in the cipher text message.

Fig. 2.6 *Algorithm to break caesar cipher*

The process shown above will reveal the original plain text. Thus, given a cipher text message *L ORYH BRX*, it is easy to work backwards and obtain the plain text *I LOVE YOU* as shown in Fig. 2.7.

Cipher text	L		O	R	Y	H		B	R	X
Plain text	I		L	O	V	E		Y	O	U

Fig. 2.7 *Example of breaking caesar cipher*

2.3.2 Modified Version of Caesar Cipher

Caesar Cipher is good in theory, but not so good in practice. Let us now try and complicate the Caesar Cipher to make an attacker's life difficult. How can we generalize Caesar Cipher a bit more? Let us assume that the cipher text alphabets corresponding to the original plain text alphabets may not necessarily be three places down the order, but instead, can be *any* places down the order. This can complicate matters a bit.

Thus, we are now saying that an alphabet A in plain text would not necessarily be replaced by D. It can be replaced by any valid alphabet, i.e. by E or by F or by G, and so on. Once the replacement scheme is decided, it would be constant and will be used for all other alphabets in that message. As we know, the English language contains 26 alphabets. Thus, an alphabet A can be replaced by any *other* alphabet in the English alphabet set, (i.e. B through Z). Of course, it does not make sense to replace an alphabet by itself (i.e. replacing A with A). Thus, for each alphabet, we have 25 possibilities of replacement. Hence, to break a message in the modified version of Caesar Cipher, our earlier algorithm would not work. Let us write a new algorithm to break this version of Caesar Cipher, as shown in Fig. 2.8.

1. Let k be a number equal to 1.
2. Read the complete cipher text message.
3. Replace each alphabet in the cipher text message with an alphabet that is k positions down the order.
4. Increment k by 1.
5. If k is less than 26, then go to step 2. Otherwise, stop the process.
6. The original text message corresponding to the cipher text message is one of the 25 possibilities produced by the above steps.

Fig. 2.8 *Algorithm to break the modified caesar cipher*

Let us take a cipher text message produced by the modified Caesar Cipher, and try breaking it to obtain the original plain text message by applying the algorithm shown in Fig. 2.8. Since each alphabet in the plain text can be potentially replaced by any of the other 25 alphabets, we have 25 possible plain text messages to choose from. Thus, the output produced by the above algorithm to break a cipher text message *KWUM PMZN* is shown in Table 2.1.

Table 2.1 *Attempts to break modified Caesar cipher text using all possibilities*

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

We can see that the cipher text shown in the first row of the figure needs 25 different attempts to break in, as depicted by the algorithm shown earlier. As it turns out, the 18th attempt reveals the correct plain text corresponding to the cipher text. Therefore, we can actually stop at this juncture. For the sake of completeness, however, we have shown all the 25 steps, which is, of course, the worst possible case.

A mechanism of encoding messages so that they can be sent securely is called as cryptography. Let us take this opportunity to introduce a few terms used in cryptography. An attack on a cipher text message, wherein the attacker attempts to use all possible permutations and combinations, is called as a **Brute-force attack**. The process of trying to break any cipher text message to obtain the original plain text message itself is called as **Cryptanalysis**, and the person attempting a cryptanalysis is called as a **cryptanalyst**.

Note Cryptanalyst is a person who attempts to break a cipher text message to obtain the original plain text message. The process itself is called as cryptanalysis.

As we have noticed, even the modified version of the Caesar Cipher is not very secure. After all, the cryptanalyst needs to be aware of only the following points to break a cipher text message using the Brute-force attack, in this scheme:

1. Substitution technique was used to derive the cipher text from the original plain text.
2. There are only 25 possibilities to try out.
3. The language of the plain text was English.

Note A cryptanalyst attempting a Brute-force attack tries all possibilities to derive the original plain text message from a given cipher text message.

Anyone armed with this knowledge can easily break a cipher text produced by the modified version of Caesar Cipher. How can we make even the modified Caesar Cipher tough to crack?

2.3.3 Mono-alphabetic Cipher

The major weakness of the Caesar Cipher is its predictability. Once we decide to replace an alphabet in a plain text message with an alphabet that is k positions up or down the order, we replace all other alphabets in the plain text message with the same technique. Thus, the cryptanalyst has to try out a maximum of 25 possible attacks, and she is assured of a success.

Now imagine that rather than using a uniform scheme for all the alphabets in a given plain text message, we decide to use random substitution. This means that in a given plain text message, each A can be replaced by any other alphabet (B through Z), each B can also be replaced by any other random alphabet (A or C through Z), and so on. The crucial difference being, there is no relation between the replacement of B and replacement of A. That is, if we have decided to replace each A with D, we need not necessarily replace each B with E—we can replace each B with any other character!

To put it mathematically, we can now have any permutation or combination of the 26 alphabets, which means $(26 \times 25 \times 24 \times 23 \times \ldots 2)$ or 4×10^{26} possibilities! This is extremely hard to crack. It might actually take years to try out these many combinations even with the most modern computers.

Note Mono-alphabetic ciphers pose a difficult problem for a cryptanalyst because it can be very difficult to crack thanks to the high number of possible permutations and combinations.

There is only one hitch. If the cipher text created with this technique is short, the cryptanalyst can try different attacks based on her knowledge of the English language. As we know, some alphabets in the English language occur more frequently than others.

Language analysts have found that given a single alphabet in cipher text, the probability that it is a P is 13.33%—the highest. After P comes Z, which is likely to occur 11.67%. The probability that the alphabet is C, K, L, N or R is almost 0—the lowest.

A cryptanalyst looks for patterns of alphabets in a cipher text, substitutes the various available alphabets in place of cipher text alphabets, and then tries her attacks.

Apart from single-alphabet replacements, the cryptanalyst also looks for repeated patterns of words *to* try the attacks. For example, the cryptanalyst might look for two-alphabet cipher text patterns since the word *to* occurs very frequently in English. If the cryptanalyst finds that two alphabet combinations are found frequently in a cipher text message, she might try and replace all of them with *to*, and then try and deduce the remaining alphabets/words. Next, the cryptanalyst might try to find repeating three-alphabet patterns and try and replace them with the word *the*, and so on.

2.3.4 Homophonic Substitution Cipher

The **Homophonic Substitution Cipher** is very similar to Mono-alphabetic Cipher. Like a plain substitution cipher technique, we replace one alphabet with another in this scheme. However, the difference between the two techniques is that the replacement alphabet set in case of the simple substitution techniques is fixed (e.g. replace A with D, B with E, etc.), whereas in the case of Homophonic Substitution Cipher, one plain text alphabet can map to more than one cipher text alphabet. For instance, A can be replaced by D, H, P, R; B can be replaced by E, I, Q, S, etc.

Note Homophonic Substitution Cipher also involves substitution of one plain text character with a cipher text character at a time, however the cipher text character can be any one of the chosen set.

2.3.5 Polygram Substitution Cipher

In **Polygram Substitution Cipher** technique, rather than replacing one plain text alphabet with one cipher text alphabet at a time, a block of alphabets is replaced with another block. For instance, HELLO could be replaced by YUQQW, but HELL could be replaced by a totally different cipher text block TEUI, as shown in Fig. 2.9. This is true in spite the first four characters of the two blocks of text (HELL) being the same. This shows that in Polygram Substitution Cipher, the replacement of plain text happens block-by-block, rather than character-by-character.

Note Polygram Substitution Cipher technique replaces one block of plain text with a block of cipher text—it does not work on a character-by-character basis.

2.3.6 Polyalphabetic Substitution Cipher

Leon Battista invented the Polyalphabetic Substitution Cipher in 1568. This cipher has been broken many times, and yet it has been used extensively. The Vigenere Cipher and the Beaufort Cipher are examples of Polyalphabetic Substitution Cipher.

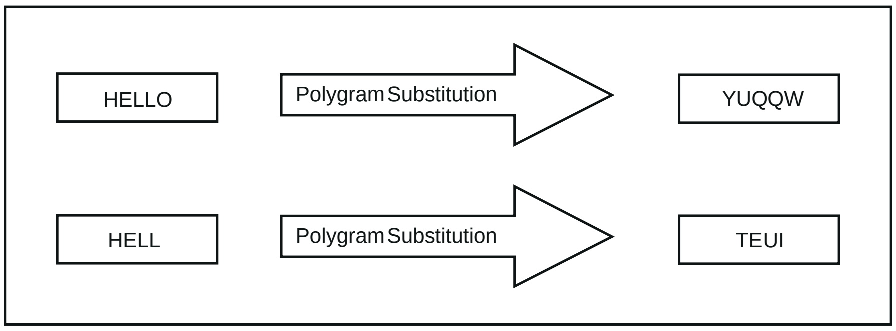


Fig 2.9 *Polygram substitution*

This cipher uses multiple one-character keys. Each of the keys encrypts one plain text character. The first key encrypts the first plain text character; the second key encrypts the second plain text character, and so on. After all the keys are used, they are recycled. Thus, if we have 30 one-letter keys, every 30th character in the plain text would be replaced with the same key. This number (in this case, 30) is called as the **period** of the cipher.

2.4 TRANSPOSITION TECHNIQUES

As we discussed, substitution techniques focus on substituting a plain text alphabet with a cipher text alphabet. Transposition techniques differ from substitution techniques in the way that they do not simply replace one alphabet with another: they also perform some permutation over the plain text alphabets.

2.4.1 Rail Fence Technique

The **Rail Fence Technique** is an example of transposition. It uses a simple algorithm as shown in Fig. 2.10.

1. Write down the plain text message as a sequence of diagonals.
2. Read the plain text written in *step 1* as a sequence of rows.

Fig. 2.10 *Rail fence technique*

Let us illustrate the Rail Fence Technique with a simple example. Suppose that we have a plain text message *Come home tomorrow*. How would we transform that into a cipher text message using the Rail Fence Technique? This is shown in Fig. 2.11.

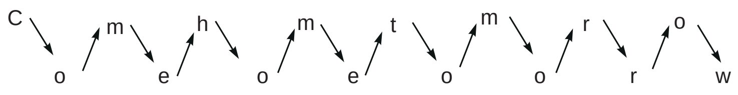
As the figure shows, the plain text message *Come home tomorrow* transforms into *Cmhmtmrooeoeoorw* with the help of Rail Fence Technique.

Note Rail fence technique involves writing plain text as sequence of diagonals and then reading it row-by-row to produce cipher text.

It should be quite clear that the Rail Fence Technique is quite simple for a cryptanalyst to break into. It has very little sophistication built in.

Original plain text message: ***Come home tomorrow***

1. After we arrange the plain text message as a sequence of diagonals, it would look as follows (write the first character on the first line i.e. *C,* then second character on the second line, i.e. *o*, then the third character on the first line, i.e. *m*, then the fourth character on the second line, i.e. *e*, and so on). This creates a zigzag sequence, as shown below.



2. Now read the text row-by-row, and write it sequentially. Thus, we have: ***Cmhmtmrooeoeoorw*** as the cipher text.

Fig. 2.11 *Example of rail fence technique*

2.4.2 Simple Columnar Transposition Technique

1. *Basic technique*

Variations of the basic transposition technique such as Rail Fence Technique exist. Such a scheme is shown in Fig. 2.12, which we shall call as **Simple Columnar Transposition Technique**.

1. Write the plain text message row-by-row in a rectangle of a pre-defined size.
2. Read the message column-by-column. However, it need not be in the order of columns 1, 2, 3 etc. It can be any random order such as 2, 3, 1, etc.
3. The message thus obtained is the cipher text message.

Fig. 2.12 *Simple columnar transposition technique*

Let us examine the Simple Columnar Transposition Technique with an example. Consider the same plain text message *Come home tomorrow*. Let us understand how it can be transformed into cipher text using this technique. This is illustrated in Fig. 2.13.

Original plain text message: ***Come home tomorrow***

1. Let us consider a rectangle with six columns. Therefore, when we write the message in the rectangle row-by-row (suppressing spaces), it would look as follows:

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
C	o	m	e	h	o
m	e	t	o	m	o
r	r	o	w		

2. Now, let us decide the order of columns as some random order, say 4, 6, 1, 2, 5, 3. Then read the text in the order of these columns.
3. The cipher text thus obtained would be eowoocmroerhmmto.

Fig. 2.13 *Example of simple columnar transposition technique*

Note✍ The Simple Columnar Transposition Technique simply arranges the plain text as a sequence of rows of a rectangle that are read in columns randomly.

Like the Rail Fence Technique, the Simple Columnar Transposition Technique is also quite simple to break into. It is a matter of trying out a few permutations and combinations of column orders to get hold of the original plain text. To make matters complex for a cryptanalyst, we can modify the Simple Columnar Transposition Technique to add another twist: perform more than one rounds of transposition using the same technique.

2. *Simple columnar transposition technique with multiple rounds*

To improve the basic Simple Columnar Transposition Technique, we can introduce more complexity. The idea is to use the same basic procedure as used by the Simple Columnar Transposition Technique, but do it more than once. That adds considerably more complexity for the cryptanalyst.

The basic algorithm used in this technique is shown in Fig. 2.14.

1. Write the plain text message row-by-row in a rectangle of a pre-defined size.
2. Read the message column-by-column. However, it need not be in the order of columns 1, 2, 3 etc. It can be any random order such as 2, 3, 1, etc.
3. The message thus obtained is the cipher text message of round 1.
4. Repeat steps 1 to 3 as many times as desired.

Fig. 2.14 *Simple columnar transposition technique with multiple rounds*

As we can see, the only addition in this technique to the basic Simple Columnar Transposition Technique is step 4, which results in the execution of the basic algorithm on more than one occasion. Although this sounds trivial, in reality, it makes the cipher text far more complex as compared to the basic Simple Columnar Transposition Technique. Let us extend our earlier example to now have multiple rounds of transposition, as shown in Fig. 2.15.

As the Fig. 2.15 shows, multiple rounds or iterations add more complexity to the cipher text produced by the basic Simple Columnar Transposition Technique. More the number of iterations, more complex is the cipher text thus produced.

Note✍ Cipher text produced by the Simple Columnar Transposition Technique with multiple rounds is much more complex to crack as compared to the basic technique.

2.4.3 Vernam Cipher (One-Time Pad)

The **Vernam Cipher**, also called as **One-Time Pad**, is implemented using a random set of non-repeating characters as the input cipher text. The most significant point here is that once an input cipher text for transposition is used, it is never used again for any other message (hence the name *one-time*). The length of the input cipher text is equal to the length of the original plain text. The algorithm used in Vernam Cipher is described in Fig. 2.16.

Original plain text message: ***Come home tomorrow***

1. Let us consider a rectangle with six columns. Therefore, when we write the message in the rectangle row-by-row, it would look as follows:

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
C	o	m	e	h	o
m	e	t	o	m	o
r	r	o	w		

2. Now, let us decide the order of columns as some random order, say 4, 6, 1, 2, 5, 3. Then read the text in the order of these columns.
3. The cipher text thus obtained would be ***eowoocmroerhmmto*** in round 1.
4. Let us perform steps 1 through 3 once more. So, the tabular representation of the cipher text after round 1 is as follows:

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
e	o	w	o	o	c
m	r	o	e	r	h
m	m	t	o		

5. Now, let us use the same order of columns, as before, that is 4, 6, 1, 2, 5, and 3. Then read the text in the order of these columns.
6. The cipher text thus obtained would be ***oeochemmormorwot*** in round 2.
7. Continue like this if more number of iterations is desired, otherwise stop.

Fig. 2.15 *Example of simple columnar transposition technique with multiple rounds*

1. Treat each plain text alphabet as a number in an increasing sequence, i.e. A = 0, B = 1, … Z = 25.
2. Do the same for each character of the input cipher text.
3. *Add* each number corresponding to the plain text alphabet to the corresponding input cipher text alphabet number.
4. If the sum thus produced is greater than 26, subtract 26 from it.
5. Translate each number of the sum back to the corresponding alphabet. This gives the output cipher text.

Fig. 2.16 *Algorithm for vernam cipher*

Let us apply the Vernam Cipher algorithm to a plain text message *HOW ARE YOU* using a one-time pad *NCBTZQARX* to produce a cipher text message *UQXTUYFR* as shown in Fig. 2.17.

It should be clear that since the one-time pad is discarded after a single use, this technique is highly secure and suitable for small plain text message, but is clearly impractical for large messages. The Vernam Cipher was first implemented at AT&T with the help of a device called as the Vernam Machine.

Note Vernam Cipher uses a one-time pad, which is discarded after a single use, and therefore, is suitable only for short messages.

	H	O	W	A	R	E	Y	O	U
1. Plain text	7	14	22	0	17	4	24	14	20
+									
2. One-time pad	13	2	1	19	25	16	0	17	23
	N	C	B	T	Z	Q	A	R	X
3. Initial Total	20	16	23	19	42	20	24	31	43
4. Subtract 26, if > 25	20	16	23	19	16	20	24	5	17
5. Cipher text	U	Q	X	T	Q	U	Y	F	R

Fig. 2.17 *Example of vernam cipher*

2.4.4 Book Cipher/Running Key Cipher

The idea used in **Book Cipher**, also called as **Running Key Cipher** is quite simple, and is similar in principle to the Vernam Cipher. For producing cipher text, some portion of text from a book is used, which serves the purpose of a one-time pad. Thus, the characters from a book are used as one-time pad, and they are *added* to the input plain text message similar to the way a one-time pad works.

2.5 ENCRYPTION AND DECRYPTION

We have discussed the concepts of plain text and how it can be transformed into cipher text so that only the sender and the recipient can make any sense out of it. There are technical terms to describe these concepts, which we shall learn now. In technical terms, the process of encoding plain text messages into cipher text messages is called as **encryption**. Figure 2.18 illustrates the idea.

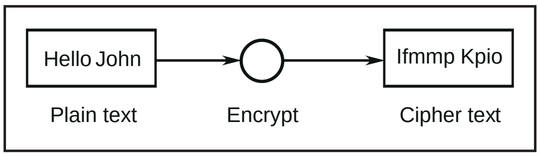


Fig. 2.18 *Encryption*

The reverse process of transforming cipher text messages back to plain text messages is called as **decryption**. Figure 2.19 illustrates the idea.

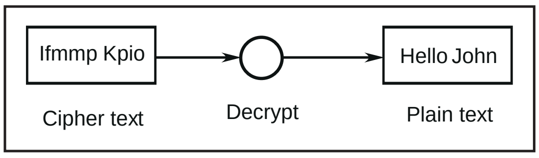


Fig. 2.19 *Decryption*

Note Decryption is exactly opposite of encryption. Encryption transforms a plain text message into cipher text, whereas decryption transforms a cipher text message back into plain text.

In computer-to-computer communications, the computer at the sender's end usually transforms a plain text message into cipher text by performing encryption. The encrypted cipher text message is then sent to the receiver over a network (such as the Internet, although it can be any other network). The receiver's computer then takes the encrypted message, and performs the reverse of encryption, i.e. it performs the decryption process to obtain the original plain text message. This is shown in Fig. 2.20.

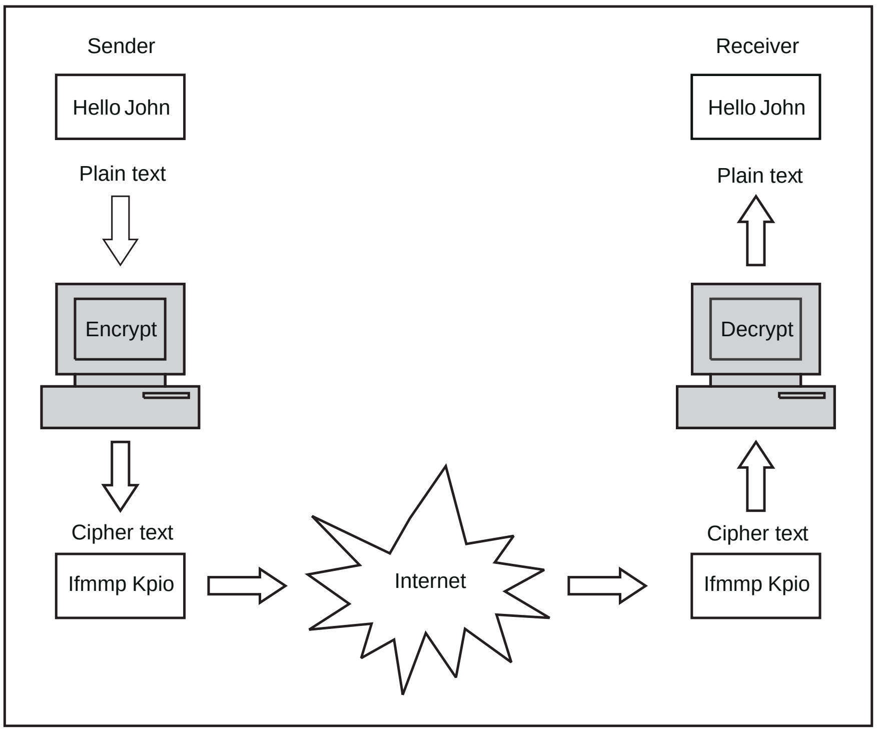


Fig. 2.20 *Encryption and decryption in the real world*

To encrypt a plain text message, the sender (we shall henceforth treat the term *sender* to mean the *sender's computer*) performs encryption, i.e. applies the encryption algorithm. To decrypt a received encrypted message, the recipient performs decryption, i.e. applies the **decryption algorithm**. The algorithm is similar in concept to the algorithms we discussed earlier.

Clearly, the decryption algorithm must be the same as the **encryption algorithm**. Otherwise, decryption would not be able to retrieve the original message. For instance, if the sender uses the Rail Fence Technique for encryption and the receiver uses the Simple Columnar technique for decryption, the decryption would yield a totally incorrect plain text. Thus, the sender and the receiver must agree on a common algorithm for any meaningful communication to take place. The algorithm basically takes one text as input and produces another as the output.

The second aspect of performing encryption and encryption of messages is the **key**. What is a key? A key is something similar to the one time pad used in the Vernam Cipher. Anyone can use the Vernam Cipher. However, as long as only the sender and the receiver know the one time pad, no one except the sender and the receiver can do anything with the message.

Note Every encryption and decryption process has two aspects: *the* algorithm *and the* key *used for encryption and decryption*.

This is shown in Fig. 2.21.

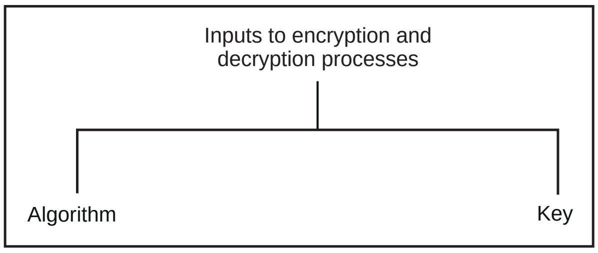


Fig. 2.21 *Aspects of encryption and decryption*

Thus, as an example, the sender and receiver can safely agree to use Vernam Cipher as the algorithm, and XYZ as the key, and be assured that no one else is able to get any access to their conversation. Others might know that the Vernam Cipher is in use. However, they do not know that XYZ is the encryption/decryption key.

Note In general, the algorithm used for encryption and decryption processes is usually known to everybody. However, it is the key used for encryption and decryption that makes the process of cryptography secure.

Broadly, there are two cryptographic mechanisms, depending on what keys are used. If the *same* key is used for encryption and decryption, we call the mechanism as **Symmetric Key Cryptography**. However, if two *different* keys are used in a cryptographic mechanism, wherein one key is used for encryption, and *another, different* key is used for decryption, we call the mechanism as **Asymmetric Key Cryptography**. This is shown in Fig. 2.22.

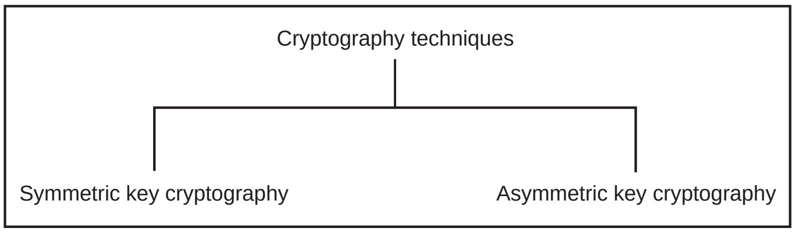


Fig. 2.22 *Cryptography techniques*

We shall study the basic concepts behind these two mechanisms now. We shall study the various computer-based cryptographic algorithms in each of these categories in great detail in subsequent chapters.

Note Symmetric Key Cryptography involves the usage of the same key for encryption and decryption. Asymmetric Key Cryptography involves the usage of one key for encryption, and another, different key for decryption.

2.6 SYMMETRIC AND ASYMMETRIC KEY CRYPTOGRAPHY

2.6.1 Symmetric Key Cryptography and the Problem of Key Distribution

Before we discuss computer-based symmetric and asymmetric key cryptographic algorithms (in the next few chapters), we need to understand why we need two different types of cryptographic algorithms in the first place. To understand this, let us consider a simple problem statement.

Person A wants to send a highly confidential letter to another person B. A and B both reside in the same city, but are separated by a few miles, and for some reason, cannot meet each other.

Let us now see how we can tackle this problem. The simplest solution would appear to be that A puts the confidential letter in an envelope, seals it, and sends it by post. A hopes that no one opens it before it reaches B. This is shown in Fig. 2.23.

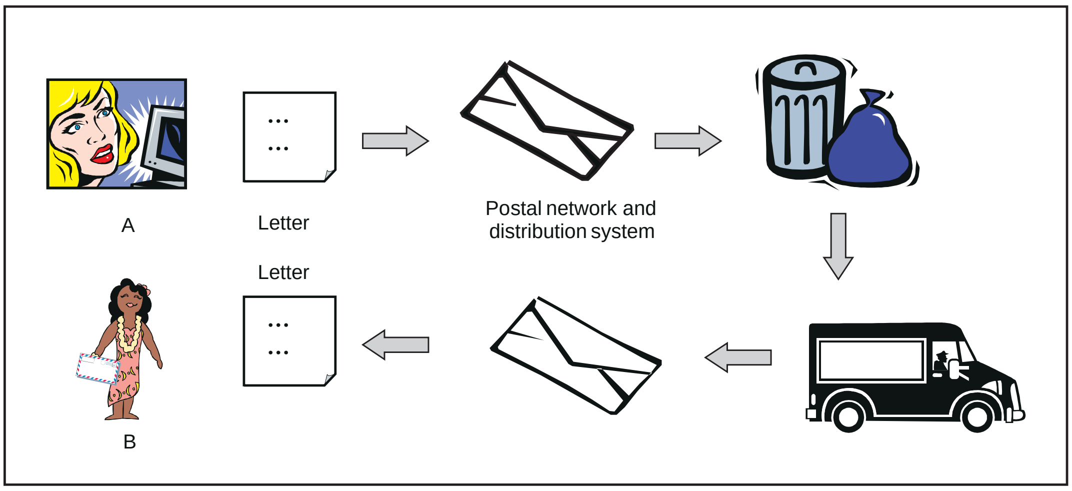


Fig. 2.23 *Simplest way to send a confidential letter*

Clearly, this solution does not seem to be acceptable. What is the guarantee that an unscrupulous person does not obtain and open the envelope before it reaches B? Sending the envelope by registered post or courier might slightly improve the situation, but will not guarantee that the envelope does not get opened before it reaches B. After all, someone can open the envelope, read the confidential letter and reseal the envelope.

Another option is to send the envelope via a hand-delivery mechanism. Here, A hands the envelope over to another person P, who personally hand-delivers the envelope to B. This seems to be a slightly better solution. However, it is still not full proof.

Consequently, A comes up with another idea. A now puts the envelope inside a box, seals that box with a highly secure lock, and sends the box to B (through the mechanism of post/courier/hand-delivery). Since the lock is highly secure, nobody can open the box while in transit, and therefore, open the envelope. Consequently, nobody will be able to read/access the highly confidential letter! The problem is resolved! If we think about it, we will realize that the problem indeed seems to be resolved. However, this solution has given birth to a new problem. How on earth can the intended recipient (B) now open the box, and therefore, the envelope? This solution has not only prevented unauthorized access to the letter, but also the authorized access. That is, even B would not be able to open the lock. This defeats the purpose of sending the letter in this manner, in the first place.

What if A also send the key of the lock along with the box, so that B can open the lock, and get access to the envelope inside the box, and hence, the letter? This seems absurd. If the key travels with the box, anybody who has access to the box in transit (e.g. P) can unlock and open the box.

Therefore, A now comes up with an improved plan. A decides that the locked box should travel to B as discussed (by post/courier/hand-delivery). However, she will not send the key used to lock the box along with the box. Instead, she will decide a place and a time to meet B in person, meet B at that time, and hand over the key personally to B. This will ensure that the key does not land up in the wrong hands, and that only B can access the confidential letter! This now seems to be a full-proof solution! Is it, really?

If A can meet B in person to hand over the key, she can as well hand the confidential letter to B in person! Why have all these additional worries and overheads? Remember that the whole problem started because A and B cannot, for some reason, meet in person!

As a result, we will observe that no solution is completely acceptable. Either it is not full-proof, or is not practically possible. This is the problem of **key distribution** or **key exchange**. Since the sender and the receiver will use the same key to lock and unlock, this is called as *symmetric key operation* (when used in the context of cryptography, this operation is called as **symmetric key cryptography**). Thus, we observe that the key distribution problem is inherently linked with the symmetric key operation.

Let us now imagine that not only A and B but also thousands of people want to send such confidential letters securely to each other. What would happen if they decide to go for symmetric key operation? If we examine this approach more closely, we can see that it has one big drawback if the number of people that want to avail of its services is very large.

We will start with small numbers and then inspect this scheme for a larger number of participants. For instance, let us assume that A now wants to communicate with two persons, B and C securely. Can A use the same kind of lock (i.e. a lock with the same properties, which can be opened with the same key) and key for sealing the box to be sent to B and C? Of course, this is not advisable at all! After all, if A uses the same kind of lock and key to seal the boxes addressed for B and C, what is the guarantee that B does not open the box intended for C, or vice versa (because B and C would also possess the same key as A)? Even if B and C live in the two extreme corners of the city, A cannot simply take such a chance! Therefore, no matter how much secure the lock and key is, A must use a *different* lock-and-key pair for B and C. This means that A must buy two *different* locks and the corresponding two keys (i.e. one key per lock). This is shown in Fig. 2.24. It must also somehow send the respective lock-opening keys to B and C.

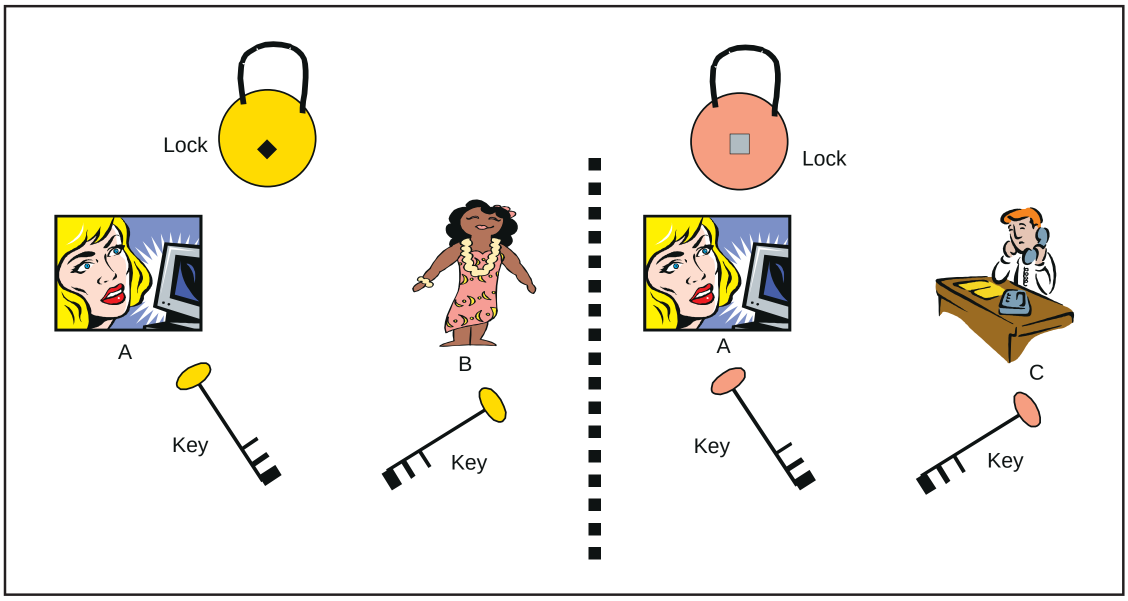


Fig. 2.24 *Use of separate locks and keys per communication pair*

Thus, we have the following situation:

- When A wanted to communicate only with B, we needed one lock-and-key pair (A-B).
- When A wants to communicate with B and C, we need two lock-and-key pairs (A-B and A-C). Thus, we need one lock-and-key pair per person with whom A wants to communicate. If B also wants to communicate with C, we have B-C as the third communicating pair, requiring its own lock-and-key pair. Thus, we would need three lock-and-key pairs to serve the needs of three communicating pairs.
- Let us consider the participation of a fourth person D. Let us also imagine that all of the four persons (A, B, C and D) want to be able to communicate with each other securely. Thus, we have six communicating pairs, namely A-B, A-C, A-D, B-C, B-D and C-D. Thus, we need six lock-and-key pairs, one per communicating pair, to serve the needs of four communicating pairs.
- If E is the fifth person joining this group, we have ten communicating pairs, namely A-B, A-C, A-D, A-E, B-C, B-D, B-E, C-D, C-E and D-E. Thus, we would need ten lock-and-key pairs to make secure communication between all these pairs possible.

Let us now tabulate these results as shown in Table 2.2 to see if any pattern emerges.

Table 2.2 *Number of parties and correspondingly number of lock-and-key pairs required*

Parties involved	*Number of lock-and-key pairs required*
2 (A, B)	1 (A-B)
3 (A, B, C)	3 (A-B, A-C, B-C)
4 (A, B, C, D)	6 (A-B, A-C, A-D, B-C, B-D, C-D)
5 (A, B, C, D, E)	10 (A-B, A-C, A-D, A-E, B-C, B-D, B-E, C-D, C-E, D-E)

We can see that:

- If the number of parties is 2, we need 2 * (2 – 1)/2 = 2 * (1)/2 = 1 lock-and-key pair.
- If the number of parties is 3, we need 3 * (3 – 1)/2 = 3 * (2)/2 = 3 lock-and-key pairs.
- If the number of parties is 4, we need 4 * (4 – 1)/2 = 4 * (3)/2 = 6 lock-and-key pairs.
- If the number of parties is 5, we need 5 * (5 – 1)/2 = 5 * (4)/2 = 10 lock-and-key pairs.

Therefore, can we see that, in general, for *n* persons, the number of lock-and-key pairs is *n* * (*n* – 1)/2! Now, if we have about 1,000 persons in this scheme, we will have 1000 * (1000 – 1)/2 = 1000 * (999)/2 = 99,9000/2 = 499,500 lock-and-key pairs!

Moreover, we must keep in mind that a record of which lock-and-key pair was issued to which communicating pair must be maintained by somebody. Let us call this somebody as T. This is required because it is quite possible that some persons might lose the lock or key, or both. In such cases, T must ensure that the proper duplicate key is issued, or that the lock is replaced with an exact replica of the lock, or that a different lock and key pair is issued (for security reasons), depending on the situation. This is quite a bit of task! Also, who is T, after all? T must be highly trustworthy and accessible to everybody. This is because each communicating pair has to approach T to obtain the lock-and-key pair. This is quite a tedious and time-consuming process!

2.6.2 Diffie-Hellman Key Exchange/Agreement Algorithm

1. *Introduction*

Whitefield Diffie and Martin Hellman devised an amazing solution to the problem of key agreement, or key exchange in 1976. This solution is called as the **Diffie-Hellman Key Exchange/Agreement Algorithm**. The beauty of this scheme is that the two parties, who want to communicate securely, can agree on a symmetric key using this technique. This key can then be used for encryption/decryption. However, we must note that Diffie-Hellman key exchange algorithm can be used only for key agreement, but not for encryption or decryption of messages. Once both the parties agree on the key to be used, they need to use other symmetric key encryption algorithms (we shall discuss some of those subsequently) for actual encryption or decryption of messages.

Although the Diffie-Hellman key exchange algorithm is based on mathematical principles, it is quite simple to understand. We shall first describe the steps in the algorithm, then illustrate its use with a simple example, and then discuss the mathematical basis for it.

2. *Description of the algorithm*

Let us assume that Alice and Bob want to agree upon a key to be used for encrypting/decrypting messages that would be exchanged between them. Then, the Diffie-Hellman key exchange algorithm works as shown in Fig. 2.25.

1. Firstly, Alice and Bob agree on two large prime numbers, n and g. These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.
2. Alice chooses another large random number x, and calculates A such that:
   $A = g^x \bmod n$
3. Alice sends the number A to Bob.
4. Bob independently chooses another large random integer y and calculates B such that:
   $B = g^y \bmod n$
5. Bob sends the number B to Alice.
6. A now computes the secret key K1 as follows:
   $K1 = B^x \bmod n$
7. B now computes the secret key K2 as follows:
   $K2 = A^y \bmod n$

Fig. 2.25 *Diffie-Hellman key exchange algorithm*

It might come as a surprise, but K1 is actually equal to K2! This means that K1 = K2 = K is the symmetric key, which Alice and Bob must keep secret and can henceforth use for encrypting/decrypting their messages with. The mathematics behind this is quite interesting. We shall first prove it, and then examine it.

3. *Example of the algorithm*

Let us take a small example to prove that the Diffie-Hellman works in practical situations. Of course, we shall use very small values for ease of understanding. In real life, these values are very large. The process of key agreement is shown in Fig. 2.26.

Having taken a look at the actual proof of Diffie-Hellman key exchange algorithm, let us now think about the mathematical theory behind it.

4. *Mathematical theory behind the algorithm*

Let us first take a look at the technical (and quite complicated) description of the complexity of the algorithm:

Note Diffie-Hellman key exchange algorithm gets it security from the difficulty of calculating discrete logarithms in a finite field, as compared with the ease of calculating exponentiation in the same field.

Let us try to understand what this actually means, in simple terms.

(a) Firstly, take a look at what Alice does in step 6. Here, Alice computes:
$K1 = B^x \bmod n$.
What is B? From step 4, we have:
$B = g^y \bmod n$.
Therefore, if we substitute this value of B in step 6, we will have the following equation:
$K1 = (g^y)^x \bmod n = g^{yx} \bmod n$.

(b) Now, take a look at what Bob does in step 7. Here, Bob computes:
$K2 = A^y \bmod n$.

1. Firstly, Alice and Bob agree on two large prime numbers, n and g. These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.

   Let n = 11, g = 7.

2. Alice chooses another large random number x, and calculates A such that:
   $A = g^x \bmod n$

   Let x = 3. Then, we have, $A = 7^3 \bmod 11 = 343 \bmod 11 = 2$.

3. Alice sends the number A to Bob.

   Alice sends 2 to Bob.

4. Bob independently chooses another large random integer y and calculates B such that:
   $B = g^y \bmod n$

   Let y = 6. Then, we have, $B = 7^6 \bmod 11 = 117649 \bmod 11 = 4$.

5. Bob sends the number B to Alice.

   Bob sends 4 to Alice.

6. A now computes the secret key K1 as follows:
   $K1 = B^x \bmod n$

   We have, $K1 = 4^3 \bmod 11 = 64 \bmod 11 = 9$.

7. B now computes the secret key K2 as follows:
   $K2 = A^y \bmod n$

   We have, $K2 = 2^6 \bmod 11 = 64 \bmod 11 = 9$.

Fig. 2.26 *Example of Diffie-Hellman key exchange*

What is A? From step 2, we have:
$A = g^x \bmod n$.
Therefore, if we substitute this value of A in step 7, we will have the following equation:
$K2 = (g^x)^y \bmod n = g^{xy} \bmod n$.
Now, basic mathematics says that:
$K^{yx} = K^{xy}$
Therefore, in this case, we have: K1 = K2 = K. Hence the proof.

An obvious question now is, if Alice and Bob can both calculate K independently, so can an attacker! What prevents this? The fact is, Alice and Bob exchange n, g, A and B. Based on these values, x (a value known only to Alice) and y (a value known only to Bob) cannot be calculated easily. Mathematically, the calculations do find out x and y are extremely complicated, if they are sufficiently large numbers. Consequently, an attacker cannot calculate x and y, and therefore, cannot derive K.

5. *Problems with the algorithm*

Can we now consider that the Diffie-Hellman key exchange algorithm solve all our problems associated with key exchange? Unfortunately, not quite!

Diffie-Hellman key exchange algorithm can fall pray to the **man-in-the-middle attack** (or to be politically correct, *woman-in-the-middle attack*), also called as **bucket brigade attack**. The way this happens is as follows.

1. Alice wants to communicate with Bob securely, and therefore, she first wants to do a Diffie-Hellman key exchange with him. For this purpose, she sends the values of n and g to Bob, as usual. Let n = 11 and g = 7. (As usual, these values will form the basis of Alice's A and Bob's B, which will be used to calculate the symmetric key K1 = K2 = K.)
2. Alice does not realize that the attacker Tom is listening quietly to the conversation between her and Bob. Tom simply picks up the values of n and g, and also forwards them to Bob as they originally were (i.e. n = 11 and g = 7). This is shown in Fig. 2.27.

Alice	Tom	Bob
n = 11, g = 7	n = 11, g = 7	n = 11, g = 7

Fig. 2.27 *Man-in-the-middle attack—Part I*

3. Now, let us assume that Alice, Tom and Bob select random numbers x and y as shown in Fig. 2.28.

Alice	Tom	Bob
x = 3	x = 8, y = 6	y = 9

Fig. 2.28 *Man-in-the-middle attack—Part II*

4. One question at this stage could be: why does Tom selects both x and y? We shall answer that shortly. Now, based on these values, all the three persons calculate the values of A and B as shown in Fig. 2.29. Note that Alice and Bob calculate only A and B, respectively. However, Tom calculates both A and B. We shall revisit this shortly.

Alice	Tom	Bob
A $= g^x \bmod n$	A $= g^x \bmod n$	B $= g^y \bmod n$
$= 7^3 \bmod 11$	$= 7^8 \bmod 11$	$= 7^9 \bmod 11$
$= 343 \bmod 11$	$= 5764801 \bmod 11$	$= 40353607 \bmod 11$
$= 2$	$= 9$	$= 8$
	B $= g^y \bmod n$	
	$= 7^6 \bmod 11$	
	$= 117649 \bmod 11$	
	$= 4$	

Fig. 2.29 *Man-in-the-middle attack—Part III*

5. Now, the real drama begins, as shown in Fig. 2.30.

   As shown in the figure, the following things happen:

   (a) Alice sends her A (i.e. 2) to Bob. Tom intercepts it, and instead, sends his A (i.e. 9) to Bob. Bob has no idea that Tom had hijacked Alice's A and has instead given his A to Bob.

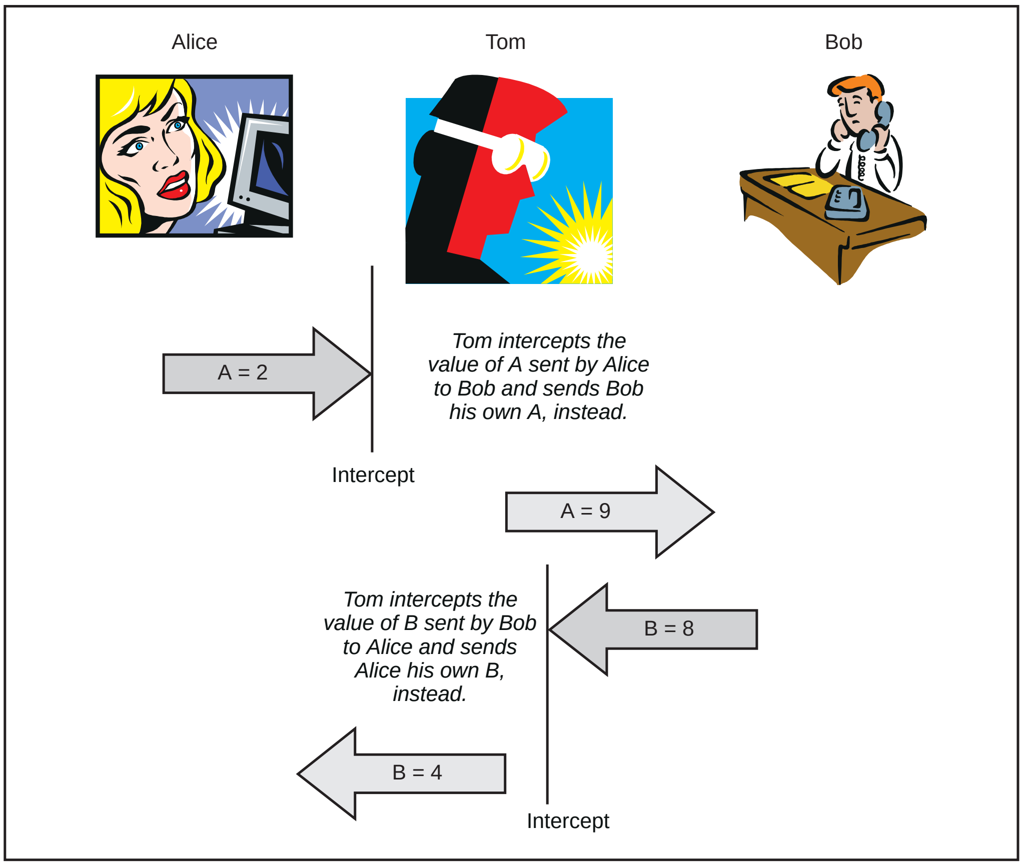

Alice
Tom
Bob
A = 2
Tom intercepts the value of A sent by Alice to Bob and sends Bob his own A, instead.
Intercept
A = 9
Tom intercepts the value of B sent by Bob to Alice and sends Alice his own B, instead.
B = 8
B = 4
Intercept

Fig. 2.30 *Man-in-the-middle attack—Part IV*

(b) In return, Bob sends his B (i.e. 8) to Alice. As before, Tom intercepts it, and instead, sends his B (i.e. 4) to Alice. Alice thinks that this B came from Bob. She has no idea that Tom had intercepted the transmission from Bob, and changed B.

(c) Therefore, at this juncture, Alice, Tom and Bob have the values of A and B as shown in Fig. 2.31.

Alice	Tom	Bob
A = 2, B = 4*	A = 2, B = 8	A = 9*, B = 8

(Note: * indicates that these are the values after Tom hijacked and changed them.)

Fig. 2.31 *Man-in-the-middle attack—Part V*

6. Based on these values, all the three persons now calculate their keys as shown in Fig. 2.32. We will notice that Alice calculates only K1, Bob calculates only K2, whereas Tom calculates both K1 and K2. Why does Tom need to do this? We shall discuss that soon.

Alice	Tom	Bob
K1 = B^x mod n	K1 = B^x mod n	K2 = A^y mod n
= 4^3 mod 11	= 8^8 mod 11	= 9^9 mod 11
= 64 mod 11	= 16777216 mod 11	= 387420489 mod 11
= 9	= 5	= 5
	K2 = A^y mod n	
	= 2^6 mod 11	
	= 64 mod 11	
	= 9	

Fig. 2.32 *Man-in-the-middle attack—Part VI*

Let us now revisit the question as to why Tom needs two keys. This is because at one side, Tom wants to communicate with Alice securely using a shared symmetric key (9), and on the other hand, he wants to communicate with Bob securely using a *different* shared symmetric key (5). Only then can he receive messages from Alice, view/manipulate them and forward them to Bob, and vice versa. Unfortunately for Alice and Bob, both will (incorrectly) believe that they are directly communicating with each other. That is, Alice will feel that the key 9 is shared between her and Bob, whereas Bob will feel that the key 5 is shared between him and Alice. Actually, what is happening is, Tom is sharing the key 9 with Alice and 5 with Bob!

This is also the reason why Tom needed both sets of the secret variables x and y, as well as later on, the non-secret variables A and B.

As we can see, the *man-in-the-middle attack* can work against the Diffie-Hellman key exchange algorithm, causing it to fail. This is plainly because the *man-in-the-middle* makes the actual communicators believe that they are talking to each other, whereas they are actually talking to the man-in-the-middle, who is talking to each of them!

However, not everything is lost. If we think deeply and try to come up with the scheme that will still work, an alternatives emerges: namely an **asymmetric key operation**.

2.6.3 Asymmetric Key Operation

In this scheme, A and B do not have to jointly approach T for a lock-and-key pair. Instead, B alone approaches T, obtains a lock and a key (K1) that can seal the lock, and sends the lock and key K1 to A. B tells A that A can use that lock and key to seal the box before sending the sealed box to B. How can B open the lock, then?

An interesting property of this scheme is that B possesses a *different but related* key (K2), which is obtained by B from T along with the lock and key K1, only which can open the lock. It is guaranteed that no other key, and of course, including the one used by A (i.e. K1) for locking, can open the lock. Since one key (K1) is used for locking, and *another, different* key (K2) is used for unlocking; we will call this scheme as *asymmetric key operation*. Also, T is clearly defined here as a **trusted third party**. T is certified as a highly trustworthy and efficient agency by the government.

This means that B possesses a **key pair** (i.e. two keys K1 and K2). One key (i.e. K1) can be used for locking, and only the corresponding other key (i.e. K2) from the key pair can be used for unlocking. Thus B can send the lock and key K1 to anybody (e.g. A) who wants to send anything securely to B. B would request the sender (e.g. A) to use that lock and key

K1 to seal the contents. B can then open the seal using the key K2. Since the key K1 is meant for locking, and is available to the general public, we shall call K1 as **public key**. Note that K1 need not be secret—in fact, it *should not be secret*! Thus, unlike what happens in the case of symmetric key operation, the (locking) key need not be guarded secretly now. The other key K2 is meant for unlocking, and is strictly held secret/private by A. Therefore, we shall call it as **private key** or **secret key**.

This is shown in Fig. 2.33.

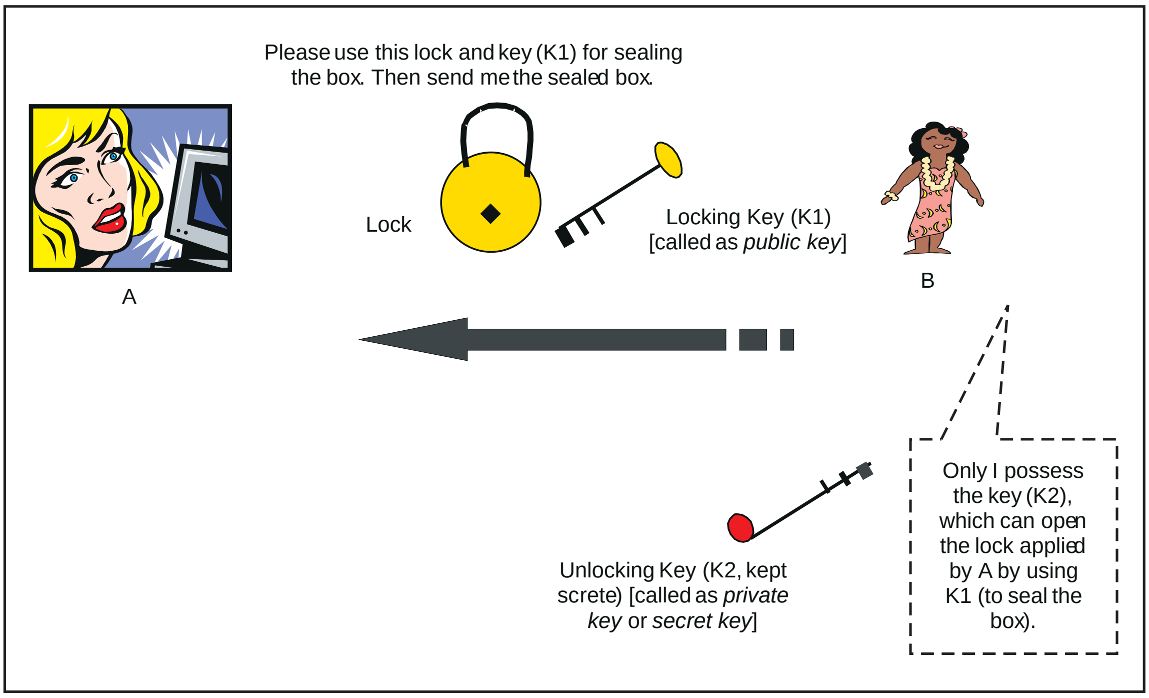

Please use this lock and key (K1) for sealing the box. Then send me the sealed box.
A
Lock
Locking Key (K1) [called as *public key*]
B
Only I possess the key (K2), which can open the lock applied by A by using K1 (to seal the box).
Unlocking Key (K2, kept screte) [called as *private key* or *secret key*]

Fig. 2.33 *Use of key pair*

Note that if B wants to receive something securely from another person say C, B need *not* obtain a fresh lock-and-key pair. B can send *the same* lock-and-key (K1) pair (or a copy of the lock and B's *public key* K1, in case A and C want to send something securely to B at the same time) to C. Thus, C will also use the same lock and B's *public key* K1 to seal the contents before sending them to B. As before, B will use the corresponding *private key* K2 to open the lock. Extending this concept a step further, if B wants to receive messages securely from 10,000 different persons, B can send the same lock-and-public key K1 (or their copies) to each one of them! It need not have 10,000 unique locks and keys (unlike the symmetric key approach)! Always, B's public key K1 will be used by the sender for locking, and B's private key K2 will be used for unlocking by the receiver (i.e. B).

Clearly, this is an extremely convenient approach, as compared to symmetric key operation!

Let us now consider what happens if three persons A, B and C want to communicate with each other. That is, A, B and C must all be able to send/receive messages securely to/from each other. For this to be possible, all the three persons can obtain a lock-and-public key pair from the trusted third party (T). Whenever any one of them wants to receive a message securely from another person, she has to send her lock-and-public key to the sender. That is, when A wants to receive a message securely from B, A sends her lock and public key to B. B can use that to seal the message and send the sealed message to A. A can then use her private key to open the lock. Similarly, when B wants to receive a message securely from A, B sends her lock and public key to A, using which B can secure the message. Since only B has her own private key, she can open the lock and access the message.

Extending this basic idea, if 1,000 people want to be able to securely communicate with each other, only 1,000 locks, 1,000 public keys and the corresponding 1,000 private keys are required. This is in stark contrast to the symmetric key operation wherein for 1,000 participants, we needed 499,500 lock-and-key pairs (please refer to our earlier discussion).

Therefore, in general, when using asymmetric key operation, the recipient has to send the lock and her public key to the sender. The sender uses these to apply the lock and sends the sealed contents to the recipient. The recipient uses her private key to open the lock. Since only the recipient possesses the private key, all concerned are assured that only the intended recipient can open the lock.

2.7 STEGANOGRAPHY

Note Steganography is a technique that facilitates hiding of a message that is to be kept secret inside other messages.

This results in the concealment of the secret message itself! Historically, the sender used methods such as invisible ink, tiny pin punctures on specific characters, minute variations between handwritten characters, pencil marks on handwritten characters, etc.

Of late, people hide secret messages within graphic images. For instance, suppose that we have a secret message to send. We can take another image file and we can replace the last two rightmost bits of each byte of that image with (the next) two bits of our secret message. The resulting image would not look too different, and yet carry a secret message inside! The receiver would perform the opposite trick: it would read the last two bits of each byte of the image file, and reconstruct the secret message. This concept is illustrated in Fig. 2.34.

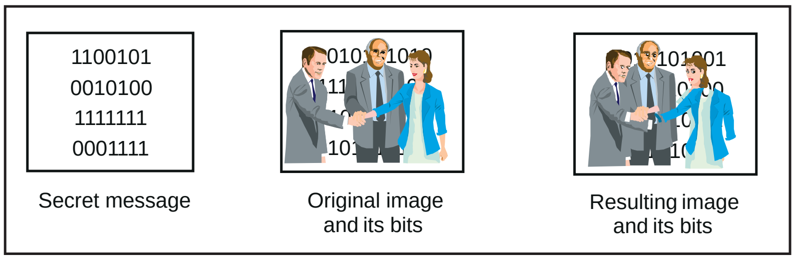


Fig. 2.34 *Steganography example*

2.8 KEY RANGE AND KEY SIZE

We have already seen one way to classify attacks on information itself, or information sources (such as computers or networks). However, the encrypted messages can be attacked, too! Here, the cryptanalyst is armed with the following information:

- The encryption/decryption algorithm
- The encrypted message
- Knowledge about the key size (e.g. the value of the key is a number between 0 and 100 billion)

As we have seen previously, the encryption/decryption algorithm is usually not a secret—everybody knows about it. Also, one can access an encrypted message by various means (such as by listening to the flow of information over a network). Thus, only the actual value of the key remains a challenge for the attacker. If the key is found, the attacker can resolve the mystery by working backwards to the original plain text message, as shown in Fig. 2.35. We shall consider the brute force attack here, which works on the principle of trying every possible key in the **key range**, until you get the right key.

It usually takes a very small amount of time to try a key. The attacker can write a computer program that tries many such keys in one second. In the best case, the attacker finds the right key in the first attempt itself, and in the worst case, it is the 100 billionth attempt. However, the usual observation is that the key is found somewhere in between the possible range. Mathematics tells us that on an average, the key can be found after about half of the possible values in the key range are checked. Of course, this is just a guideline, and may or may not work in real practice for a given situation.

As the figure shows, the attacker has access to the cipher text block and the encryption/decryption algorithm. She also knows the key range (a number between 0 and 100 billion). She now starts trying every possible key, starting from 0. After every decryption, she looks at the generated *plain text* (actually, it is not truly plain text, but decrypted text, but we shall ignore this technical detail). If she notices that the decryption has yielded unintelligent plain text, she continues the process with the next key in the sequence. Finally, she is able to find the right key with a value 90,171,451,191, which yields the plain text *To: Payroll.*

How does the attacker determine if the plain text, and therefore the key, are the right ones? This can be determined depending on the value of the plain text. If the plain text seems reasonable (i.e. very close to actual English words/sentences/numbers that make sense), it is highly probable that the plain text is indeed what corresponds to the cipher text.

This means that our key, and therefore, the plain text message, are now cracked! How can we prevent an attacker from succeeding in such attempts? As we know, currently, our key range is 0 to 100 billion. Also, let us assume that the attacker took only 5 minutes to successfully crack our key. However, we want our message to remain secret for at least 5 years. This means that the attacker must spend at least 5 years in trying out every possible key, in order to obtain our original plain text message. Therefore, the solution to our problem lies in expanding or increasing the *key range* to a size, which requires the attacker to work for more than 5 years in order to crack the key. Perhaps our key range should be from 0 to 100 billion billion billion billion.

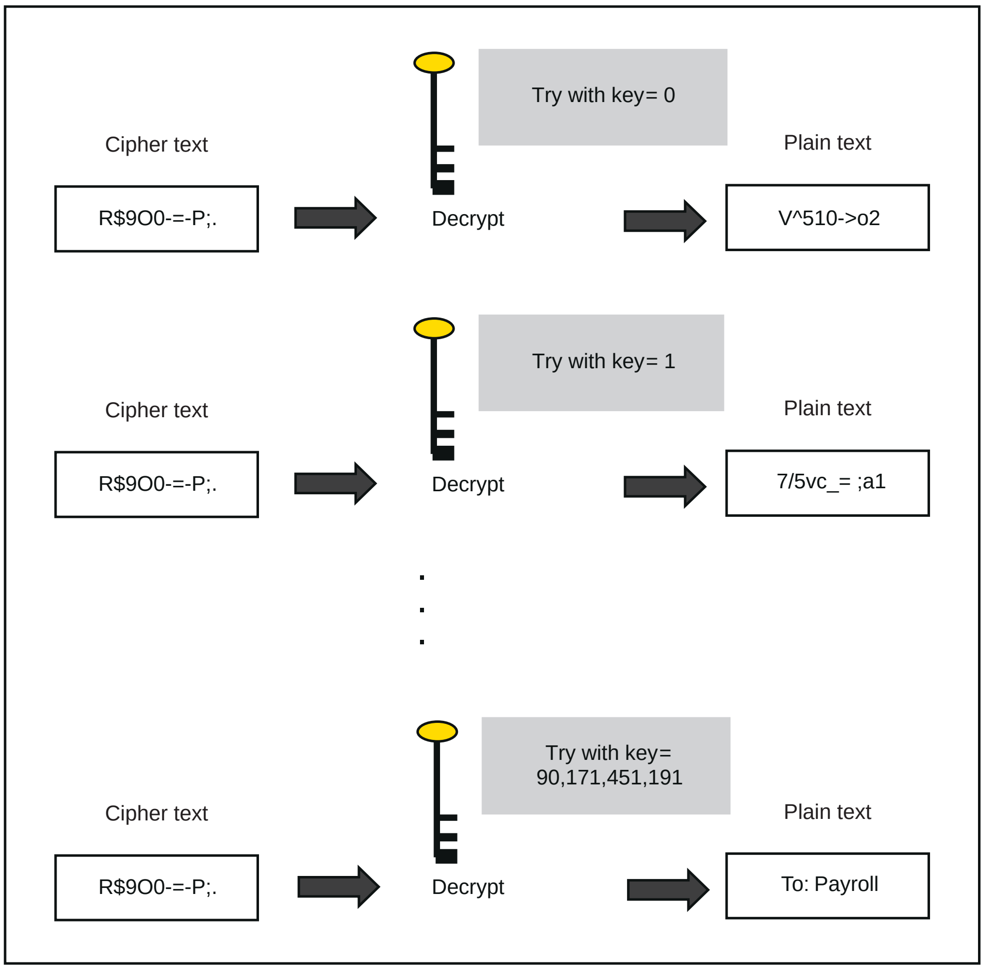


Fig. 2.35 *Brute force attack*

In computer terms, the concept of *key range* leads us to the principle of **key size**. Just as we measure the value of stocks with a given index, gold in troy ounces, money in dollars, pounds or rupees, we measure the strength of a cryptographic key with *key size*. We measure key size in bits, and represent is using the binary number system. Thus, our key might be of size 40 bits, 56 bits, 128 bits, and so on. In order to protect ourselves against a brute force attack, the key size should be such that the attacker cannot crack it within a specified amount of time. How long it should be? Let us study this.

At the simplest level, the key size can be just 1 bit. This means that the key can be either 0 or 1. If the key size is 2, the possible key values are 00, 01, 10, 11. Obviously, these examples are merely to understand the theory, and have no practical significance.

From a practical viewpoint, a 40-bit key takes about 3 hours to crack. However, a 41-bit key would take 6 hours, a 42-bit key takes 12 hours, and so on. This means that every additional bit doubles the amount of time required to crack the key. Why is this so? This works on the simple theory of binary numbers wherein every additional bit doubles the number of possible states of the number. This is shown in Fig. 2.36.

A 2-bit binary number has four possible states:

00
01
10
11

If we have one more bit to make it a 3-bit binary number, the number of possible states also doubles to eight, as follows:

000
001
010
011
100
101
110
111

In general, if an *n bit* binary number has *k* possible states, an *n* + 1 *bit* binary number will have 2*k* possible states.

Fig. 2.36 *Understanding key range*

Thus, with every incremental bit, the attacker has to perform double the number of operations as compared to the previous key size. It is found that for a 56-bit key, it takes 1 second to search 1 per cent of the key range. Taking this argument further, it takes about 1 minute to search about half of the key range (which is what is required, on an average, to crack a key). Using this as the basis, let us have a look at the similar values (time required for a search of 1 per cent and 50 per cent of the key space) for various key sizes. This is shown in Table 2.3.

Table 2.3. *Efforts required to break a key*

Key size in bits	*Time required to search 1 per cent of the key space*	*Time required to search 50 per cent of the key space*
56	1 second	1 minute
57	2 seconds	2 minutes
58	4 seconds	4 minutes
64	4.2 minutes	4.2 hours
72	17.9 hours	44.8 days
80	190.9 days	31.4 years
90	535 years	321 centuries
128	146 billion millennia	8 trillion millennia

We can represent the possible values in the key range using hexadecimal notation, and see visually how an increase in the key size increases the key range, and therefore, the complexity for an attacker. This is shown in Fig. 2.37.

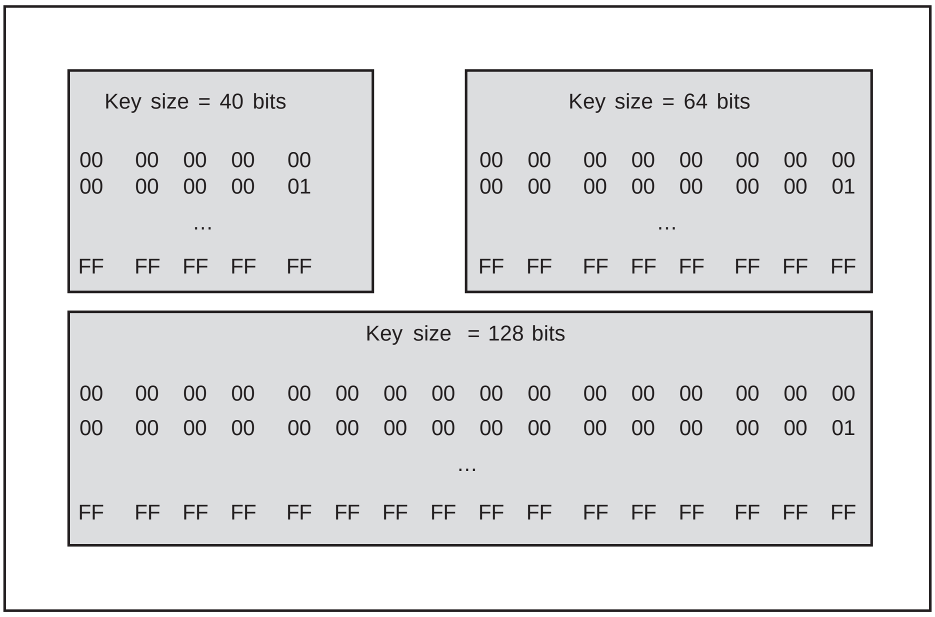


Fig. 2.37 *Key sizes and ranges*

Clearly, we can assume with reasonable confidence that a 128-bit key is quite safe (because 2^{128} means about 340,000,000,000,000,000,000,000,000,000,000,000,000 possible keys!), considering the capabilities of computers today. Obviously, as computing power and techniques improve, these numbers change. May be, in a few years time, a 128-bit key will be cracked. Then, we would need to rely on keys of size at least 256 bits, or 512 bits.

One may think that with the technological progress chasing key sizes so fast, how long can this go on and on? Today, 56-bit keys are not safe, tomorrow, 128-bit keys may not be sufficient, another day, 256-bit keys can be cracked, and so on! How far can we go? Well, it is argued that we may not have to look beyond 512-bit keys at any point of time in the future. That is, 512-bit keys will always be safe. What substantiates this argument?

Suppose that every atom in the universe is actually a computer. Then, we would have 2^{300} such computers in the world. Now, if each of these computers could check 2^{300} keys in one second, it would take 2^{162} millennia to search 1 per cent of a 512-bit key! The Big Bang theory suggests that the amount of time that has passed since the universe came into existence is less than 2^{24} millennia, which is significantly less than the key search time. Thus, 512-bit keys will always be safe. (Although, we would not bet on this!)

2.9 POSSIBLE TYPES OF ATTACKS

Based on the discussion so far, when the sender of a message encrypts a plain text message into its corresponding cipher text, there are three possibilities for an attack on this message, as follows:

- **Cipher text only attack:** In this type of attack, the attacker does not have any clue about the plain text. She has some or all of the cipher text. Based on the frequency of letters (e.g. the alphabets e, i, a are very common in English, etc.) the attacker makes an attempt to guess the plain text.

- **Known plain text attack:** In this case, the attacker knows about some pairs of plain text and corresponding cipher text. Using this information, the attacker tries to find other pairs, and therefore, know more and more of the plain text.
- **Chosen plain text attack:** Here, the attacker selects a plain text block, and tries to look for the encryption of the same in the cipher text.

Chapter Summary

- *Normal language of communication uses plain text or clear text.*
- *A codified message produces cipher text.*
- *Cipher text cannot be understood by anyone who does not know about the coding scheme used for producing the cipher text.*
- *Plain text can be transformed into cipher text using substitution or transposition techniques.*
- *In substitution cipher, the characters of plain text are replaced by other characters or symbols.*
- *Caesar Cipher is the world's first well known substitution cipher. It replaces each character in the plain text message with a character that is three places down the line.*
- *Modified versions of Caesar Cipher are available.*
- *An attack on cipher text wherein the attacker tries all possibilities is called as Brute-force attack.*
- *An attacker is called as cryptanalyst and the attack itself is called as cryptanalysis.*
- *Mono-alphabetic Cipher is a modified version of Caesar Cipher, and is much harder to crack that Caesar Cipher.*
- *Homophonic Substitution Cipher is very similar to Mono-alphabetic Cipher. However, it adds more complexity to it.*
- *In Polygram Substitution Cipher, one block of text is replaced by another block.*
- *Transposition techniques involve permutations and combinations over the plain text to produce cipher text.*
- *In Rail fence Technique, the plain text contents are written as sequence of diagonals, and are read as sequence of rows.*
- *In Simple Columnar Transposition Technique, the plain text contents are written as rows and read as columns, but not necessarily in the same order. It can also have multiple rounds.*
- *Vernam Cipher (also called as One-time pad) uses random cipher text every time.*
- *Book Cipher or Running Key Cipher uses some text from a book to produce cipher text.*
- *The process of encoding plain text messages into cipher text is called as encryption.*
- *The process of decoding cipher text messages back into plain text is called as decryption.*
- *Cryptography can be based on a single key (Symmetric) or two keys (Asymmetric).*

Key Terms and Concepts

- Asymmetric Key Cryptography
- Brute-force attack
- Caesar Cipher
- Clear text
- Cryptanalyst
- Decryption
- Encryption
- Homophonic Substitution Cipher
- Man-in-the-middle attack
- One-Time Pad
- Polygram Substitution Cipher
- Running Key Cipher
- Simple Columnar Transposition Technique with multiple rounds
- Transposition Cipher
- Book Cipher
- Bucket brigade attack
- Cipher text
- Cryptanalysis
- Cryptography
- Decryption algorithm
- Encryption algorithm
- Key
- Mono-alphabetic Cipher
- Plain text
- Rail Fence Technique
- Simple Columnar Transposition Technique
- Substitution Cipher
- Symmetric Key Cryptography
- Vernam Cipher

Multiple-choice Questions

1. The language that we commonly use can be termed as __________.
   (a) pure text (b) simple text
   (c) plain text (d) normal text
2. The codified language can be termed as __________.
   (a) clear text (b) unclear text
   (c) code text (d) cipher text
3. In substitution cipher, the following happens.
   (a) characters are replaced by other characters
   (b) rows are replaced by columns
   (c) columns are replaced by rows
   (d) none of the above
4. Transposition cipher involves __________.
   (a) replacement of blocks of text with other blocks
   (b) replacement of characters of text with other characters
   (c) strictly row-to-column replacement
   (d) some permutation on the input text to produce cipher text
5. Caesar Cipher is an example of __________.
   (a) Substitution Cipher
   (b) Transposition Cipher
   (c) Substitution as well as Transposition Cipher
   (d) none of the above

6. Vernam Cipher is an example of __________.
   (a) Substitution Cipher
   (b) Transposition Cipher
   (c) Substitution as well as Transposition Cipher
   (d) none of the above
7. Cryptanalyst is a person who __________.
   (a) devises cryptography solutions
   (b) attempts to break cryptography solutions
   (c) none of these
   (d) both of these
8. Homophonic Substitution Cipher is __________ to break as compared to Mono-alphabetic Cipher.
   (a) easier (b) the same
   (c) difficult (d) easier or same
9. The process of writing the text as diagonals and reading it as sequence of rows is called as __________.
   (a) Rail Fence Technique (b) Caesar Cipher
   (c) Mono-alphabetic Cipher (d) Homophonic Substitution Cipher
10. The mechanism of writing text as rows and reading as columns is called as __________.
   (a) Vernam Cipher
   (b) Caesar Cipher
   (c) Simple Columnar Transposition Technique
   (d) Homophonic Substitution Cipher
11. Vernam Cipher is also called as __________.
   (a) Rail Fence Technique (b) One-time pad
   (c) Book Cipher (d) Running Key Cipher
12. Book Cipher is also called as __________.
   (a) Rail Fence Technique (b) One-time pad
   (c) Mono-alphabetic Cipher (d) Running Key Cipher
13. Conversion of plain text into cipher text is called as __________.
   (a) encryption (b) decryption
   (c) cryptography (d) cryptanalyst
14. Conversion of cipher text into plain text is called as __________.
   (a) encryption (b) decryption
   (c) cryptography (d) cryptanalyst

Review Questions

1. What is plain text? What is cipher text? Give an example of transformation of plain text into cipher text.
2. What are the two basic ways of transforming plain text into cipher text?
3. What is the difference between Substitution Cipher and Transposition Cipher?
4. Discuss the concept of Caesar Cipher.
5. What is the output of plain text Hello there, my name is Atul if we use Caesar Cipher to encode it?

6. How can Caesar Cipher be cracked?
7. What is Mono-alphabetic Cipher? How is it different from Caesar Cipher?
8. Why is Mono-alphabetic Cipher difficult to crack?
9. Discuss Homophonic Substitution Cipher with reference to Mono-alphabetic Cipher.
10. What is the main feature of Polygram Substitution Cipher?
11. Discuss the algorithm for Rail Fence Technique.
12. Assume a plain text *Security is important*, and generate the corresponding cipher text using Rail Fence Technique.
13. How does Simple Columnar Transposition technique work? Assume the same plain text (*Security is important*) and generate the corresponding cipher text using this technique.
14. What is the modified version of Simple Columnar Transposition technique?
15. What is the principle behind One-time pads? Why are they highly secure?
16. How is Book Cipher different from One-time pad?
17. What is encryption? What is decryption? Draw a block diagram showing plain text, cipher text, encryption and decryption.
18. Distinguish between Symmetric and Asymmetric Key Cryptography.

Design/Programming Exercises

1. Write a Java program to perform encryption and decryption using the following algorithms:
   - Caesar Cipher
   - Rail Fence Technique
   - Simple Transposition Technique
2. Alice meets Bob and says *Rjjy rj ts ymj xfggfym. bj bnqq inxhzxx ymj uqfs.* If she is using Caesar Cipher, what does she want to convey?
3. What would be the transformation of a message 'Happy birth day to you' using Rail Fence technique?
4. The following message was received by Bob: hs *yis ls. eftstof n ^ TyymrieraseMr e ho ec ^ etose Dole ^* . If the message is encrypted by using the Simple Transposition method, with the key as 24153, find the original plain text.
5. During the World War II, a German spy used a technique known as *Null Cipher*. Using this technique, the actual message is created from the first alphabet of each word in the message that is actually transmitted. Find out the hidden secret message if the transmitted message was *President's embargo ruling should have immediate notice. Grave situation affecting international law, statement foreshadows ruin of many neutrals. Yellow journals unifying national excitement immensely.*
6. Consider a scheme involving the replacement of alphabets as follows:

Original	A	B	C	…	X	Y	Z
Changed to	Z	Y	X	…	C	B	A

   If Alice sends a message *HSLDNVGSVNLMVB*, what should Bob infer from this?
7. Diffie-Hellman Exercises:
   (a) Alice and Bob want to establish a secret key using the Diffie-Hellman Key Exchange protocol. Assuming the values as n = 11, g = 5, x = 2 and y = 3, find out the values of A, B and the secret key (K1 or K2).
   (b) Next time, they choose n = 10, g = 3, x = 5 and y = 11. Find out the values of A, B, K1 and K2.

8. Encrypt the following message using Mono-alphabetic Substitution Cipher with key = 4.
   This is a book on Security
9. Decrypt the following message using Mono-alphabetic Substitution Cipher with key = 4.
   wigyvmxc rixiv gsqiw jsv jvii
10. Encrypt the following plain text bit pattern with the supplied key, using the XOR operation, and state the resulting cipher text bit pattern.

Plain text	10011110100101010
Key	01000101111101101

11. Transform the cipher text generated in the above exercise back to the original plain text.
12. Consider a plain text message I AM A HACKER. Encrypt it with the help of the following algorithm:
    (a) Replace each alphabet with its equivalent 7-bit ASCII code.
    (b) Add a 0 bit as the leftmost bit to make each of the above bit patterns 8 positions long.
    (c) Swap the first four bits with the last four bits for each alphabet.
    (d) Write the hexadecimal equivalent of every four bits.
13. Write a C program to perform the task of the above exercise.