Chapter 1

A Primer on Crypto Basics

In This Chapter

- It's not just for spies anymore
- Basic information on early cryptography
- Cipher, cipher, who's got the cipher?
- It's all hashed up
- What are cryptosystems?
- Some everyday uses
- Elitist attitudes towards cryptography

Computers and use of the Internet have fostered new interest in cryptography partly due to the new emphasis on personal privacy. Little did I realize that in our efforts to make it easy for computers to share stuff, it would make it easy for other people to see all of our personal stuff, too. Perhaps you've discovered for yourself that it is far too easy for unknown persons to read your e-mail, private documents, love letters, financial information, and so on. The Internet is truly the Global Village . . . a village where everyone can see what you do and hear what you say. The good news is that you can use cryptography to protect yourself from the eavesdroppers and Peeping Toms of the village.

Not only can cryptography scramble your files, but it can also be used to prove who you are (and maybe who you aren't!). Cryptography can be used to alert you if the contents of a file have been changed, attest to the identity of the person who sent you a message, keep online communications safe and secure, and, of course, hide important data. And the best news of all is that not every cryptographic solution is expensive, and you don't need to be a rocket scientist to incorporate crypto solutions into your network.

It's Not about James Bond

There's no need for fancy gizmos, fast cars, or beautiful women. As nice as those may be (for some!), the world of cryptography can be used on even low-tech systems. Forget the cloak and dagger and put away your raincoat

and fedora — most cryptography is done out in the open now. The special programs and codes used to scramble data are available for all the world to see. In fact, having them out in the open helps make cryptography more secure because more people can test for weaknesses.

Because cryptography is usually associated with spies, secret messages, and clandestine meetings, you might have thought that cryptography stopped being used at the end of the Cold War. Believe it or not, its use is actually on the rise. I think that's partially due to more awareness of personal identity theft and also because more is being written in the media about how data needs more protection that a common PC gives you.

Cryptography is about scrambling data so that it looks like babble to anyone except those who know the trick to decoding it. Almost anything in the world can be hidden from sight and revealed again. The magician David Copperfield has made his living from hiding enormous things from plain view — like elephants and the Statue of Liberty — and then magically revealing them again. Any magician will tell you that in order to make things disappear and appear again, you have to have a plan of action — a formula or recipe — to make the magic work. Although you can't directly equate magic acts with cryptography (although cryptography may seem like magic), there is a similarity between magic and cryptography in that they both need to have a formula in order to work correctly time after time.

Go with the rhythm

In cryptography, the magic recipe for hiding data is called an *algorithm.* An algorithm is a precise set of instructions that tells programs how to scramble and unscramble data. A simple algorithm might read like this:

```
Step 1:Delete all instances of the letter "e" in the data
Step 2:Replace the letter "t" with the number "7"
Step 3:Reverse the order of the data and rewrite it from the
           end to the beginning
```

Now, this is just me playing around with what a simple algorithm *might* look like, just so you can get an idea of what I'm talking about. The steps above are not an actual algorithm; it's my pretend algorithm of the week. Algorithms used in programs today are mathematical functions with the instructions written in programming code.

Here's just a portion of a real algorithm called *DES* (Data Encryption Standard) that was adopted by the government in 1977. DES is a block cipher that transforms 64-bit data blocks under a 56-bit secret key by means of permutation and substitution. (You're not meant to understand that last sentence yet!) So, here is just a tiny, tiny bit of the DES algorithm:

```
Get a 64-bit key from the user. (Every 8th bit is consid-
        ered a parity bit. For a key to have correct
        parity, each byte should contain an odd number of
        "1" bits.)
Calculate the key schedule.
Perform the following permutation on the 64-bit key. (The
        parity bits are discarded, reducing the key to 56
        bits. Bit 1 of the permuted block is bit 57 of
        the original key, bit 2 is bit 49, and so on with
        bit 56 being bit 4 of the original key.)
Permuted Choice 1 (PC-1)
57 49 41 33 25 17  9
1  58 50 42 34 26 18
10 2  59 51 43 35 27
19 11 3  60 52 44 36
63 55 47 39 31 23 15
7  62 54 46 38 30 22
14 6  61 53 45 37 29
21 13 5  28 20 12  4
```

In actuality, the remainder of the DES algorithm could easily fill six or seven pages! What I've shown you is just a small portion of the entire recipe. Interestingly, although DES is complex, it was found to have serious flaws that were exposed in 1998. These flaws lead teams of cryptographers to re-work DES because the original algorithm could be cracked and was no longer considered safe to use. The algorithm the cryptographers came up with to replace DES is called *3DES (Triple DES)*. I' tell you more about 3DES in Chapter 2 about algorithms.

Rockin' the rhythm

The reason that algorithms are so complex is to ensure that they can't be easily broken. It wouldn't do a spy any good to send out a secret message if everyone in the world could crack the code and read it. The algorithms we use today have been tested by crypto experts to check their strength, but sometimes it takes years to find the fatal flaw. When this happens, notices are sent out via vendors and the media to let users know that they may need to make some changes in encryption programs they are using.

Most algorithms are mind-numbingly complex mathematical equations — or at least they appear that way to me! Fortunately, you normally don't have to deal with the algorithm itself — the encryption software does that for you. For that reason, I'm not going to dwell on the math behind the science. Just like you don't need to be a mechanical genius to drive a car, you don't need to be a mathematician to be able to use encryption products. (Hooray!) For most encryption products, the most difficult part is the initial setup. After that, the scrambling and unscrambling is mostly done without your interaction.

There are tons of different algorithms used in the world of cryptography. Why? For the same reason you use different recipes to make a cake. Some recipes are better, some recipes are easier, and some recipes depend on time and care to make them turn out right. The same thing happens with algorithms — we need to use faster, easier, stronger algorithms, and some are better than others at accomplishing the task. It all depends on your needs as to which algorithms you'll eventually use in your system.

There are also tons of arguments as to what makes a good algorithm and what makes a bad algorithm. Get any three crypto geeks in a room to discuss the differences and, chances are, they'll still be arguing a week later. Good algorithms are generally referred to as *strong crypto* and bad algorithms are called *weak crypto*. You'll find arguments galore in newsletters and mail lists that attempt to describe why one algorithm is better than the other. You'll need to know at least the basics on how to tell one from the other, so you'll be seeing information on good versus bad later on in this book. Often the problem has more to do with the installation and setup of the software than problems with the product or the algorithm.

Starting with this chapter, I give you the plain, old-fashioned basics that are good for you to know. This subject is really complex, and humongous tomes have been written by others, but that's not what I'll be doing here. I know you're not trying to get a college degree on the subject — you just want to know enough to buy the right stuff, install it correctly, and be able to use it. If that's what you want, then you've got the right book!

Getting to Know the Basic Terms

I'm going to start you off with some introductory terms. These are not meant to confuse you; rather, they are meant to gradually introduce you to some of the lingo used in cryptography.

- **Encrypt:** Scrambling data to make it unrecognizable
- **Decrypt:** Unscrambling data to its original format
- **Cipher:** Another word for *algorithm*
- **Key:** A complex sequence of alpha-numeric characters, produced by the algorithm, that allows you to scramble and unscramble data
- **Plaintext:** Decrypted or unencrypted data (it doesn't have to be text only)
- **Ciphertext:** Data that has been encrypted

Cryptography through the ages

Making secret messages and then sending them on to someone else to figure out is nothing new. The ancient Greeks used ciphers to send secret messages to their armies in the field. Benedict Arnold used a cipher based on a book called Blackstone's *Commentaries* (a book of essays about the law). In one sense, the Egyptian hieroglyphics can also be considered to be ciphers.

Ciphers really came into their own during WWI and WWII. Entire military and government departments were dedicated to the tasks of coming up with new methods of making secret messages. In addition to making secret messages, these offices also had to figure out how to decrypt the enemy's secret messages. It was from that base of intelligence that modern cryptography has come to be. The government soon discovered that, war or no war, they still had to create secret messages.

I want to mention *keys* at this point because they are all-important to cryptography. A *key* locks and unlocks secret messages — just like a door key locks and unlocks doors. Because keys are central to good cryptography, you can be sure that you'll be learning much more about them in Chapter 4. For now, though, I'm going to keep focused on ciphers and discuss some of the common cipher types.

What Makes a Cipher?

Over the ages there have been as many ways to hide and change data as there have been changes in clothing fashions. Likewise, some of these ciphers have fallen out of fashion while others have become classics.

Generally, ciphers are much simpler forms of algorithms than we use today. Many of these early ciphers were very easy to crack. In today's algorithms, we use the principles of these early ciphers, but much complexity has been added to make them harder to crack. Here, then, are some of the basic ciphers from which our modern cryptography has emerged.

Concealment ciphers

Concealment ciphers have been used for centuries to hide a message in plain sight. They have been used to give orders to troops at war, to tell spies where to meet their contacts, and to even help people like Mary, Queen of Scots, coordinate rendezvous times with her admirers.

The next paragraph is an example of a very old concealment cipher that was given to a prisoner in England during the time of Oliver Cromwell. Hidden within the message are the instructions to the prisoner on how to escape:

> *Worthie Sir John: Hope, that is the best comfort of the afflicated, cannot much, I fear me, help you now. That I would saye to you, is this only: if ever I may be able to requite that I do owe you, stand not upon asking me: Tis not much I can do: but what I can do, bee you verie sure I wille. I knowe that, if deathe comes, if ordinary men fear it, it frights not you, accounting is for a high hounour, to have such a rewarde of your loyalty. Pray yet that you may be spare this soe bitter, cup, I fear not that you will grudge any suffereings; onlie if bie submission you can turn them away, tis the part of a wise man. Tell me, as If you can, I do for you anythinge that you can wolde have done. The general goes back on Wednesday. Restinge your servant to command. R.J.*

I don't know how the key was given to the prisoner so he could decrypt the message, but the key is, "the third letter after every punctuation mark." If you follow that key, you will find that the concealed message is:

"panel at east end of chapel slides"

And, yes, the prisoner did escape! He asked to go to the chapel prior to his execution so he could pray for his soul. The guards left him in the chapel and manned the entrance. When they figured he had had long enough and went in to check on him, surprise! No prisoner! How do you explain that one to the King?

Substitution ciphers

Just as it sounds, a *substitution cipher* substitutes one letter or character for another. As a child you may have gotten a secret decoder ring from an offer on a cereal box or chocolate milk powder. The decoder ring consisted of two dials, both containing all the letters of the alphabet. The trick was to twirl one dial around the other so that the letters of the alphabet did not match up. Then you found the letter you wanted to use on one ring and substituted the letter on the other ring. Carry on letter by letter and then you have a secret message. Although this is technically not a ring shown below, here's an example of how the substitutions would line up:

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

Using the graph above, you would locate your letter and then substitute it with the letter directly below it. Therefore, the phrase:

```
ATTACK AT DAWN AT THE NORTHERN BRIDGE
```

would become

```
SLLSUC SL VSOF SL LZW FGJLZWJF TJAVYW
```

Of course to decrypt your message, your intended recipient would also have to have a decoder ring and he would need to know how far to twirl his dial so it matched yours. This number would indicate the switch in letters — and it is also the key to decrypting the message. In the example above, the switch is 18 letters to the right of the letter *A;* therefore, the key is "18." This cipher is probably one of the best known in the world and is also referred to as the "Caesar Cipher" because of historical references linking Julius Caesar and this type of cipher.

Transposition ciphers

One of the oldest known ciphers is called a *transposition cipher*. This type of cipher changes the order of the letters of the original message. One method is to write the message in a series of columns and rows in a grid — or you could write the message backwards. One of the oldest transposition ciphers is the Spartan *scytale* (also spelled as *skytale)*. This information comes from Plutarch, who was an ancient Greek priest and scholar. Plutarch tells how Lacedaemonian generals exchanged messages by winding a narrow ribbon of parchment spirally around a staff or a spear. The message was then written length-wise across the wound-up parchment. When the parchment was unwound, you could only see parts of words or phrases that were written and the pattern of the words seemed random. This cipher could be read only by the person who had a spear of exactly the same circumference, who could rewind the parchment, so that the letters would reappear in their original order. If the spear used was too thick or too skinny, the words would not match up when the parchment was wound around it. So, in this case the receiver had to be aware of two secrets — or two keys — to read the message.

The German Enigma cipher machine

The most commonly known substitution cipher is the Enigma machine that was used by the Germans in World War II to encrypt their secret military messages. The Enigma machine looked roughly like a typewriter except that it had a number of different rotors, sort of like the odometer on your car. These rotors were placed next to one another on a shaft and then spun to set the shift in letters for substitution. But because there was more than one substitution involved, the messages were even more scrambled than using the single substitution I used in the example above.

The Enigma machine was one of the first usages of a strong cipher. It took the concerted effort of many nations, many minds, and a number of years to finally crack the Enigma code.

He had to know to wind the parchment around a spear of some sort, and he also had to know how thick the pole should be.

While substitution ciphers preserve the order of the letters used in the message, transposition ciphers reorder the letters. Transposition ciphers are rarely used nowadays, but they have been very important in the past. Although there are literally hundreds of different types, I'm going to show you one of the simpler ones. You can do this one yourself — all you need is paper and a pencil.

The encrypted message gets to you looking like this:

```
GRYSO IISAU  VNTFS  EKOEE  EEAHX
```

The key to this cipher is a block grid. If you know how many rows and columns are on the grid, then you can decrypt the message. Looking at the grid below, can you see how the message was created and what the message really says?

G	I	V	E	E
R	I	N	K	E
Y	S	T	O	A
S	A	F	E	H
O	U	S	E	X

Solution: if you read *down* the columns, you see the encrypted message. If you read *across* the rows, you can decrypt the message, which reads: *GIVE ERIN KEYS TO A SAFEHOUSE X*.

You may have noticed that the encrypted message did not match the same spacing as the decrypted message. That's done on purpose to (hopefully) confuse you further. If you don't know where one word starts and another begins, it makes it harder for a casual viewer to make any sense of what's written. In many old encrypted messages you'll note that the messages are in all caps and are written out in groups of five letters. In fact, it has become a standard to type a message in groups of five letters for simple encrypted messages.

Hash without the corned beef

A small departure from the ciphers I've been discussing comes under the heading of hashes. A *hash* is not meant to be decrypted. "What," you say? That's right. A hash is what is referred to as a *one-way function* — you use a hash to encrypt something, but the result is never decrypted.

The purpose of a hash is to create a "fingerprint" of your data. The hash algorithm goes through its permutations, and the result is a bunch of alpha-17 fixed lengths.) The purpose of a hash is to prove the integrity of the data (encrypted or not) that has been sent. When you receive the data, the hash is included at the bottom of the data. You can run the same hash algorithm against the data you receive, and if the data has not been changed en route, you will get the same set of alpha-numeric characters. If your result is not the same, then something happened during the transmission and the data you received was changed from the original.

Many software companies include a hash value with their programs. That way, you can check to see if the software you got matches the value the software vendor sends you. If they don't match, then you need to get another copy of the software. Any software can be hijacked and have Trojans or other malicious programs inserted into them. A hash helps you detect whether or not this is a possibility.

I have much more information on hashes in Chapter 4.

XOR what?

Now I'll probably be slammed by all the brilliant crypto-geeks in the world for putting XOR here because it is not really a cipher. It's actually a mathematical *operation*. I'll justify putting it here because a majority of modern algorithms use the XOR operation during the encryption process.

Yes, the operation is pronounced just like it looks: *Ex-Or.* When I first heard this uttered, I thought there might be something missing from the person's statement. Was he trying to say, "Ex, or else" or "X or Y"? I finally asked "Ex, or what?" and soon discovered that XOR stands for *Exclusive-Or.* Although the name of this operation does sound silly, it's one of those things that you are bound to hear associated with modern cryptography.

When you see an algorithm diagram, you'll see the symbol for XOR, shown here in Figure 1-1.

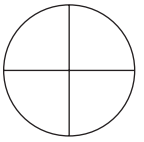

Figure 1-1: The symbol for XOR.

You'll often hear snake oil salespeople tell you that their software encrypts data with XOR. These salespeople either don't know what they are talking about or they are out to deceive you. Many buyers have discovered, to their

dismay, that a simple XOR operation is practically no encryption at all, and it's very easy to break. So if you hear someone tell you that their product "encrypts" with XOR only, you'll know that person is selling nothing more than snake oil — in other words, nothing worth purchasing. On the other hand, if XOR is done numerous times throughout the encryption process, it has the possibility of making the algorithm stronger.

XOR is possible because of binary code. You know, that code where the characters on your keyboard are converted to ones and zeroes, which correspond to the ASCII code for the characters on your keyboard? Yeah, that's the one. The binary code of 01100001 = 97 = a. Likewise, 01100010 = 98 = b. What XOR does is compare each of the zeroes and ones, in sequence, and, if the numbers are the same, it marks the spot with a zero. If the numbers are *not the same*, it places a one in that spot. It's kinda hard to explain completely in text, so I'll give you a visual example:

```
Plaintext  = baby = 01100010 01100001 01100010 01111001
XOR key    = data = 01100100 01100001 01110100 01100001
Ciphertext = ????   00000110 00000000 00010110 00011000
```

You've probably at least heard of *binary code* — the series of ones and zeroes in the example above are characters in their binary code form. Basically, a computer understands the electricity that passes through miniscule "gates" on its chips. If the gate is closed and no electricity can pass through, that's a zero. If the gate is open and the electricity can complete the connection, that's a one. But how do these ones and zeroes become characters that you can recognize?

First of all, the binary codes are limited to eight spaces. Each single space (a one or zero) is known as a *bit.* A combination of eight bits equals one *byte.* A byte also corresponds (in a computer's understanding) to a character.

If you count the ones and zeroes in the example above, you'll see that each has only eight ones and zeroes in a block of space — that's a byte. Before those ones and zeroes can be changed to a character, their numeric value is calculated first. Rather than go through the entire mathematical explanation of how to count in binary math (also known as Base 2 or 1^2), I'll give you an example you can relate to.

Not to give away my age, but I was taught how to use an abacus when I was in grade school. If you travel to Asia today, you'll see that many shopkeepers still use an abacus as a calculator to figure out how much you owe them. An abacus simply uses beads on a dowel that are used as a place-maker to help you count and add. Believe it or not, binary code numbers can be counted and added the same way. Here's how:

128	64	32	16	8	4	2	1	Value of each bit in a byte
0	1	1	0	0	0	1	0	0 = Don't Count
								1= Do Count
0	64	32	0	0	0	2	0	64 + 32 + 2 = 98

So, what you see is that each one or zero is a place holder or a marker for the numbers along the top of the table. If you see a 0 in the row below the number, it means "Don't count this number." If you see a 1 in the row below the number, it means "Yes, count this number."

You count from left to right in this case, first looking to see if there is a one in the 1 column; there isn't, so I don't count the 1. However, there is a 1 beneath the 2 column, so I do count that. As a went along, I found that I needed to count the numbers 2, 32, and 64. When you add those all together, you get the number 98.

Now comes the easy part, and that's called *ASCII* (American Standard Code for Information Interchange). This is simply a table that assigns a keyboard character to the numbers 0 to 256 (if you added all the numbers in a byte, the maximum number is 256). As it so happens, the number 98 corresponds to the lower case "b" on the ASCII table.

There are a number of Web sites that explain the ASCII table and can do conversions for you. Here's the one I used to look up the codes for my example: `www.ascii.cl/index.htm`.

The encrypted data in my example actually comes out as `^F^@^V^Z`, which are actually control codes used by your computer (holding down the Ctrl key and tapping another key produced a control code). I know if I got a message in the mail that looked like that, I wouldn't have a clue as to what it really was!

What's really cool about XOR (or maybe not, depending on whether you are using it for security or for fun) is that you can see how easily you can get back to the plaintext by reversing the operation:

```
Ciphertext = ????   00000110 00000000 00010110 00011000
XOR key    = data = 01100100 01100001 01110100 01100001
Plaintext  = baby = 01100010 01100001 01100010 01111001
```

Of course the key to decrypting the message is knowing what characters were used for the XOR key. In this example I used the word *data*.

That's all there really is to XOR. It's just magical enough to be fun for people who were never good at math (like me!) For cryptographers, coding in XOR is easy and, if there are a number of iterations of XOR throughout the algorithm, it's pretty effective at giving data a good jumble.

Breaking Ciphers

One big problem with ancient ciphers is that they were easily figured out, and the secret messages weren't secret for very long. As cryptography got more complex, the secret messages stayed secret for a longer period. As I mention in a sidebar earlier in the chapter, the Enigma machine took several years to break and it was finally cracked through a combination of eavesdropping, engineering, pattern recognition, human laziness (on the German side), and some sheer luck. The Enigma team listened and heard clacking and clicking, which told them they were dealing with a machine, and then they managed to make a duplicate machine themselves (and got it right with luck). They noticed that some messages started with the same grouping of letters and were very lucky that the Germans used the same phrases many times to synchronize remote machines.

Not-so-secret keys

If you leave the keys to your car in the ignition and the doors unlocked, what do you think the chances are that it will be stolen soon? (If it's a new Mercedes SL55 AMG valued at over $110,000, I'd say the chances are pretty good that it would be gone by morning.) The point is, if you leave the keys where other people can find them, you're the one to blame. One of the biggest weaknesses in cryptography has been the poor use or sharing of keys. Like your password, you don't write it on a sticky note and put it on your monitor. (Do people still do that?)

The art of cryptanalysis

Cryptanalysis is the art of breaking ciphers, and the National Security Agency (NSA) is renowned as one of the world's largest employers of cryptanalysts. The CIA is also very into crypto (which makes sense, as they are the home of spy versus spy), and they have a crypto challenge for anyone who wants to give it a try. When the new CIA headquarters was built in 1990, a sculpture called "Kryptos" was installed in front of the main entrance. The sculpture is an encrypted message. Part of the code has been cracked, but the man who got it spent more than 400 hours on it before he even got close to cracking just the first part. He finally managed all but the last 97 characters. If you want to give it a try, visit `www.odci.gov/cia/information/tour/kryptos_code.html`.

Key-length is mentioned a lot in books and articles about cryptography. That's because the longer the key (drum roll, please), the harder it is to guess what the key is! All the examples of keys I've given you in this chapter are very short, very easy keys. Even if these keys weren't already common knowledge, they still wouldn't take long to guess. You could probably even do it with plain old paper and pencil.

The job of keeping keys a secret has been one that has plagued us for centuries. You have to share the key at some point in some manner, or the recipient won't be able to decipher the message. This is such a major job that I've devoted all of Chapter 7 to the subject of managing keys.

Known plaintext

If you know for certain both a plaintext word and its ciphertext mate in a message, it can make cracking the message a piece of cake. For example, if you look at an encrypted message with a string of characters like XROL and you know that it means CAKE, you can go through the entire message substituting all the Xs with Cs, Rs with As, and so on. If nothing else, it can certainly give you a clue as to what the words might be. It's kind of like playing *Wheel of Fortune.* If you play around with these variations long enough, you might just discover the key for the entire message.

Pattern recognition

The first thing you look for when you're trying to break a code is a pattern in the encrypted message. For example, the letter *E* is the most commonly used letter in the English language, so you look for a letter in the message that is used more than any of the others. That *may* indicate that that particular letter is actually the letter *E*. Failing that, the second most common letter in the English language is the letter *T*.

Finding a pattern was part of the solution for the man who has partially cracked the Kryptos sculpture at the CIA headquarters building (which I talk about a couple sections back). He looked long and hard for a grouping of letters that would correspond to the word *the*. Because *the* is extremely common in the English language, it's usually a safe bet to start there, and many cryptanalysts do.

What a brute!

If you've failed to decrypt a secret message by trying to figure out the key or by looking for patterns, you might just try banging your head against the wall. Brute force might just work.

I'm kind of joking and kind of not when I list *brute force* as a method of breaking an encrypted message or file. Actually it's done quite often thanks to computers getting faster and the ability of linking computers together for strength and will power. In some cases, computers working in parallel can be more powerful than one of the most powerful computers used by the NSA.

Brute force is a trial and error method of trying every possible combination of characters against the encrypted data in an attempt to discover the key. That's one of the reasons I always stress that you use the longest keys possible. For example, a 56-bit key has 2^{56} possible keys. That's more than 72 quadrillion keys that must potentially be tried in order to find the correct one. You might think that, given those numbers, a 56-bit key would be pretty safe then. Wrong. In 1997, a distributed computing effort cracked the RSA's 56-bit RC5 encryption in less than 250 days. One of the more famous brute force cracks was that of the DES algorithm. Many people didn't think it was possible to crack DES by using brute force, and everyone in the crypto world was talking about it when it happened.

Cryptosystems

By definition, a *cryptosystem* is the combination of three elements: the *encryption engine, keying information,* and *operational procedures* for their secure use. In other words, almost every encryption program can be considered a cryptosystem because it has everything together in one package. The encryption engine is the part of the software that starts the encryption with the selected algorithm, and the keying system is the portion of the software that creates (and sometimes manages) the keys needed to encrypt and decrypt data. The operational procedures are how all of these parts interact and how the output, or result, is formatted and what file extension (if any) is used. So, almost every encryption product you buy off the shelf is, in a sense, a cryptosystem. Some people may argue that a cryptosystem is the complete infrastructure of encryption programs, hardware, and network connections, but I'll stick to the more traditional definition.

Many of the self programs include more than one algorithm for you to use for encryption. Usually these programs give you a one type of algorithm — such as symmetric or asymmetric algorithms — for you to decide upon. Sometimes you'll see a drop-down list of the algorithms for use, or there will be a configuration setting in which you choose which you care to use. The differences between the algorithms are discussed in detail in Chapter 2, but it's safe to say that they come in two types: *symmetric,* which means one key is used, and *asymmetric,* which means two keys are used.

Because symmetric algorithms work much faster than asymmetric algorithms (again, more on that in Chapter 2), some cryptosystems use both types in their software package. This is referred to as *hybrid cryptosystems*. Usually

with this type of system asymmetric algorithms are used to exchange two keys between sender and recipient, and a symmetric algorithm actually does the encryption.

That's all you need to know for now as I'll get into all of this stuff in much more detail as you read through this book, and I'll have many examples to help get the point across.

Everyday Uses of Encryption

Whether you realize it or not, there are a lot of ways that you deal with some form of encryption every day. As businesses now rely heavily on the Internet and other forms of networks to buy, sell, organize, inform, provide services, and form alliances, they also have to deal with the fact that sometimes these networks are transmitting very sensitive data. Some businesses decide on their own that protection of this data is a good thing, and others have either learn that through bad experiences or have to comply with new laws that deal with the protection of personal data.

Computers have become so insidious that many of us don't even realize sometimes that we are interacting with them. Most of these systems are encrypting the data as it goes across the wires. Have you used your debit card to buy gas at an automatic pump lately? Read on for some other examples of everyday encryption.

Network logons and passwords

When you log on to a network, either at home or at work, you are normally asked for your UserID (or User Name) and your password. When PCs first appeared, they didn't have the capability of networking, so there was no need for such security. But when networking software became generally available, businesses especially realized the need to keep unauthorized users off the networks and to compartmentalize which sections of a network the staff were allowed to roam. Hence, the UserID and password was the logical choice for controlling access.

Because there were (and are) various networking applications, not all the logon procedures were developed the same way. It's the same old story — all the vendors make their own version of networking, hoping that theirs will become the standard. Alas, we run into something very common in computing and networking, and that is applying the lowest common denominator in order to achieve interoperability.

At first, passwords were passed from the user's computer to the server in plaintext. Definitely not a good idea, but who would have thought that people would want to do harm with computers? Some sort of encryption had to be used to protect the passwords, so each of the vendors developed their own algorithms or hashes for accomplishing this task. The first encryption hashes were pretty dismal and could easily be cracked. To make a long story short, password encryption got better, but some elements of the older, crackable hashes had to be included for backward compatibility and other interoperability issues.

The remaining bad apple in the batch is called *LANMAN,* which stands for Local Area Network Manager.

LANMAN is a method of storing your password that Microsoft included so that your password could be exchanged with other non-Microsoft networks such as Novell. The trouble with LANMAN is the way that it is encrypted and stored on the computers. To be frank, it's the worst password encryption method I've ever seen — the passwords can be cracked in less time than it takes to blink an eye. Here's how the LANMAN encrypts and stores passwords:

- Passwords are converted to all uppercase characters.
- Password length is a maximum of 14 characters. If your password is longer than 14 characters, LANMAN shortens it. If your password is less than 14 characters, LANMAN "pads" it with extra character.
- When LANMAN "pads" a short password with extra characters, those extra characters are *always the same characters*, no matter what the original password is.
- The password is split into two, seven-character pieces.

So first of all, LANMAN breaks the rule of using upper- and lowercase characters in your password by changing all the characters to uppercase. For example, if your password was cATclaW, LANMAN stores it as CATCLAW. Secondly, because the next set of seven characters is always the same set of characters, hackers know they can just throw that portion away and concentrate on the first seven characters. Thirdly, the algorithm that Windows uses to encrypt the character is extremely weak and any password cracker worth its salt can crack them in no time.

Windows, by default, stores your password both in the LANMAN method and a stronger encryption method. Hackers don't have to bother much with the stronger stored version since the LANMAN is so easily cracked.

Although most Windows networks no longer have a need for LANMAN support, Windows still stores LM password hashes (also known as LANMAN hashes) by default on Windows NT, 2000 and XP systems (but not in Windows 2003). There is an article on Microsoft's support site on how to disable

LANMAN which can be found at: `http://support.microsoft.com/default.aspx?scid=kb;en-us;147706`.

If your system has LANMAN enabled and you don't absolutely need it, please disable it soon!

Secure Web transactions

The odds are in your favor that if you've ever purchased something from an online shop with your Web browser, you've interacted with at least one form of encryption. In fact, you should ensure that any shop you order from is using at least 128-bit encryption because otherwise, all of your personal information is probably not being protected — or at least not being protected to the fullest extent possible.

As I mention earlier, having your credit card number and personal data travel across the wires in the cleat is *not* a good thing! However, prior to 1995, there was no technology in place to ensure secure Web transactions. All personal data was sent in the clear, and you couldn't even verify who you were really sending that data to. Because it's very easy to hijack a transaction between your computer and a Web site without you even realizing it, you could be sending your credit card number to an imposter and you'd have no idea it happened.

In order to correct the problem of the Web sending and receiving data in the clear, some fixing had to be done to the HTTP protocol that handles the sending and receiving of data. *S-HTTP* (Secure HTTP) was created so messages and files could be sent encrypted. S-HTTP doesn't actually provide the encryption; it just makes it possible for encryption to be added on. But vendors competing against each other again resulted in the fact that not all Web browsers and servers can use S-HTTP. There go those standards again. . . .

Another fix created to solve the security problem was the creation of *SSL* (Secure Sockets Layer). SSL is designed to allow a secure connection between your browser and a Web server, and all data that travels between the two can be encrypted, not just individual messages like S-HTTP. Again, SSL doesn't actually provide the encryption; it just makes it possible for encryption to be used. SSL has become sort of a de-facto standard, and all Web browsers and servers are capable of using it. There are two levels of encryption available: 40-bit and 128-bit. The *bit* is the size of the key and, — and I'll keep harping about this — the longer the key, the better the security.

SSL and S-HTTP have very different designs and goals, so it is possible to use the two protocols together — and some merchants and banks do use both. You'll know when a secure connection has been established when a small key or lock appears in your browser's status bar (see Figure 1-2) and the URL has changed to "https" instead of just plain "http."

Figure 1-2: When you see the lock, you know you're secure.

There's much more on secure e-commerce and encryption in Chapter 11. I tell you there how to check that 128-bit encryption is being used for secure transactions. I also talk about examining Web site certificates to check that they are authentic to verify a merchant or bank's identity.

ATMs

Ahhhhh, what would I do without ATMs nowadays? It's wonderful to be able to go downtown or travel overseas and not have to worry about stacks of bills in your wallet. No need to rob a bank; just stick your card in the hole in the wall, and the machine makes money on demand for you (or at least it seems that way).

A lot of famous stories and movies are about big bank robberies, and people seem to always cheer when the crafty robber gets away will millions and lives happily ever after. But, hey! That was your money the robber stole, too! Obviously, security is a big issue here. Banks have had to add much more physical security to their ATMs in the past few years, but they are also pretty good at keeping up on the back-end security through the use of encryption.

The magnetic strip on the back of your ATM card contains a wealth of information that is picked up by the bank's computers when you slide your card into the slot (or in a merchant's point-of-sale machine). One of the bits of data on the strip is your account number, encrypted of course. When you enter your PIN, an encryption key is compared to an encrypted account number to see if they match. If they do, you're in luck. Punch in the wrong PIN too many times and the machine slurps up your card.

But just because your bank utilizes encryption when handling transactions, don't be lulled into a false sense of security. The *implementation* of encryption has to be good or the security can be breached. In 2001, two university student researchers in England found a huge hole in the way most systems were handling encryption of the account number. It wasn't the banks' fault — the vendor who sold them the systems for handling their ATM transactions had goofed. It turned out that the first four digits of the account number were always sent in the clear! The students used this information to eventually get the 3DES key used for encryption, and they were able to demonstrate that they could then crack up to 7,000 PINs per hour.

This didn't get too much media coverage in the United States, but luckily the amount of press this was getting overseas forced the computer vendor to come up with a fix for the system.

Music and DVDs

If you haven't heard of the *DMCA* (Digital Millennium Copyright Act of 1998), then you probably don't get into copying DVD movies or downloading music from the Internet. I don't want to get started on whether the DMCA is a good thing or not — that's a huge and contentious issue in itself and doesn't really have anything to do with cryptography except that the DMCA allows film and music companies to protect their copyrights with encryption.

Each DVD player sold has a computer chip in it that contains a decryption key so it can read the encrypted portion of a DVD disc. The computer chip is also contains a country code that matches where the DVD player will be sold and used (assumedly, in any case).

When the DVD discs are made, a section of the disc is encrypted by the music industry's proprietary system called *CSS* (Content Scrambling System). This system encrypts part of the disc with a country code — sort of like the country codes for the telephone system. One code is used for North America, another for the South Pacific, and so on.

Now comes the fun part. If you buy your DVD player in Japan and try to play a DVD bought in America, the disc won't play. That's because CSS is an access control system that prevents the playback of discs on players that don't have the decryption keys that the movie industry provides to authorized manufacturers. Many, many people don't know this and have found out the hard way when they try to play the movie they bought while vacationing in England orsomewhere other than the United States.

There's a young man in Norway who owned some DVD movies and wanted to watch them on a computer he built, but the computer did not have the ability to read the encrypted code on the DVD discs. So, he wrote a program he called DeCSS and installed it on his computer. It worked! He could now watch his movies on his computer. The young man was so pleased with it, he put it on the Internet to distribute for free. That was probably his first big mistake.

Part of the DMCA says that the manufacturing of or trafficking in technologies capable of circumventing technical protection measures used to restrict access to copyrighted works is highly illegal. Well, the young man did circumvent the technical protection on DVD discs, and he made it available for distribution. The guy has been tried and acquited for this crime once, but it looks like another trial may be coming up soon.

I'm not arguing who's right or who's wrong here; I just wanted you to be aware of the fact that sometimes the use of cryptography gets just a little bit sticky.

Communication devices

Until about 10 years ago, everything you said over a cell phone could be picked up and listened to over a simple, inexpensive Radio Shack radio scanner. There were huge scandals in the UK when tapes of Princess Di's cell phone conversations were released to the tabloid press. It seems a lot of people like to eavesdrop on others' conversations. Even Newt Gingrich was stung when his cell phone conversations showed up in the press.

The majority of cell phones in America have their phone numbers and voice transmissions encrypted, but only to a certain point. The part of the call between the cell phone and the tower is encrypted, but as soon as the conversation reaches your provider's gateway to the land-line phone system, it's decrypted. That's right, all land-line telephone communications (with the exception of government and military systems) are unencrypted. If you know how and where to place a gator clip and a phone receiver, you can listen in on anyone's phone calls.

The GSM (Groupe Speciale Mobile) wireless phone is the standard in Europe and is the world's most widely used cellular technology. More than 215 million digital phones use it worldwide, including more than 100 million in Europe and 5 million in the United States. GSM transmissions are encrypted, but the A5/1 algorithm keys, which are used to scramble and unscramble the data, are much shorter than advertised and thus much easier to break.

Why Encryption Isn't More Commonplace

Until fairly recently, it was unlawful for average American citizens to even own encryption technology. That was the realm of the NSA, and all encryption products were tightly controlled. In the early 1990s, a number of privacy activists and cryptographers helped loosen the restrictions on who could own encryption, and the courts have since ruled that we have the right to privacy in our communications and storage of our own data. Given that our right to own and use encryption is so new, it's not surprising that not many people know much about it.

Now the market is burgeoning with new products and encryption technologies, which makes it even more difficult for people to decide what to buy and implement — if they decide on encryption at all! However, some states are

passing laws stating that companies that store personal information need to use cryptography to protect that information. It's likely that we will see a wave a new laws like these over the next five years.

Another thought that comes to mind is the Internet. Again, until recently, we trusted the Internet and saw no need to protect ourselves. But with hacking and identity theft becoming more common, it makes sense to start looking at ways to protect ourselves and our information.

Difficulty in understanding the technology

By and large, encryption programs have suffered from a lack of intuitive interfaces — if people don't understand how to use the software, they won't use it. Period. This is the fault of the developers. They seem to have forgotten that cryptography is new to most users.

The graphical interface to many encryption programs was almost indecipherable, even to people like me who know what they are doing. It's no wonder then that people who have bought encryption products have never gotten around to use them. They don't know how to work the commands and menus.

Another reason people find cryptography so hard to understand is that the creators of cryptosystems — usually mathematicians — are the same ones who have written most of the textbooks explaining the subjects. Now, I don't want aeronautical engineers explaining to me how a plane flies because I won't understand what they are saying. For that same reason, I don't want a mathematician to explain to me how to encrypt my e-mail. Because of this I have taken the non-mathematical route to explaining how cryptography works.

Luckily, things are changing in the world of cryptography. User interfaces for encryption products are becoming easier to use, and publishers are seeing the need for easy-to-understand books (like this one) for these products. The vendors themselves are also helping by putting large amounts of "How-To" information on their Web sites with FAQs (Frequently Asked Questions) to help you find the answer to your problem.

You can't do it alone

One of the biggest problems with cryptography is that you can't do it alone! You need at least two people — a sender and a receiver. Otherwise, the encrypted files or messages just sit there. It's sort of like when the first video phones appeared — there was no sense in buying one for yourself if you

didn't have anyone to call who had one, too. What's the point in showing your face on the telephone line if there's no one on the other side to see it?

If you're going to be receiving encrypted files and messages, you need to have the same software, or compatible software, as the sender. That's simple common sense. Likewise, if you are sending encrypted files or messages, you need to be sure that the people on the receiving end have some means of decrypting what you've sent.

Luckily, many products operate on similar standards and can be made to work with similar products. It may take a bit of trial and error to get it working correctly, but the good news is that you usually have to do that only once.

Sharing those ugly secrets

If I ask you to water my houseplants while I'm on vacation, you'll obviously need the key to get into the house. If I don't have the opportunity to hand you the key in person, I'll have to hide it somewhere and then let you know where it is. I can't leave a message on the front door — that would be too easy for someone else to intercept. I shouldn't put the key under the doormat because that's usually the first place a thief looks. I could call you on the phone and tell you the location of the key, but how do I know that I'm really talking to you?

That last scenario may be a bit far-fetched, but I'm sure you can see my point by now. How do you share a secret without letting the whole world know? That, in a nutshell, is the largest problem facing cryptography — how do you safely and securely share the keys? There are tons and tons of papers and books covering this subject alone. It's safe to say that modern cryptography products do have ways of safely sharing the keys, but it takes some effort and common sense on the users' part, too.

I'll be covering keys and the correct methods of sharing them in Chapter 4. In fact, there will be so much about this subject that I'm likely to get you all keyed up! (Ouch!)

Cost may be a factor

Although there are free crypto products available for use, few of them are suitable for a business environment. Then, as with all business decisions, whether or not to employ cryptography comes down to the question of how much it will cost. As I mention earlier, you can't do it alone, so you also have to make sure that your solution will interoperate with what your partners and customers are using.

In addition to the cost of the crypto products themselves, you also have to take into account the man-hours spent just coming to a decision. There's a lot of research to do (much of which I help you with), and you may need to add servers and sub-nets to your existing network. There is time and money involved in the setup and configuration of the system, the training of users, and personnel to handle maintenance of the system.

This may seem like an overwhelming task at first, but I help you break it down so you can make a decision that you can ultimately afford. When it comes to crypto products, newer is not necessarily better, as I discuss throughout this book. The last thing you want to do is to buy crypto products solely based on their "gee-whiz" appeal.

Special administration requirements

Crypto products require special handling, which means that you need to have experienced staff to operate and maintain the systems. This is not something the accountant can do as an adjunct to his or her normal duties; you need a skilled professional. Why? Because if your crypto systems are not set up and maintained correctly, you run the risk of exposing all of your secrets. In addition, your staff will lose their keys and forget their passphrases, and new users need to be added to the system and trained on its use. If you're trying to increase the security of your system and protect your company's assets, you might as well do the job as well as you can. In this case, "good enough" sometimes *isn't* enough.

You'll find information on identifying your requirements, deciding what you need, and telling good products from bad products in the following chapters. I also give you sneak peeks into various products you may encounter, and I give you a really good description of that incredibly elusive beast, *PKI* (Public Key Infrastructure). I explain more about PKI in the chapters in Part II.

Thanks for staying with me so far — now let the journey begin.