Chapter 1

Cryptography Basics

IN THIS CHAPTER

- The basics of cryptography
- Applications of cryptography
- Digital signatures

FROM THE DAWN OF CIVILIZATION, to the highly networked societies that we live in today — communication has always been an integral part of our existence. What started as simple sign-communication centuries ago has evolved into many forms of communication today — the Internet being just one such example. Methods of communication today include

- Radio communication
- Telephonic communication
- Network communication
- Mobile communication

All these methods and means of communication have played an important role in our lives, but in the past few years, network communication, especially over the Internet, has emerged as one of the most powerful methods of communication — with an overwhelming impact on our lives.

Such rapid advances in communications technology have also given rise to security threats to individuals and organizations. In the last few years, various measures and services have been developed to counter these threats. All categories of such measures and services, however, have certain fundamental requirements, which include

- **Confidentiality**, which is the process of keeping information private and secret so that only the intended recipient is able to understand the information. For example, if Alice has to send a message to Bob, then Bob only (and no other person except for Bob) should be able to read or understand the message.
- **Authentication**, which is the process of providing proof of identity of the sender to the recipient, so that the recipient can be assured that the person sending the information is who and what he or she claims to be. For example, when Bob receives a message from Alice, then he should be able

to establish the identity of Alice and know that the message was indeed sent by Alice.

- **Integrity**, which is the method to ensure that information is not tampered with during its transit or its storage on the network. Any unauthorized person should not be able to tamper with the information or change the information during transit. For example, when Alice sends a message to Bob, then the contents of the message should not be altered with and should remain the same as what Alice has sent.
- **Non-repudiation**, which is the method to ensure that information cannot be disowned. Once the non-repudiation process is in place, the sender cannot deny being the originator of the data. For example, when Alice sends a message to Bob, then she should not be able to deny later that she sent the message.

Before we look at the various mechanisms that provide these security services, let us look at the various types of security attacks that can be faced by an organization:

- **Interruption**: In an attack where one or more of the systems of the organization become unusable due to attacks by unauthorized users. This leads to systems being unavailable for use. Figure 1-1 displays the process of interruption.

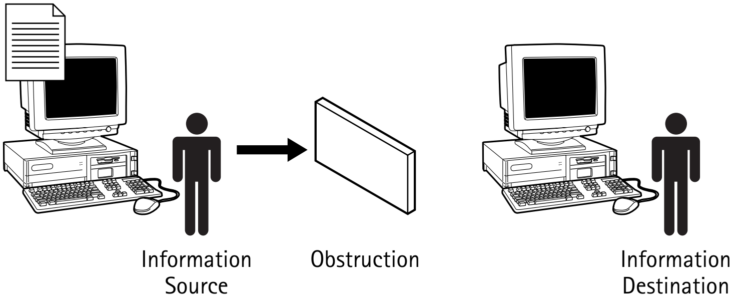

Figure-1-1: Interruption

- **Interception**: An unauthorized individual intercepts the message content and changes it or uses it for malicious purposes. After this type of attack, the message does not remain confidential; for example, if the contents of message that Alice sends to Bob are read or altered during its transmission of message by a hacker or an interceptor. In this situation, Bob cannot consider such a message to be a confidential one. Figure 1-2 displays the process of interception.
- **Modification**: The content of the message is modified by a third party. This attack affects the integrity of the message. Figure 1-3 displays the process of modification.

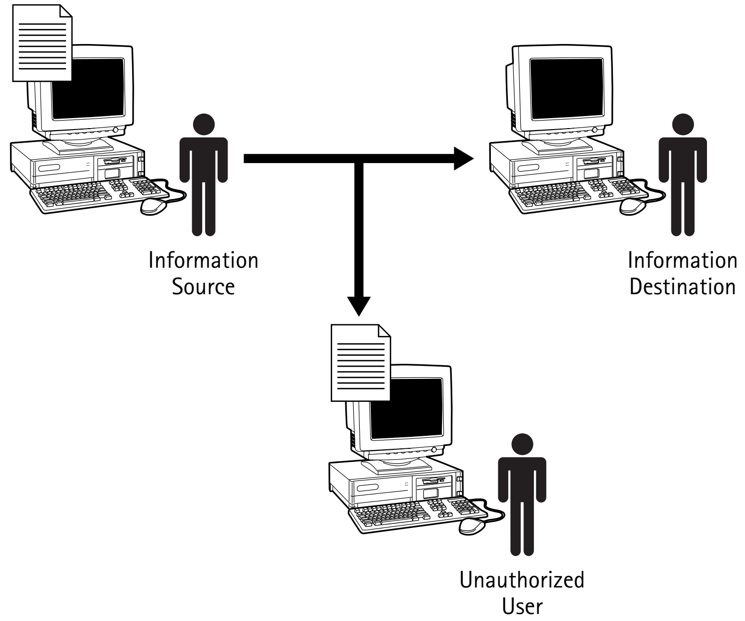

Figure 1-2: Interception

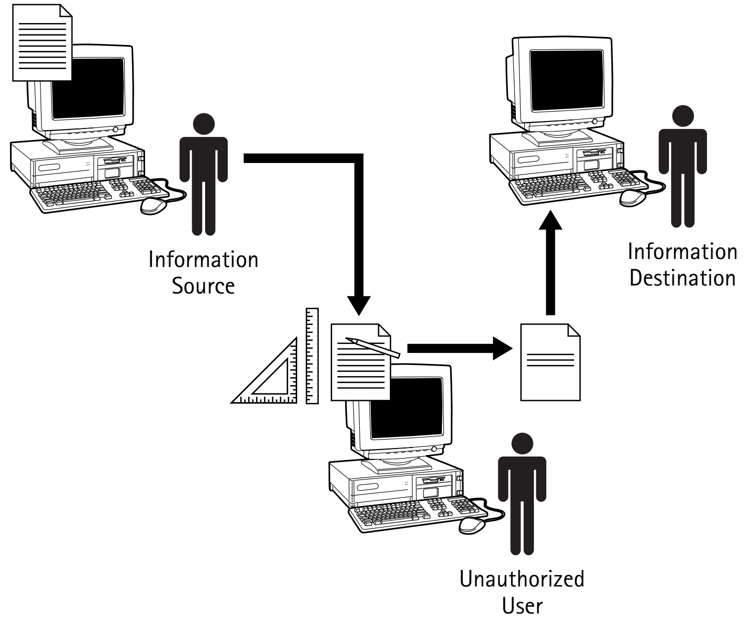

Figure 1-3: Modification

- **Fabrication:** In this attack, a third party inserts spurious messages into the organization network by posing as a valid user. This attack affects the confidentiality, authenticity, and integrity of the message. Figure 1-4 displays fabrication.

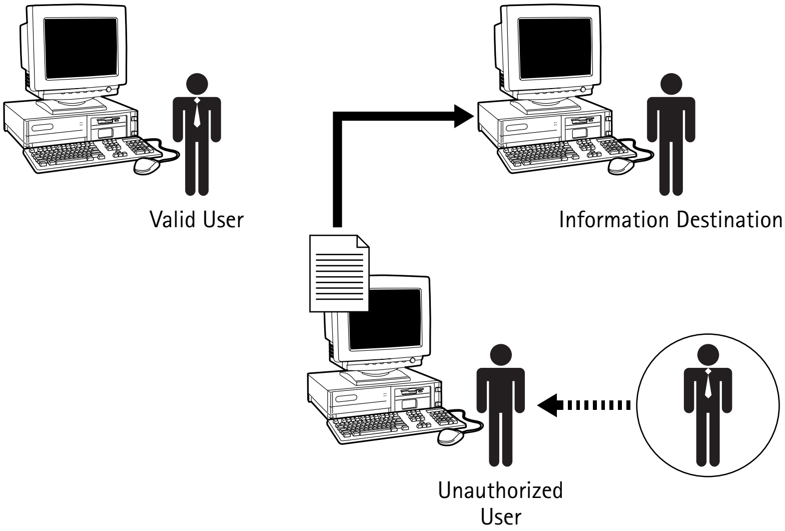

Figure 1-4: Fabrication

From securing sensitive military information to securing personal messages, you often would be confronted with the need of masking information to protect it. One of the most important methods that help provide security to messages in transit is *cryptography*. It helps overcome the security issues as described above, involved in the delivery of messages over any communication channel. This chapter provides an overview of cryptography and popular cryptographic techniques.

The term *cryptology* has its origin in the Greek kryptós lógos, which means "hidden word." Other examples of cryptography date back to circa 1900 B.C. when Egyptians began using hieroglyphics in inscriptions.

The Basics of Cryptography

Cryptography is the science of protecting data, which provides means and methods of converting data into unreadable form, so that

- The data cannot be accessed for unauthorized use.
- The content of the data frames is hidden.
- The authenticity of the data can be established.
- The undetected modification of the data is avoided.
- The data cannot be disowned by the originator of the message.

Cryptography is one of the technological means to provide security to data being transmitted on information and communications systems. Cryptography is especially useful in the cases of financial and personal data, irrespective of the fact that the data is being transmitted over a medium or is stored on a storage device. It provides a powerful means of verifying the authenticity of data and identifying the culprit, if the confidentiality and integrity of the data is violated. Because of the development of electronic commerce, cryptographic techniques are extremely critical to the development and use of defense information systems and communications networks.

History of Cryptography

As already discussed, the messages were first encrypted in ancient Egypt as a result of hieroglyphics. The Egyptians encrypted messages by simply replacing the original picture with another picture. This method of encryption was known as substitution cipher. In this method, each letter of the cleartext message was replaced by some other letter, which results in an encrypted message or ciphertext. For example, the message

```
WELCOME TO THE WORLD OF CRYPTOGRAPHY
```

can be encrypted by using substitution cipher as

```
XFMDPNF UP UIF XPSME PG DSZQUPHSBQIZ
```

In the preceding example, each letter of the plaintext message has been replaced with the next letter in the alphabet. This type of substitution is also known as Caesar cipher.

Caesar cipher is an example of shift cipher because it involves shifting each letter of the plaintext message by some number of spaces to obtain the ciphertext. For example, if you shift the letters by 5, you get the following combination of plaintext and ciphertext letters:

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

However, simple substitution ciphers are not a very reliable type and can easily be broken down. In such a case, an alternative way is to use multiple alphabets instead of one alphabet. This type of a cipher, which involves multiple cipher alphabets, is known as a *polyalphabetic substitution cipher*. An example of the polyalphabetic substitution cipher is the Vigenere cipher.

With the recent advances in mathematical techniques, there has an acceleration in the development of newer methods of encryption. Today, cryptography has emerged so powerful that it is considered rather impossible to break some ciphers.

Cryptography has now become an industry standard for providing information security, trust, controlling access to resources, and electronic transactions. Its use is no longer limited to just securing sensitive military information. In fact, cryptography is now recognized as one of the major components of the security policy of an organization.

Before moving further with cryptography, let us first look at a few terms that are commonly associated with cryptography:

- **Plaintext:** Is the message that has to be transmitted to the recipient. It is also commonly referred to as *cleartext*.
- **Encryption:** Is the process of changing the content of a message in a manner such that it hides the actual message.
- **Ciphertext:** Is the output that is generated after encrypting the plain text.
- **Decryption:** Is the reverse of encryption and is the process of retrieving the original message from its encrypted form. This process converts ciphertext to plaintext.
- **Hash algorithm:** Is an algorithm that converts text string into a string of fixed length.
- **Key:** Is a word, number, or phrase that is used to encrypt the cleartext. In computer–based cryptography, any text, key word, or phrase is converted to a very large number by applying a hash algorithm on it. The large number, referred to as a key, is then used for encryption and decryption.
- **Cipher:** Is a hash algorithm that translates plaintext into an intermediate form called *ciphertext*, in which the original message is in an unreadable form.
- **Cryptanalysis:** Is the science of breaking codes and ciphers.

Before looking at the details of various cryptographic techniques, let us now look at the steps involved in the conventional encryption model:

1. A sender wants to send a *Hello* message to a recipient.
2. The original message, also called plaintext, is converted to random bits known as ciphertext by using a key and an algorithm. The algorithm

being used can produce a different output each time it is used, based on the value of the key.

3. The ciphertext is transmitted over the transmission medium.
4. At the recipient end, the ciphertext is converted back to the original text using the same algorithm and key that were used to encrypt the message.

This process is also shown in Figure 1-5.

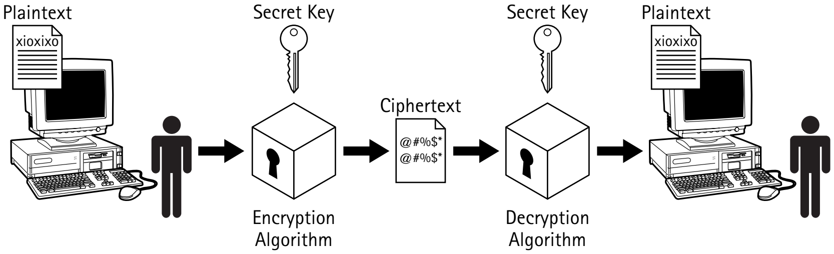

Figure-1-5: Conventional encryption model

Having looked at an overview of cryptography, let us now look at the various cryptography techniques available. For the purpose of classification, the techniques are categorized on the basis of the number of keys that are used. The two main cryptography techniques are

- **Single key cryptography:** This cryptography technique is based on a single key. It is also known as symmetric key or private key or secret key encryption.
- **Public key cryptography**: This cryptography technique is based on a combination of two keys — secret key and public key. It is also known as asymmetric encryption.

Let us look at each of these methods in detail.

Single Key Cryptography

The process of encryption and decryption of information by using a single key is known as secret key cryptography or *symmetric key cryptography*. In symmetric key cryptography, the same key is used to encrypt as well as decrypt the data. The main problem with symmetric key algorithms is that the sender and the receiver have to agree on a common key. A secure channel is also required between the sender and the receiver to exchange the secret key.

Here's an example that illustrates the process of single key cryptography. Alice wants to send a "For Your Eyes" message to Bob and wants to ensure that only Bob is able to read the message. To secure the transmission, Alice generates a secret key, encrypts the message with this key, and sends the message to Bob.

Figure 1-6 represents the process of secret key cryptography.

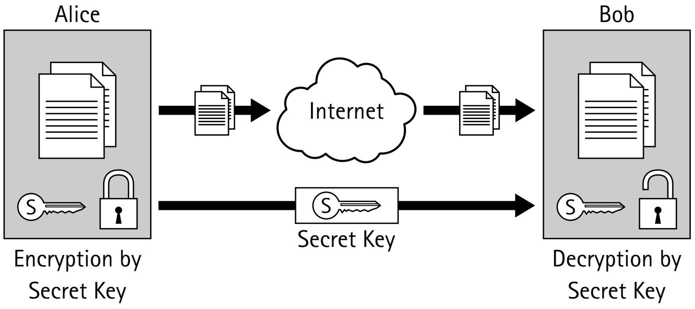

Figure 1-6: Secret key cryptography

Now, to read the encrypted message, Bob would need the secret key that has been generated by Alice. Alice can give the secret key to Bob in person or send the key to Bob by any other means available. If Alice sends the key to Bob in person, it could be time-consuming depending on the physical distance between the two of them or other circumstances such as Bob's availability. After Bob receives the secret key, he can decrypt the message to retrieve the original message.

Many secret key algorithms were developed on the basis of the concept of secret key cryptography. The most widely used secret key algorithms include

- Data Encryption Standard (DES)
- Triple-DES (3DES)
- International Data Encryption Algorithm (IDEA)
- RC4
- CAST-128
- Advanced Encryption Standard (AES)

Let us consider these algorithms in detail in the following sections.

DATA ENCRYPTION STANDARD (DES)

DES, which is an acronym for the Data Encryption Standard, is the common name for the Federal Information Processing Standard (FIPS) 46-3. It describes the *Data Encryption Algorithm* (DEA). DEA is also defined in the ANSI standard X3.92. The DES algorithm is one of the most widely used encryption algorithms in the world. The Data Encryption Standard (DES) algorithm was developed by the IBM team in the 1970s and was adopted by National Institute of Standards and Technology (NIST) for commercial applications.

NOTE

Refer to RFCs 1827 and 2144 for more information on DES.

DES is still surrounded by controversy. This controversy was originally fueled by the following facts:

- The key length used by this algorithm was reduced to 56 bits by the U.S. government, although the original design called for a key length of 128 bits, leading to a compromise on security. Although the algorithm for DES was published, the rationale for the design was never published.
- DES became widely available to the U.S. public and to approved users in other countries. However, DES was excluded by the U.S. government from protection of any of its own classified information.

The major weaknesses and attacks that are faced by DES are described below.

BRUTE FORCE ATTACK The simplest attack to decipher a DES key is the brute force attack. The brute force attack on the DES algorithm is feasible because of the relatively small key length (56 bit) and ever-increasing computational power of the computers. Until the mid-1990s, brute force attacks were beyond the capabilities of hackers because the cost of computers that were capable of hacking was extremely high and unaffordable. With the tremendous advancement in the field of computing, high-performance computers are relatively cheaper and, therefore, affordable. In fact, general purpose PCs today can be successfully used for brute force attacks. Many hackers today are using more powerful techniques, such as Field Programmable Gate Array (FPGA) and Application-Specific Integrated Circuits (ASIC) technology that provide faster and cheaper means of hacking.

You can break through any cipher by trying all keys that possibly exist. However, in brute force attacks, the time taken to break a cipher is directly proportional to the length of the key. In a brute force attack, keys are randomly generated and applied to the ciphertext until the legitimate key is generated. This key decrypts the data into its original form. Therefore, the encryption key length is a major factor that needs to be considered while choosing a key. The longer the encryption keys, the stronger the security. For example, in case of a 32-bit long key, the number of steps required to break the cipher are about 2^{32} or 10^9. Similarly, a 40-bit key requires about 2^{40} steps. This is something which can be achieved in one week by anyone sitting on his personal computer. A 56-bit key is known to have been broken by professionals and governments by using special hardware in a few months time. Today, 128-bit encryption is considered to be the safest and most reliable means of encrypting messages.

On January 19, 1999, a group of computer enthusiasts from all over the world formed a coalition to decipher a DES encrypted ciphertext and as a result recovered the key in a record-breaking time of 22 hours and 15 minutes. This coalition was known as Distributed.Net. Its members worked with DES Cracker and a worldwide network of nearly 100,000 PCs on the Internet to recover the key. The DES Cracker machine was specially designed for this purpose.

For more information on brute force attacks, refer to RFCs 2228 and 2557.

DIFFERENTIAL CRYPTANALYSIS ATTACK The differential cryptanalysis attack looks specifically at pairs of ciphertexts whose plaintext have some specific differences. It analyzes these differences as the plaintext propagates through the various rounds of DES when they are encrypted with the same key.

This technique chooses pairs of plaintext with a fixed difference. Two plaintexts can be chosen at random, as long as they satisfy specific difference conditions. Then, using the differences in the resulting ciphertexts, different probabilities can be assigned to different keys. As more and more ciphertext pairs are analyzed, one key emerges, as the most probable candidate key.

For more information on differential cryptanalysis attack, refer to RFC 2144.

LINEAR CRYPTANALYSIS ATTACK Linear Cryptanalysis attack was invented by Mitsuru Matsui in 1993. This method is based on the concept that if you XOR some of the plaintext bits together, XOR some ciphertext bits together, and then XOR the results, you will get a single bit that is the XOR of some of the key bits. A large number of such plaintexts/ciphertexts pairs are then used to guess the values of the key bits. The greater the volume of the base data, the more reliable is the guess.

For more information on linear cryptanalysis attacks, refer to RFC 2144.

TRIPLE DATA ENCRYPTION STANDARD (3DES)

Triple-DES is a minor variation of DES. Although, three times slower than DES, it can be much more secure, if used properly. In today's scenario, Triple-DES is implemented more widely than DES. This is because DES is easy to break with the help of advanced technology that is widely available today. On the other hand, 3DES has proved to be an extremely reliable solution because of the longer key length that it uses. This extended length of key plays an important role in eliminating many of the shortcut attacks that can be used to reduce the amount of time it takes to break DES.

You can increase the effective key length of your cryptographic system by using the Triple Pass DES through the process known as EDE (Encrypt, Decrypt, and Encrypt). When you use triple pass DES, it first encrypts the plaintext data with a 56-bit key. The ciphertext so obtained is then decrypted by using a different key. When you decrypt ciphertext with some different key it gives some garbage. Finally, you encrypt the garbage with the first key. This process of using triple pass DES for encryption, decryption, and again encryption is commonly referred to as EDE.

Figure 1-7 explains the process of 3DES. This is why this encryption method is referred to as "Triple-DES."

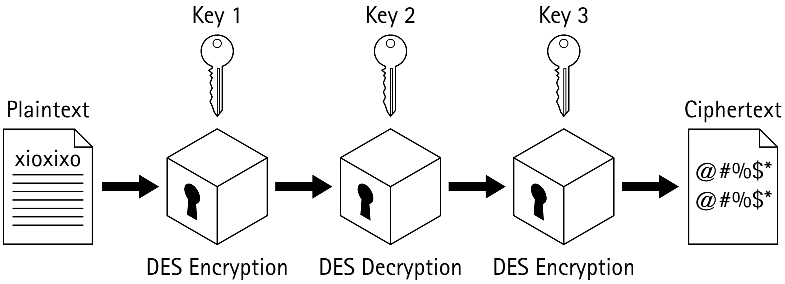

Figure 1-7: Process of 3DES

NOTE Triple-DES has been adopted by ANSI as the standard X9.52 and has been proposed as a revision to FIPS 46, known as draft FIPS 46-3.

NOTE Refer to RFCs 1828 and 2420 for more information on Triple-DES.

INTERNATIONAL DATA ENCRYPTION ALGORITHM (IDEA)

The International Data Encryption Algorithm (IDEA) is a symmetric block cipher developed by Xuejia Lai and James Massey of the Swiss Federal Institute of technology. It uses a 128-bit key to encrypt data in blocks of 64 bits. This is why it is referred to as a block cipher method. IDEA is designed to facilitate both software and hardware implementation.

The major factors that make IDEA a strong algorithm are:

- The key length is long enough to prevent comprehensive key searches. IDEA uses a key length of 128 bits, which makes it very secure.
- The ciphertext is not easily decipherable from the plaintext and the key. IDEA effectively masks the statistics of how the ciphertext depends on the statistics of the plaintext.

IDEA was developed to provide a high level of security with ease of implementation. Due to its strength and reliability IDEA is now used worldwide in many banking and industry applications.

You can find more information about the use of the IDEA Encryption Algorithm in a Certificate Management System in RFC 3058.RC2

RC2 or Ron's Code 2 is a 64-bit block cipher that was designed by Ron Rivest. It uses variable-sized keys. This algorithm was designed to replace DES. The code for this algorithm was not made public. However, many companies have licensed RC2 for use in their products. RC2 is being used in a number of software packages, such as Lotus Notes, Microsoft Windows, Internet Explorer, and Netscape Communication's Navigator and Communicator. In addition, RC2 forms an integral component of S/MIME as it provides privacy and interpretability between the export versions and domestic versions of products that use S/MIME.

You can find more information about RC2 in RFC 2268.

RC4

RC4 is a cipher that was also designed by Ron Rivest, who was the co-inventor of the RSA cipher. It is used in a number of commercial systems like Lotus Notes and Secure Netscape.

XREF

For more information on RSA, refer to the RSA section in this chapter.

It is a cipher with a key size of up to 2048 bits (256 bytes). It is listed in the category of relatively fast and strong cipher methods. It is a stream cipher that creates a stream of random bytes and XORs these bytes with the text. Using RC4 with the same key on two different messages makes it very weak. It is thus useful in situations, in which a new key can be chosen for each message.

NOTE

You can find more information about RC4 in RFC 2246.

RC5

RC5 is yet another block cipher designed by Ron Rivest for RSA Security in 1994. Along with a variable key size, and a variable number of rounds, the size of RC5 data blocks is variable. The block size can range from 32 bits, 64 bits, to 128 bits. Similarly, the number of rounds can range from 0 to 255, while the key can range from 0 bits to 2040 bits in size.

NOTE

You can find more information about RC4 in RFC 2040.

CAST-128

Carlisle Adams developed CAST-128 in May 1997. This algorithm uses a variable key length and uses block sizes of 64 bits.

The key lengths supported by CAST-128 vary from 40 bits to 128 bits, in increments of 8 bits. For key sizes that range up to 80 bits, the data block undergoes 12 rounds of encryption, while for key sizes of more than 80 bits, the algorithm has 16 rounds. For the keys whose sizes are less than 128 bits, zeroes are added to the rightmost (or the least significant) bits until the total length of the key result is 128 bits. This is done because the algorithm must have an input key of 128 bits in length.

CAST-128 has shown very good encryption/decryption performance. Its implementation has processed up to 3.3 MB/sec on a 150 MHz Pentium processor.

You can find more information about using the CAST-128 Encryption Algorithm in a Certificate Management System in RFC 2984.

ADVANCED ENCRYPTION STANDARD (AES)

With an estimated growth rate of two times every 18 months, computational power is growing in leaps and bounds. This has made Data Encryption Standard (DES) more and more insecure and vulnerable to malicious attacks. As a result, DES, which was the Federal Information Processing Standard (FIPS) until recently, has slowly become redundant. The National Institute of Standards and Technology (NIST) realized this situation and recognized the need for another standard that would be more secure than the DES. However, since DES is a federal standard, it is used widely by many organizations, particularly those in the financial industry.

Advanced Encryption Standard (AES) emerged as a powerful replacement of DES during a competition held by NIST. The competition was organized to develop a substitute of existing DES. The following algorithms reached the final round of the competition to become AES:

- **MARS**: An algorithm developed by IBM.
- **RC6**: An algorithm developed by Ron Rivest of RSA Labs, the creator of the widely used RC4 algorithm.
- **Twofish**: An algorithm from Counterpane Internet Security, Inc. This design was highly suited for large microprocessors and smart card microprocessors.
- **Serpent**: An algorithm designed by Ross Anderson, Eli Biham, and Lars Knudsen.
- **Rijndael**: An algorithm designed by Daemen and Rijmen.

Of these algorithms, Rijndael was judged the best and announced to be the new AES. The design of Rijndael was strongly influenced by another cipher called Square, which was also created by Daemen and Rijmen.

Some of the key features of Rijndael are:

- It is a secret key block cipher.
- It allows 128-, 192-, and 256-bit key lengths. The block sizes used could be 128-, 192-, or 256-bits long.
- It gives a vast speed improvement over DES. It is capable of encrypting up to 8.8 MB/sec on a 200 MHz Pentium Pro.

National Institute of Standards and Technology (NIST) chose Rijndael, due to its simplicity and high performance. It is fast, compact, and has a very simple mathematical structure.

PROBLEMS IN SYMMETRIC CRYPTOGRAPHY

The major problem with symmetric cryptography is that the process of transferring keys to the recipient is prone to security risks. Transferring the secret key over the Internet either in an e-mail message or through simple IRC services is insecure. Verbally communicating the key over a phone line runs the risk of eavesdropping. Similarly, snail mail runs the risk of possible interception. The security risks that are involved in secret key cryptography have been overcome to a large extent in another method of cryptography called public key cryptography. Public key cryptography uses a key pair instead of just one secret key. Of this key pair, one key, known as the private key, is always kept secret by the key holder. This private key is not transferred to anyone and is stored securely by the holder of the key and thus public key cryptography eliminates the need for transferring the private key. Let us take an example where Alice wants to send an encrypted message to Bob. If she is using symmetric key encryption, then both Alice and Bob need to first establish a secret key. Only after this secret key has been established, can they both communicate. However, if Alice uses public key encryption, she can send an encrypted message to Bob without first transmitting a secret key. This not only solves the problem of key distribution but also makes the process of key management a lot simpler. In addition to this, public key cryptography also provides data integrity, authentication, and non-repudiation. Public key encryption can also be used for creating digital signatures, which are used for user authentication. Let us now discuss public key cryptography in detail.

Public Key Cryptography

The approach called *asymmetric cryptography* evolved to address the security issues posed by symmetric cryptography. This method solves the problem of secret key cryptography by using two keys instead of a single key. Asymmetric cryptography uses a pair of keys. In this process, one key is used for encryption, and the other key is used for decryption. This process is known as asymmetric cryptography because both the keys are required to complete the process. These two keys are collectively known as the *key pair*. In asymmetric cryptography, one of the keys is freely distributable. This key is called the *public key* and is used for encryption. Hence, this method of encryption is also called public key encryption. The second key is the secret or private key and is used for decryption. The private key is not distributable. This key, like its name suggests, is private for every communicating entity.

In public key cryptography, the data that is encrypted with the public key can only be decrypted with the corresponding private key. Conversely, data encrypted with the private key can only be decrypted with the corresponding public key. Due to this asymmetry, public key cryptography is known as asymmetric cryptography.

HOW DOES PUBLIC KEY CRYPTOGRAPHY WORK?

Let's see how this works out in practice. Consider an example, where Alice wishes to send an encrypted file to Bob. In this situation, Bob would obtain a key pair, retain the private key, and distribute the public key. Alice, therefore, has a copy of Bob's public key. Alice then encrypts the file using Bob's public key and sends the encrypted file to Bob. Since the key pairs are complementary, only Bob's private key can decrypt this file. If someone else intercepts the file, they will be unable to decrypt the file, because only Bob's private key can be used for the decryption. Figure 1-8 explains the process of public key cryptography.

NOTE

In today's world, symmetric algorithms are used to handle the data in protocols while asymmetric algorithms are just used for key exchange due to the speed. This helps in striking a balance between speed and security.

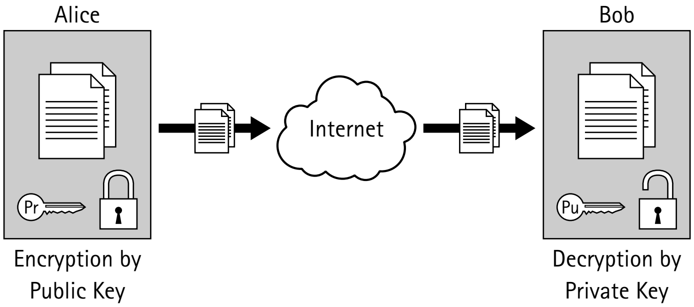

Figure 1-8: Public key encryption

This method very clearly indicates that the data you send to a user can only be encrypted by the public key. Similarly, the decryption can be done only by the private key, which is supplied by the recipient of the data. So, there is very little possibility of the data in transit being accessed or tampered by any other person. Therefore, messages can be exchanged securely. The sender and receiver do not need to share a key, as required for symmetric encryption. All communications involve only public keys, and no private key is ever transmitted or shared. The above mechanism also brings out the point that every recipient will have a unique key that he will use to decrypt the data that has been encrypted by its counterpart public key. Diffie and Hellman first discussed the process of asymmetric cryptography. One of the most common implementations of this process is the RSA algorithm.

You can find more information about the Diffie-Hellman Key Agreement Method in RFC 2631.

Let us now look at the RSA algorithm in detail.

RSA

RSA refers to a particular implementation of public key cryptography; RSA has become the de facto standard in this field, to the point that RSA and public key encryption are often used as synonyms.

In a cryptographic system with public keys, each object, person or party, must own one public key, which is publicly accessible to all other parties, and one private key, which must be kept secret. Hence, global communication requires only $2n$ keys, where n is the number of users. The procedure for the sending of a message from User A to User B is performed in the following way:

- User A obtains the public key of User B from a publicly accessible, authoritative place.
- User A then encrypts its message using this public key.
- User B receives the message and decrypts it with his/her private key.

The basic idea of this system was invented by Whitfield Diffie and Martin Hellman and is also used in RSA algorithm.

ADVANTAGES OF RSA RSA offers a few advantages that have helped in the achievement of manageable and more secure transactions. These advantages include

- Simplification of the problem of key management: In symmetric encryption the number of keys required to allow n entities to communicate is proportional to n^2. Whereas in asymmetric encryption each participant needs two keys; therefore, the total number of keys required is simply $2*n$. The growth in the number of keys with the growth in the number of users is linear and therefore manageable even when there are a large number of users.
- Enhanced security of the transactions: Not only is the number of keys greatly reduced but the security offered by these keys is highly increased. Every user must have a pair of keys that he/she generates for himself/herself. The secret key must not be shared with anyone and so the problem of transmitting it does not arise, nor do the problems of secure channels and their management; the secret key really is secret, since it is shared with

nobody. The public key, however, is shared with everyone, for example in a catalog, which it can be transmitted using the most convenient method, and therefore does not pose any problems regarding its privacy.

RSA has now become an industry standard for encryption. In fact, such is the strength of RSA that the U.S. government has restricted its export to foreign countries.

POSSIBLE ATTACKS ON RSA The RSA algorithm, although widely prevalent, has some weaknesses. Some of the common attacks that could be faced by RSA are

- Factoring of the public key: At present RSA seems to be extremely secure. It has survived over 20 years of scrutiny and is in widespread use throughout the world. The attack that is most often considered for RSA is the factoring of the public key. If this can be achieved, all messages written with the public key can be decrypted.
- Cycle attack: In this attack, the ciphertext is decrypted repeatedly, until the original text appears. A large number of recycles might be able to decrypt any ciphertext. Again, this method is very slow, and for a large key it is not a practical attack.

In spite of all the weaknesses of RSA, it continues to be regarded as a de facto industry standard for encryption, especially data transmitted over the Internet.

Combining Techniques: Symmetric and Asymmetric Encryption

The disadvantage of using public key encryption is that it is a slow process because key lengths are large (1024 bits to 4094 bits). When you compare both processes, secret key encryption is significantly faster as the key length is less (40 bits to 256 bits). On the other hand, there is a problem in transferring the key in secret key encryption. Both these techniques can be used together to provide a better method of encryption. This way you can make use of the combined advantages and overcome the disadvantages.

The steps in data transaction in a combined technique are:

1. Encrypt your file by using a symmetric encryption.
2. Use asymmetric encryption to encrypt only this key using the recipient's public key. Now send the encrypted key to the recipient. The recipient, at his end, can now decrypt the key using his/her private key.
3. Next, send the actual encrypted data. The encrypted data can be decrypted using the key that was encrypted by using the public key from the asymmetric key pair.

Figure 1-9 displays the combined technique of encryption.

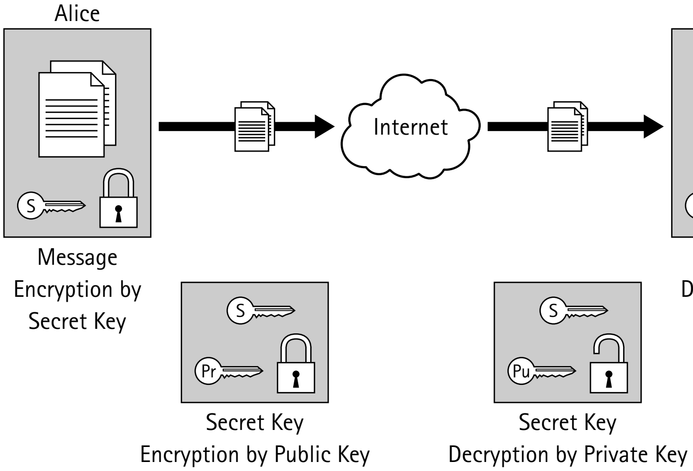
Alice
Bob
Internet
S
S
Message Encryption by Secret Key
Message Decryption by Secret Key
S
Pr
S
Pu
Secret Key Encryption by Public Key
Secret Key Decryption by Private Key

Figure 1-9: Combined technique of encryption

The combined technique of encryption is used widely. It is basically used for Secure Shell (SSH), which is used to secure communications between a client and the server and PGP (Pretty Good Privacy) for sending messages. Above all, it is the heart of Secure Sockets Layer (SSL), which is used widely by Web browsers and Web servers to maintain a secure communication channel with each other.

Applications of Cryptography

By now, you would have understood various cryptography techniques and their advantages and disadvantages. Let us now look at the implementation of cryptography to provide basic security features, which are, confidentiality, integrity, authentication, and non-repudiation.

All these security features can be provided by using any one of the following methods:

- Message encryption
- Message Authentication Code (MAC)
- Hash functions

Let us discuss each of these implementations in detail.

Message Encryption

There are multiple variations of message encryption. Messages can be encrypted either by using secret key encryption or by using public key encryption. Let us look at both the methods in detail.

USING SECRET KEY ENCRYPTION TO PROVIDE CONFIDENTIALITY AND AUTHENTICATION

Conventional encryption methods serve the purpose of authentication, integrity, and confidentiality. Let us look at an example, where Alice wants to send a message to Bob. Only Alice and Bob know the secret key, and no other party knows about the secret key. If Alice sends a message using the secret key to Bob, then Bob knows that the message is coming from Alice, as only Bob and Alice know the secret key. Once the ciphertext reaches Bob, he decrypts the message using the secret key and generates the original plaintext. If Bob recovers the plaintext by using his secret key, this means that the data has not been tampered with during transmission. If Bob is unable to recover the data, this means that someone else might have used the secret key and altered the contents of the message. If the contents of the message are altered then Bob will not be able to decrypt the message.

Figure 1-10 explains this process.

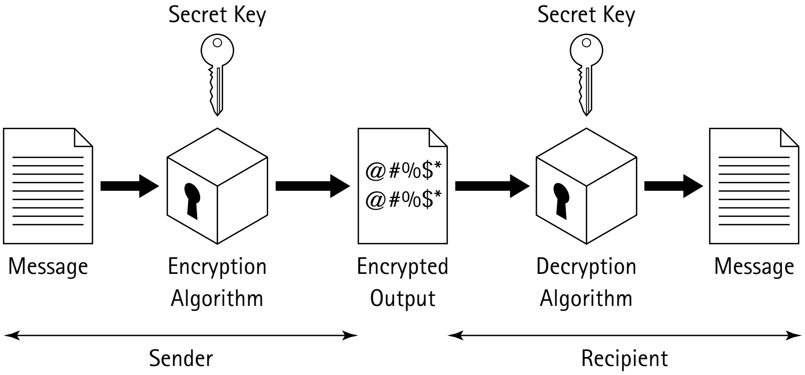

Figure 1-10: Using symmetric key encryption to provide confidentiality and authentication

Hence, the conventional encryption gives both confidentiality and authenticity to messages. However, this method does not provide information about the integrity of data.

USING SECRET KEY ENCRYPTION FOR CONFIDENTIALITY, AUTHENTICATION, AND INTEGRITY

Now let's take an example, where Bob receives a ciphertext from Alice and he decrypts it. Bob can decrypt any ciphertext and produce an output, which will be a plaintext. However, he will get a meaningful output only when Alice has sent the message. Otherwise, the plaintext generated by Bob will be a meaningless sequence of bits. Hence, there must be some automated process at Bob's end to verify that the plaintext he has recovered is a legitimate message and has come from Alice.

If the original plaintext is in a clear message in plain English then determination is easier, because it will generate a meaningless sequence that makes it easier to detect the legitimacy of the message. But if the original message is some binary object file or a digitized image, then it may be difficult to detect the integrity of the message.

To overcome this problem, one solution is to append an error detecting code to the original message, known as *frame check sequence* (FCS). So now if Alice wants to send a message M to Bob, Alice uses a function FN, which produces an output, FCS. Next, Alice will append this output FCS to the original message M. Then, the entire message along with the FCS will be encrypted using the secret key and will be sent to Bob. Bob will decrypt the entire message with the secret key and will get the message M, and the appended output FCS. Now Bob will put the Message M to the same function, which Alice had used to generate FCS, and produce the FCS. He will compare this FCS with the appended FCS, which has come with the message. If both are the same, then the message is considered legitimate.

This method provides both integrity as well as authenticity. Figure 1-11 explains this process.

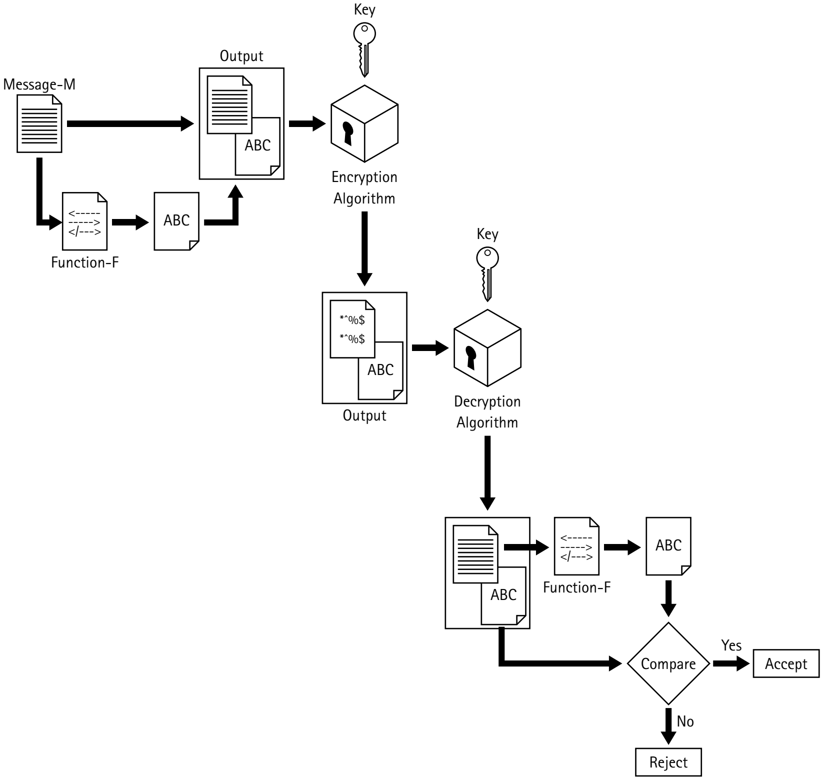

Figure 1-11: Using symmetric key encryption to provide confidentiality, authentication, and integrity

USING PUBLIC KEY ENCRYPTION TO PROVIDE CONFIDENTIALITY

A simple use of public key encryption can provide confidentiality but can't provide authenticity and integrity. Let us take an example where Alice wants to send a message to Bob. She encrypts the message with Bob's public key, and Bob decrypts the message using his private key. This method does not provide any authentication that the message is coming from Alice, because Bob's public key is known to the world. However, it does provide confidentiality to the message, as only Bob can decrypt the message. Figure 1-12 depicts this process.

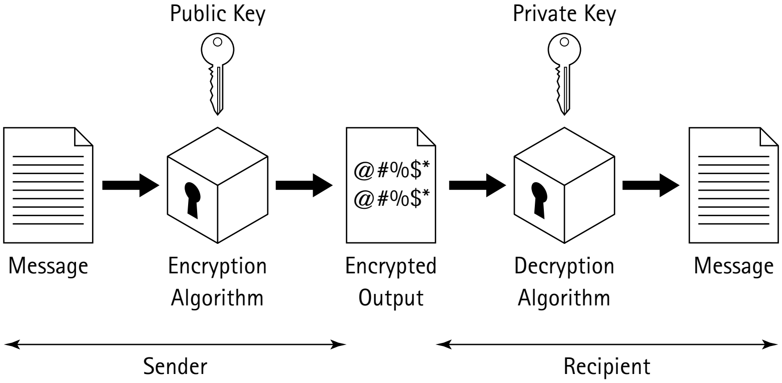

Figure 1-12: Using public key encryption to provide confidentiality

ENSURING CONFIDENTIALITY AND AUTHENTICITY BY USING PUBLIC KEY ENCRYPTION

To provide authentication, Alice must encrypt the message with her private key and Bob will decrypt the message with Alice's public key. This method will provide authenticity, but for integrity there should be a system such as FCS. This system could provide authentication that the message is coming from Alice but it does not provide confidentiality, because Alice's public key is known to all. Hence, anybody possessing Alice's public key can decrypt the message.

To provide both confidentiality and authenticity, Alice will need to encrypt the message first with her private key, which will provide authenticity. Then, she will use Bob's public key to encrypt the message, which will provide confidentiality. Figure 1-13 explains this process.

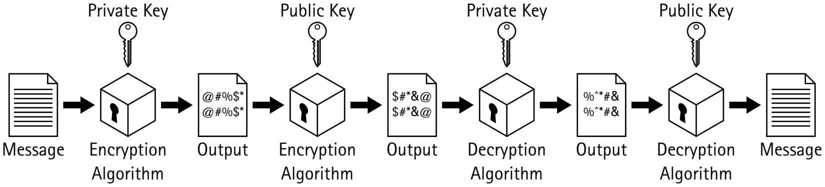

Figure 1-13: Using public key encryption to provide confidentiality and authentication

The disadvantage of the system is that it will be very time consuming and complex as public key encryption and decryption has to be done four times, and the key length of the public key is large (1024 bits to 4094 bits).

Message Authentication Code

To provide authentication and integrity, an alternative method can be used by making use of a secret key to generate a fixed-size block of data. This fixed-size block of data is called *Message Authentication Code* (MAC).

Let's take an example where Alice wants to communicate with Bob. Both Alice and Bob will share a secret key. When Alice wants to send a message to Bob, she will calculate the MAC of the message using the secret key and will append it to the message. When Bob receives the message he will use the shared secret key to generate the MAC of the message, and if both the appended MAC and the generated MAC match, both will be sure of the integrity of the message, as well as the authenticity of the message, as only Bob and Alice know the key. Figure 1-14 explains this process.

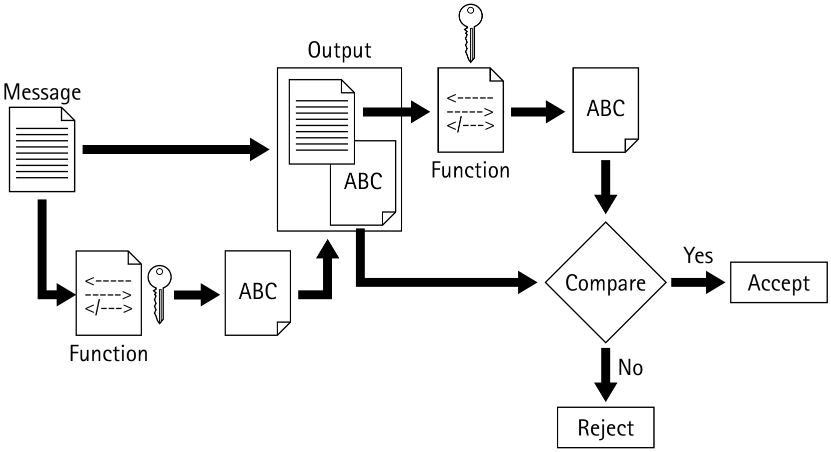

Figure 1-14: Providing authenticity and integrity using MAC

The only difference between MAC and message encryption is that MAC can only be a one-way function, which is not reversible. Once MAC has been generated, the original message can't be regenerated back from the MAC.

The process mentioned above does provide authenticity and integrity but does not provide confidentiality. To provide confidentiality, Alice needs to encrypt the message. The MAC can be appended to the message before encryption. Figure 1-15 displays this process.

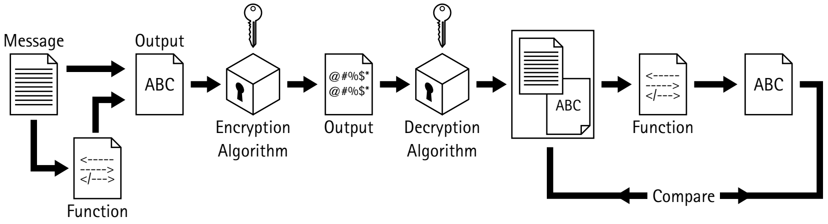

Figure 1-15: Providing authentication, integrity, and confidentiality using MAC

The MAC can also be appended to the message after encryption. In this case, the MAC will be generated by using the ciphertext and not with the original message. Figure 1-16 explains this process.

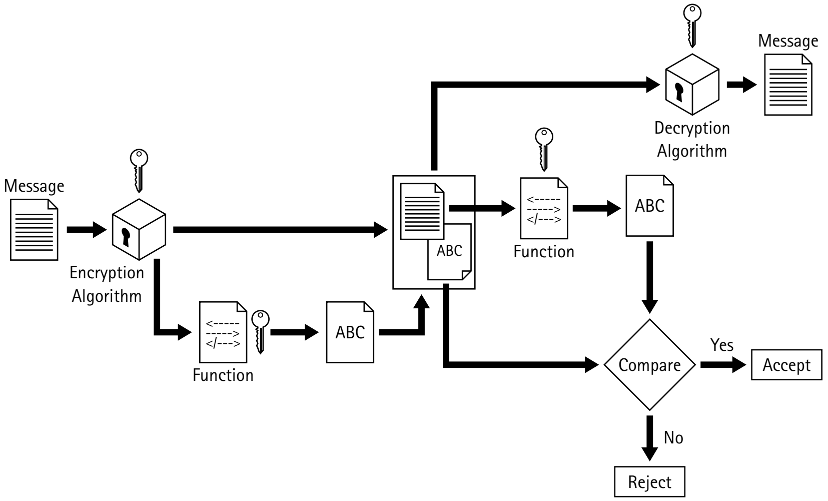

Figure 1-16: Appending the MAC to the message

Hash Functions

A hash function is a variation of the message authentication code. A *hash function*, H, is a conversion method that takes an input *m*, which is the message, and returns a fixed-size string, which is called the *hash value h* (that is, $h = H(m)$) or message digest. This output is fixed in size and is irreversible, which means that the original content can never be recovered. The hash function output could be *weakly collision free*, which means that there is a very rare chance that a similar output could be produced by another message. The output could also be *strongly collision free*, which means that a similar output can never be produced by another message.

If any two hash functions produce the same set of hash values at any time, it is termed as a *collision*. A hash function is considered to be up to the standard, only if the risk of collision is minimal.

Hash functions are normally used to provide the digital fingerprints of files to ensure that the content of the file has not been altered in transit.

There are various ways how hash functions can be used in communication between two individuals. Let us take an example to explain this communication process.

Alice wants to send a message to Bob; Alice will append the hash value of the message with the message and encrypt the message with the secret key. This will provide authenticity, because only Alice and Bob know about the secret key, and encryption is used to provide confidentiality to the message. Figure 1-17 displays this process.

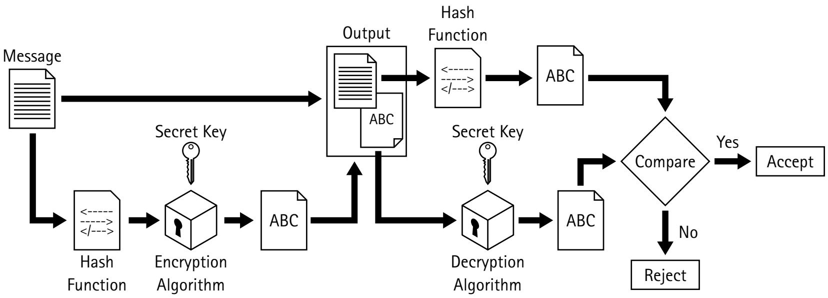

Figure 1-17: Providing authenticity and confidentiality

Alice will encrypt the message digest or the hash value by using her private key. This will generate Alice's digital signature, because only Alice can provide the encrypted hash value. Figure 1-18 explains this process.

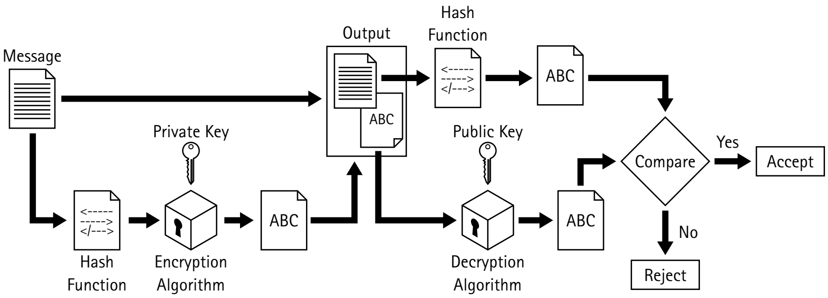

Figure 1-18: Encrypting a message by using the private key

Let's take an example, when Alice wants to send a message to Bob. Bob should know that the message is coming from Alice. Thus, Alice will append her digital signature to the message and encrypt the entire message by using the conventional secret key. Bob will use the corresponding key to decrypt the message. Figure 1-19 explains this process.

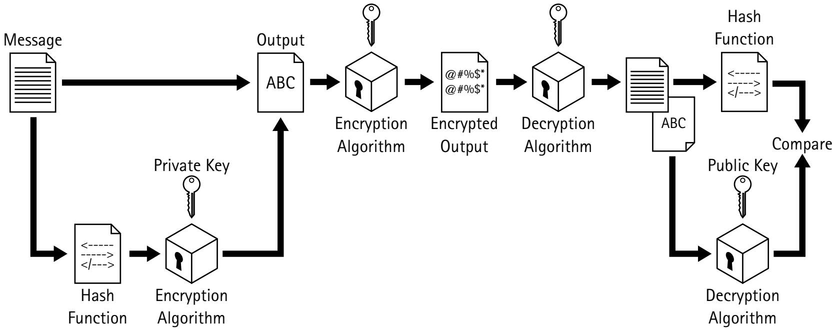

Figure 1-19: Providing integrity, authentication, and confidentiality

There are several hash functions available. The description of some of the most commonly used hash functions is given below:

- **Secure Hash Algorithm (SHA-1):** Also known as *Secure Hash Standard* (*SHS*), this hash algorithm was published by the United States government. This algorithm can produce an output of a 160-bit hash value. This algorithm has been well taken and appreciated by experts.
- **MD2, MD4:** These algorithms were released by RSA Data Security Inc. Several security leakages have been discovered in these algorithms, and they are no longer used to implement encryption. Newer algorithms like MD5 have been developed.
- **MD5:** This algorithm was also released by RSA Laboratories. This algorithm can produce an output of a 128-bit hash value. As in the case of MD4, some security loopholes have been found in MD5 too.
- **RIPEMD-160:** This hash algorithm was designed to replace MD4 and MD5 and provide better and safer hashing methodology. It can produce a 20 bytes or 160 bits message digest.

When using algorithms to create encrypted hash values, you need to ensure that you keep track of the input string and enter an appropriate input string. This is because a small change in the input characters can cause a major bit-shift on the entire output string. A shift of 1 bit in the input string will cause a shift of about half of the total bits in the resulting string. This is called the *avalanche effect*.

Digital Signatures

Any process of authentication protects two parties against a third party. However, this process does not protect the parties against each other. This means that in situations where there isn't complete trust between the sender and the recipient, something more than authentication is required. This problem can be solved using a digital signature. A digital signature is analogous to a handwritten signature and verifies the author, date, and time of signature. The signature should also be able to authenticate the content at the time of the signature. The main requirements of a digital signature are:

- It is unique to the sender.
- It should be recognizable and verifiable.

There are a variety of approaches for digital signatures, which fall broadly into two categories—*direct* and *arbitrated.*

Direct Digital Signatures

A direct digital signature can be formed by encrypting the entire message with the sender's private key or by encrypting a hash value of the message with the sender's private key. Figure 1-20 explains the process of creating digital signatures.

The output is called a digital signature and is attached to the message. To verify the signature, the recipient does a computation involving the message, the signature, and the sender's public key. If the result conforms, the signature is considered to be authentic. Otherwise, the signature is considered either to be a fake or the message has been tampered with. This is because the computed value is based on the signature and the contents of the message. Any change in the values of the digital signature or the contents of the message results in a mismatch between the computed value and the value that is received. This indicates that either the signatures have been faked or the message contents have been modified.

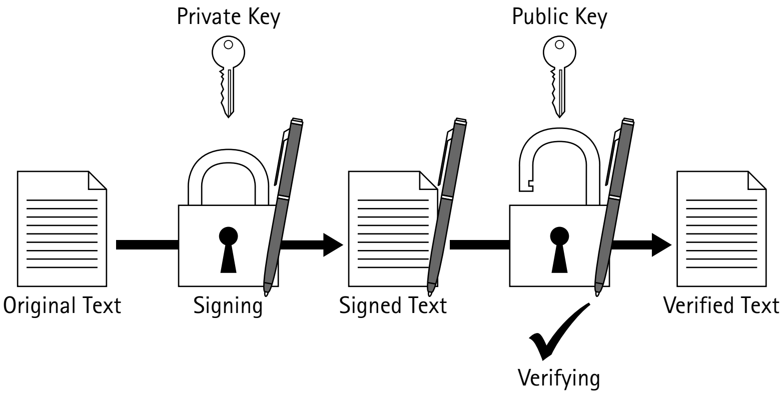

Figure 1-20: Digital signatures

Further encrypting the entire message plus the digital signature can provide confidentiality. It is important to add the digital signature to the message and then to encrypt the entire message. Rather than encrypting the message first, the digital signature must be calculated and added to the signature. If the latter approach is taken, then a third party needs to access the decryption method to read the message. Otherwise, only plaintext and the digital signature can be kept for future dispute resolutions.

This direct digital signature scheme has a single drawback—the entire scheme depends on the validity of the sender's private key. If the sender disowns the responsibility that he has sent the message and claims that private key is lost or compromised then somebody must have forged the signature.

Arbitrated Digital Signature

Arbitrated digital signature scheme is used to overcome the problem of non-repudiation encountered in a direct digital signature. In this scheme, every signed message from the sender, which has been sent to the recipient, first goes to an arbitrator who checks the signature about its origin and content. The message is then dated and sent to the recipient. The presence of the arbitrator solves the problem of sender disowning the signature. For example, when Alice sends a digitally signed message to Bob, an arbitrator first validates Alice's signature. After the signature has been validated, the message is then sent to Bob along with the date of validation and notice that the signature does belong to Alice.

How Does a Digital Signature Work?

The manner in which a digital signature works is quite simple.

Let's suppose that you want to send important documents to your business partner, who is out-of-town. After you send the documents, you need to assure your partner that the documents have not been modified and are not different from the ones that you sent, and that you actually own them. To ensure the authenticity of the documents that you are sending in an e-mail message, you need to get a hash for your document and then encrypt the hash by using the private key from the key pair that you have obtained from an authority. So where's your digital signature? The hash that you encrypted by using the key is your digital signature. In this way, the hash function is converted to a digital signature and an e-mail that you can send to the receiver. Each time that you create a digital signature for a message, your digital signature will be different because a different hash has been created each time.

Now let's look at the recipient's side.

The message reaches your business partner. How does he verify that it is a valid and authentic document? Your business partner will first create a hash for the message. Then he will decrypt the message hash that you sent. How will he do it? He will use the public key to decrypt it. Finally, he needs to match the hash you sent with the hash that was created at his end. If the two match, it is proof that your message is a valid one.

There are several standard algorithms that have been developed for creating digital signatures. One of them is Digital Signature Standard (DSS) developed by the U.S. National Security Agency (NSA) in 1994. It has been used to generate digital signatures for electronic documents.

Summary

In this chapter, you learned about the various techniques that are used to encrypt data to prevent it from being violated during transit. You learned how cryptography provides the means and methods of hiding data, establishing its authenticity, and preventing its undetected modification or unauthorized use. You learned that there are two types of cryptography:

- Symmetric cryptography, which uses one single key to encrypt as well as decrypt data. DES, 3DES, IDEA, RC2, RC4, RC5, CAST-128, and AES are various algorithms that are used in symmetric cryptography.
- Asymmetric cryptography, which uses a pair of keys — public and private keys — for data encryption and decryption. Asymmetric cryptography is based on the RSA algorithm. RSA is one of the most powerful encryption/decryption algorithms available today.

Next, you learned about the various applications of cryptography, which include

- Message encryption
- Message Authentication Code
- Hash functions

Message encryption allows the encryption of data using symmetric as well as asymmetric encryption mechanisms. Message Authentication Code, on the other hand, is an irreversible encryption method that uses a secret key to generate fixed-sized data blocks. Hash functions are a variation of MAC and allow strong collision-free output.

Finally, you learned about the role and use of digital signatures in modern encryption/decryption mechanisms. You learned that digital signatures work exceptionally well between entities that do not trust each other. Therefore, digital signatures have emerged as the most common method of data authentication over that most untrustworthy of mediums—the Internet.