Appendix K
Cryptographic Algorithms

William Stallings

Supplement to
Operating Systems, Eighth Edition
Pearson 2014
http://williamstallings.com/OperatingSystems/

The essential technology underlying virtually all automated network and computer security applications is cryptography. Two fundamental approaches are in use: symmetric encryption, also known as conventional encryption, and public-key encryption, also known as asymmetric encryption. This appendix provides an overview of both types of encryption, together with a brief discussion of some important encryption algorithms.

K.1 SYMMETRIC ENCRYPTION

Symmetric encryption was the only type of encryption in use prior to the introduction of public-key encryption in the late 1970s. Symmetric encryption has been used for secret communication by countless individuals and groups, from Julius Caesar to the German U-boat force to present-day diplomatic, military, and commercial users. It remains by far the more widely used of the two types of encryption.

A symmetric encryption scheme has five ingredients (Figure K.1):

- **Plaintext:** This is the original message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- **Secret key:** The secret key is also input to the encryption algorithm. The exact substitutions and transformations performed by the algorithm depend on the key.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts.

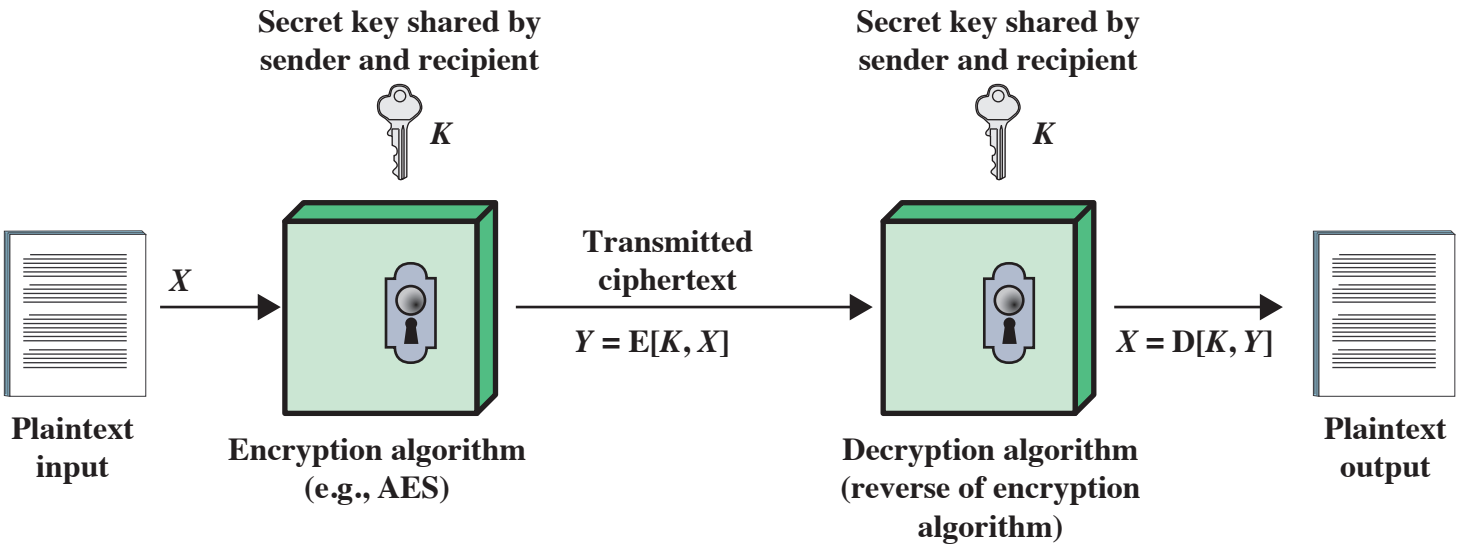


Figure K.1 Simplified Model of Symmetric Encryption

- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

There are two requirements for secure use of symmetric encryption:

1. We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.
2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

There are two general approaches to attacking a symmetric encryption scheme. The first attack is known as **cryptanalysis**. Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext-ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used. If the attack succeeds in deducing the key, the effect is catastrophic: All future and past messages encrypted with that key are compromised.

The second method, known as the **brute-force** attack, is to try every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success. Table K.1 shows how much time is involved for various key sizes. The table shows results for each key size, assuming that it takes 1 μs to perform a single decryption, a reasonable order of magnitude for today's computers. With the use of massively parallel organizations of microprocessors, it may be possible to achieve processing rates many orders of magnitude greater. The final column of the table considers the results for a system that can process 1 million keys per microsecond. As one can see, at this performance level, a 56-bit key can no longer be considered computationally secure.

Table K.1 Average Time Required for Exhaustive Key Search

Key Size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/μs	Time Required at 10^6 Decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\ \mu s = 35.8$ minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\ \mu s = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\ \mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\ \mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\ \mu s = 6.4 \times 10^{12}$ years	6.4×10^6 years

The most commonly used symmetric encryption algorithms are block ciphers. A block cipher processes the plaintext input in fixed-size blocks and produces a block of ciphertext of equal size for each plaintext block. The two most important symmetric algorithms, both of which are block ciphers, are the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES).

The Data Encryption Standard (DES)

DES has been the dominant encryption algorithm since its introduction in 1977. However, because DES uses only a 56-bit key, it was only a matter of time before computer processing speed made DES obsolete. In 1998, the Electronic Frontier Foundation (EFF) announced that it had broken a DES challenge using a special-purpose "DES cracker" machine that was built for less than $250,000. The attack took less than three days. The EFF has published a detailed description of the machine, enabling others to build their own cracker. And, of course, hardware prices continue to drop as speeds increase, making DES worthless.

The life of DES was extended by the use of triple DES (3DES), which involves repeating the basic DES algorithm three times, using either two or three unique keys, for a key size of 112 or 168 bits.

The principal drawback of 3DES is that the algorithm is relatively sluggish in software. A secondary drawback is that both DES and 3DES use a 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable.

Advanced Encryption Standard

Because of these drawbacks, 3DES is not a reasonable candidate for long-term use. As a replacement, the National Institute of Standards and

Technology (NIST) in 1997 issued a call for proposals for a new Advanced Encryption Standard (AES), which should have a security strength equal to or better than 3DES and significantly improved efficiency. In addition to these general requirements, NIST specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits. Evaluation criteria include security, computational efficiency, memory requirements, hardware and software suitability, and flexibility. In 2001, NIST issued AES as a federal information processing standard (FIPS 197).

K.2 PUBLIC-KEY CRYPTOGRAPHY

Public-key encryption, first publicly proposed by Diffie and Hellman in 1976, is the first truly revolutionary advance in encryption in literally thousands of years. For one thing, public-key algorithms are based on mathematical functions rather than on simple operations on bit patterns. More important, public-key cryptography is asymmetric, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key. The use of two keys has profound consequences in the areas of confidentiality, key distribution, and authentication.

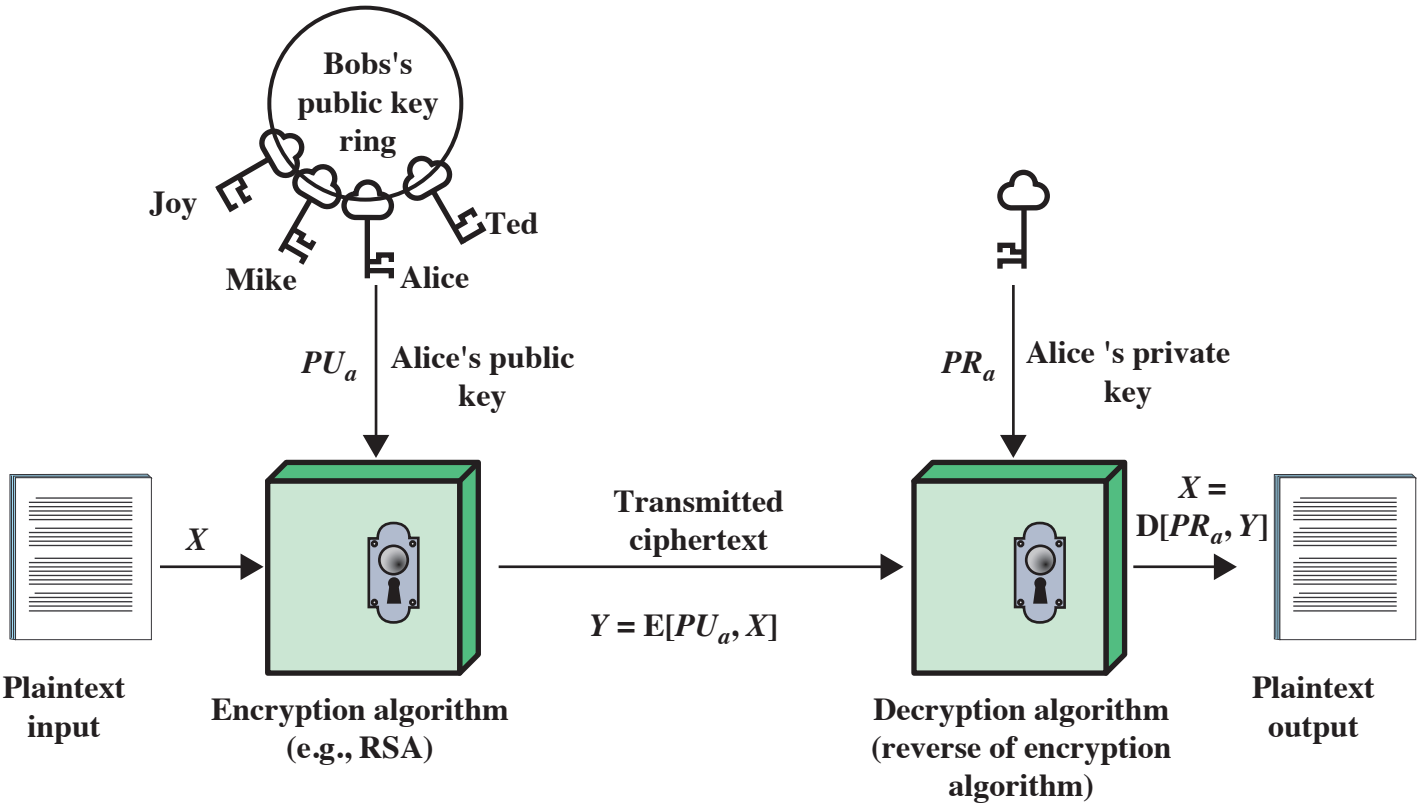
Before proceeding, we should first mention several common misconceptions concerning public-key encryption. One is that public-key encryption is more secure from cryptanalysis than symmetric encryption. In fact, the security of any encryption scheme depends on the length of the key and the computational work involved in breaking a cipher. There is nothing in principle about either symmetric or public-key encryption that makes one superior to another from the point of view of resisting cryptanalysis. A second misconception is that public-key encryption is a general-purpose technique that has made symmetric encryption obsolete. On the contrary,

because of the computational overhead of current public-key encryption schemes, there seems no foreseeable likelihood that symmetric encryption will be abandoned. Finally, there is a feeling that key distribution is trivial when using public-key encryption, compared to the rather cumbersome handshaking involved with key distribution centers for symmetric encryption. In fact, some form of protocol is needed, often involving a central agent, and the procedures involved are no simpler or any more efficient than those required for symmetric encryption.

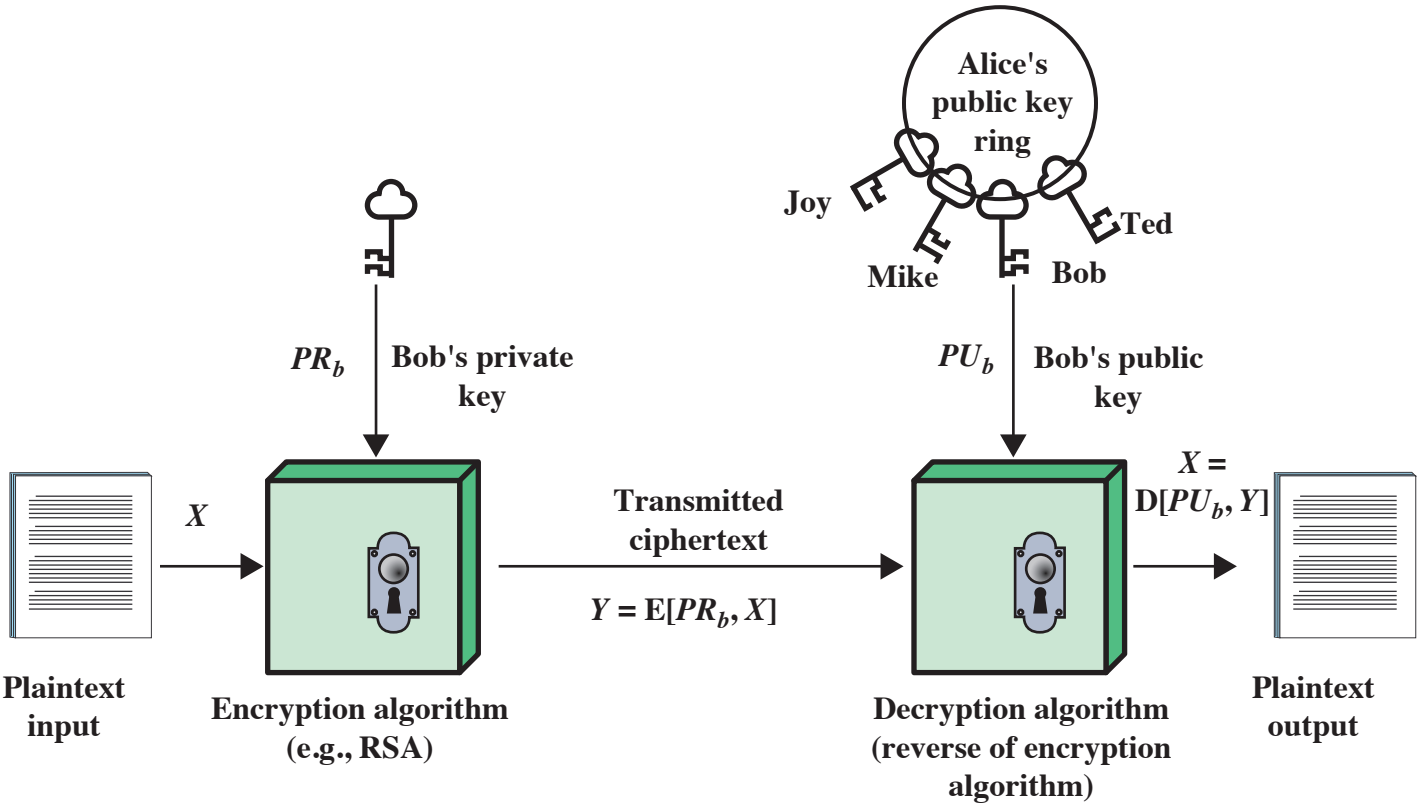
A public-key encryption scheme has six ingredients (Figure K.2):

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private key:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

The process works (produces the correct plaintext on output) regardless of the order in which the pair of keys is used. As the names suggest, the public key of the pair is made public for others to use, while the private key is known only to its owner.



(a) Encryption with public key



(b) Encryption with private key

Figure K.2 Public-Key Cryptography

Now, say that Bob wants to send a private message to Alice and suppose that he has Alice's public key and Alice has the matching private key (Figure K.2a). Using Alice's public key, Bob encrypts the message to produce ciphertext. The ciphertext is then transmitted to Alice. When Alice gets the ciphertext, she decrypts it using her private key. Because only Alice has a copy of her private key, no one else can read the message.

Public-key encryption can be used in another way, as illustrated in Figure K.2b. Suppose that Bob wants to send a message to Alice and, although it isn't important that the message be kept secret, he wants Alice to be certain that the message is indeed from him. In this case Bob uses his own private key to encrypt the message. When Alice receives the ciphertext, she finds that she can decrypt it with Bob's public key, thus proving that the message must have been encrypted by Bob: No one else has Bob's private key and therefore no one else could have created a ciphertext that could be decrypted with Bob's public key.

A general-purpose public-key cryptographic algorithm relies on one key for encryption and a different but related key for decryption. Furthermore, these algorithms have the following important characteristics:

- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
- Either of the two related keys can be used for encryption, with the other used for decryption.

The essential steps are the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.

2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure K.2a suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user protects his or her private key, incoming communication is secure. At any time, a user can change the private key and publish the companion public key to replace the old public key.

The key used in symmetric encryption is typically referred to as a **secret key**. The two keys used for public-key encryption are referred to as the **public key** and the **private key**. Invariably, the private key is kept secret, but it is referred to as a private key rather than a secret key to avoid confusion with symmetric encryption.

Rivest-Shamir-Adleman (RSA) Algorithm

One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT. The RSA scheme has since that time reigned supreme as the only widely accepted and implemented approach to public-key encryption. RSA is a cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n. Encryption involves modular

arithmetic. The strength of the algorithm is based on the difficulty of factoring numbers into their prime factors.

K.3 MESSAGE AUTHENTICATION AND HASH FUNCTIONS

Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication.

A message, file, document, or other collection of data is said to be authentic when it is genuine and came from its alleged source. Message authentication is a procedure that allows communicating parties to verify that received messages are authentic. The two important aspects are to verify that the contents of the message have not been altered and that the source is authentic. We may also wish to verify a message's timeliness (it has not been artificially delayed and replayed) and sequence relative to other messages flowing between two parties.

Authentication Using Symmetric Encryption

It is possible to perform authentication simply by the use of symmetric encryption. If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able successfully to encrypt a message for the other participant. Furthermore, if the message includes an error-detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit.

Message Authentication without Message Encryption

In this section, we examine several approaches to message authentication that do not rely on message encryption. In all of these approaches, an authentication tag is generated and appended to each message for transmission. The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.

Because the approaches discussed in this section do not encrypt the message, message confidentiality is not provided. Because symmetric encryption will provide authentication, and because it is widely used with readily available products, why not simply use such an approach, which provides both confidentiality and authentication? Here are three situations in which message authentication without confidentiality is preferable:

1. There are a number of applications in which the same message is broadcast to a number of destinations. An example is the notification to users that the network is now unavailable or an alarm signal in a control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with an associated message authentication tag. The responsible system performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm.
2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, with messages chosen at random for checking.
3. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources.

However, if a message authentication tag were attached to the program, it could be checked whenever assurance is required of the integrity of the program.

Thus, there is a place for both authentication and encryption in meeting security requirements.

Message Authentication Code

One authentication technique involves the use of a secret key to generate a small block of data, known as a message authentication code, that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K_{AB}. When A has a message M to send to B, it calculates the message authentication code as a function of the message and the key: $\text{MAC}_M = \text{F}(K_{AB}, M)$. The message plus code are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new message authentication code. The received code is compared to the calculated code (Figure K.3). If we assume that only the receiver and the sender know the identity of the secret key, and if the received code matches the calculated code, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code. Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond to the alterations in the message.

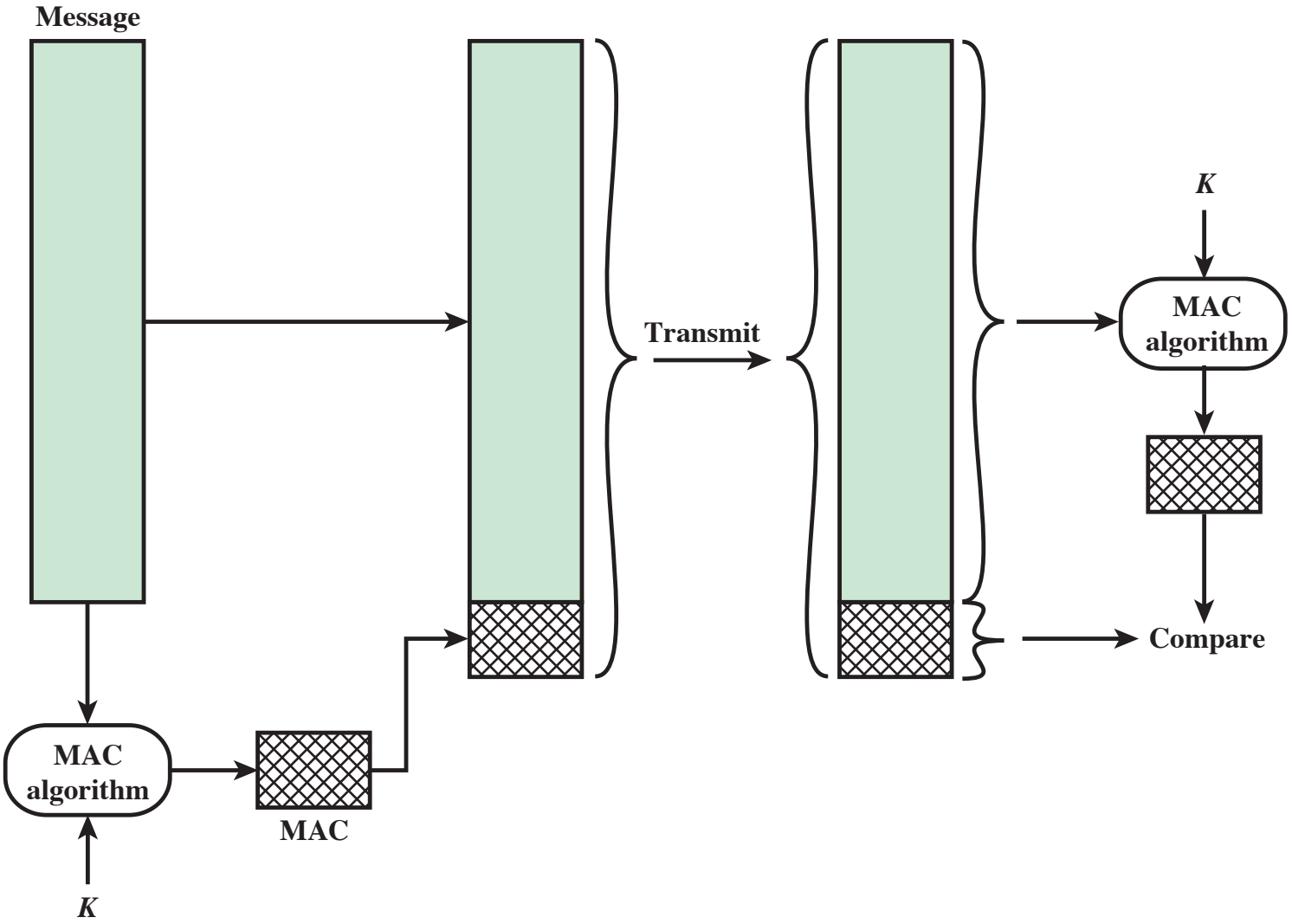


Figure K.3 Message Authentication Using a Message Authentication Code (MAC)

2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper code.
3. If the message includes a sequence number (such as is used with X.25, HDLC, and TCP), then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

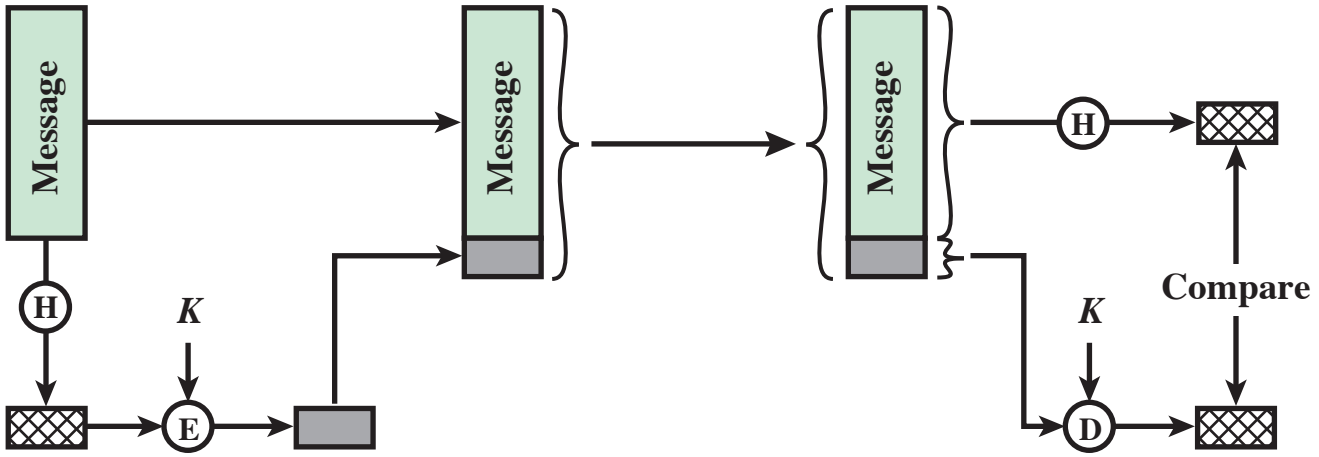
A number of algorithms could be used to generate the code. The National Bureau of Standards, in its publication *DES Modes of Operation*, recommends the use of DES. DES is used to generate an encrypted version of the message, and the last number of bits of ciphertext are used as the code. A 16- or 32-bit code is typical.

The process just described is similar to encryption. One difference is that the authentication algorithm need not be reversible, as it must for decryption. It turns out that because of the mathematical properties of the authentication function, it is less vulnerable to being broken than encryption.

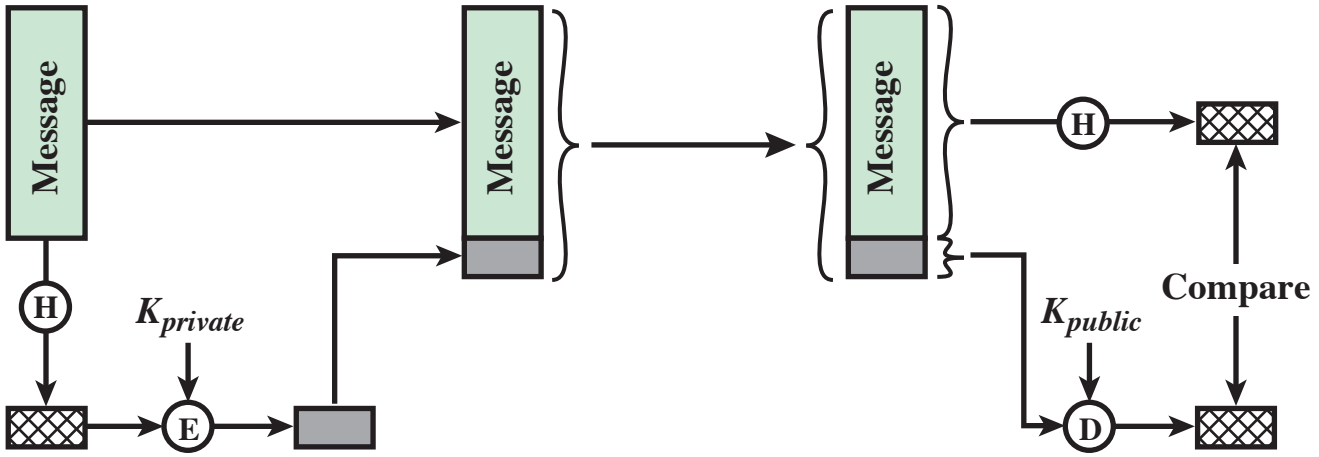
One-Way Hash Function

A variation on the message authentication code that has received much attention is the one-way hash function. As with the message authentication code, a hash function accepts a variable-size message *M* as input and produces a fixed-size message digest H(*M*) as output. Unlike the MAC, a hash function does not also take a secret key as input. To authenticate a message, the message digest is sent with the message in such a way that the message digest is authentic.

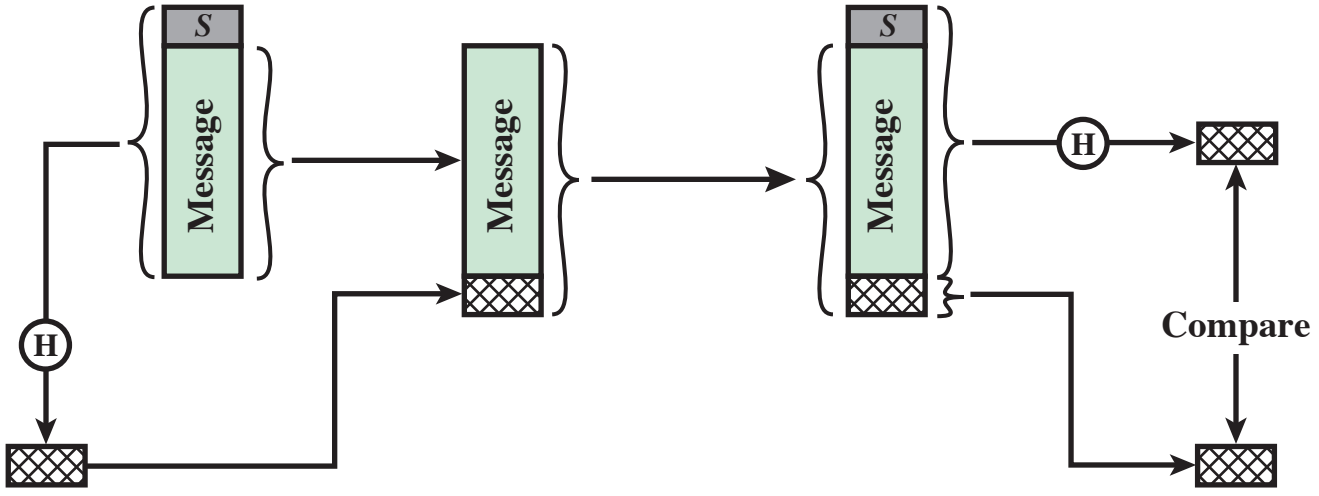
Figure K.4 illustrates three ways in which the message can be authenticated. The message digest can be encrypted using symmetric encryption (part a); if it is assumed that only the sender and receiver share



(a) Using conventional encryption



(b) Using public-key encryption



(c) Using secret value

Figure K.4 Message Authentication Using a One-Way Hash Function

the encryption key, then authenticity is assured. The message digest can also be encrypted using public-key encryption (part b). The public-key approach has two advantages: it provides a digital signature as well as message authentication, and it does not require the distribution of keys to communicating parties.

These two approaches have an advantage over approaches that encrypt the entire message in that less computation is required. Nevertheless, there has been interest in developing a technique that avoids encryption altogether. Several reasons for this interest:

- Encryption software is somewhat slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.
- Encryption hardware costs are nonnegligible. Low-cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.
- Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.
- Encryption algorithms may be covered by patents and must be licensed, adding a cost.
- Encryption algorithms may be subject to export control.

Figure K.4c shows a technique that uses a hash function but no encryption for message authentication. This technique assumes that two

communicating parties, say A and B, share a common secret value S_{AB}. When A has a message to send to B, it calculates the hash function over the concatenation of the secret value and the message: $MD_M = H(S_{AB}||M)$.[1] It then sends $[M||MD_M]$ to B. Because B possesses S_{AB}, it can recompute $H(S_{AB}||M)$ and verify MD_M. Because the secret value itself is not sent, it is not possible for an attacker to modify an intercepted message. As long as the secret value remains secret, it is also not possible for an attacker to generate a false message.

This third technique, using a shared secret value, is the one adopted for IP security; it has also been specified for SNMPv3.

K.4 SECURE HASH FUNCTIONS

An essential element of many security services and applications is a secure hash function. A hash function accepts a variable-size message *M* as input and produces a fixed-size tag H(*M*), sometimes called a message digest, as output. For a digital signature, a hash code is generated for a message, encrypted with the sender's private key, and sent with the message. The receiver computes a new hash code for the incoming message, decrypts the hash code with the sender's public key and compares. If the message has been altered in transit, there will be a mismatch.

To be useful for security applications, a hash function H must have the following properties:

1. H can be applied to a block of data of any size.

2. H produces a fixed-length output.

---

[1] || denotes concatenation.

3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the **one-way property**.
5. For any given block x, it is computationally infeasible to find $y \neq x$ such that H(y) = H(x). This is sometimes referred to as **weak collision resistance**.
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance**.

In recent years, the most widely used hash function has been the Secure Hash Algorithm (SHA). SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993. When weaknesses were discovered in SHA, a revised version was issued as FIPS 180-1 in 1995 and is generally referred to as SHA-1. SHA-1 produces a hash value of 160 bits. In 2002, NIST produced a revised version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512. These new versions have the same underlying structure and use the same types of modular arithmetic and logical binary operations as SHA-1. In 2005, NIST announced the intention to phase out approval of SHA-1 and move to a reliance on the other SHA versions by 2010. Researchers have demonstrated that SHA-1 is far weaker than its 160-bit hash length suggests, necessitating the move to the newer versions of SHA.