Chapter 6
Introduction to Public-Key Cryptography

Before we learn about the basics of public-key cryptography, let us recall that the term *public-key cryptography* is used interchangeably with *asymmetric cryptography*; they both denote exactly the same thing and are used synonymously.

As stated in Chap. 1, symmetric cryptography has been used for at least 4000 years. Public-key cryptography, on the other hand, is quite new. It was publicly introduced by Whitfield Diffie, Martin Hellman and Ralph Merkle in 1976. Much more recently, in 1997 British documents which were declassified revealed that the researchers James Ellis, Clifford Cocks and Graham Williamson from the UK's Government Communications Headquarters (GCHQ) discovered and realized the principle of public-key cryptography a few years earlier, in 1972. However, it is still being debated whether the government office fully recognized the far-reaching consequences of public-key cryptography for commercial security applications.

In this chapter you will learn:

- A brief history of public-key cryptography
- The pros and cons of public-key cryptography
- Some number theoretical topics that are needed for understanding public-key algorithms, most importantly the extended Euclidean algorithm

6.1 Symmetric vs. Asymmetric Cryptography

In this chapter we will see that asymmetric, i.e., public-key, algorithms are very different from symmetric algorithms such as AES or DES. Most public-key algorithms are based on number-theoretic functions. This is quite different from symmetric ciphers, where the goal is usually *not* to have a compact mathematical description between input and output. Even though mathematical structures are often used for small blocks *within* symmetric ciphers, for instance, in the AES S-Box, this does not mean that the entire cipher forms a compact mathematical description.

Symmetric Cryptography Revisited

In order to understand the principle of asymmetric cryptography, let us first recall the basic symmetric encryption scheme in Fig. 6.1.

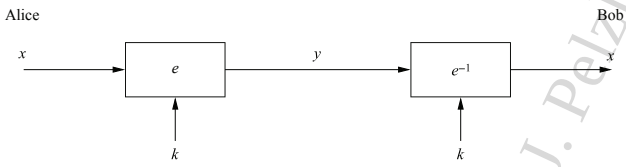

Fig. 6.1 Principle of symmetric-key encryption

Such a system is symmetric with respect to two properties:

1. The *same secret key* is used for encryption and decryption.
2. The encryption and decryption *function* are very similar (in the case of DES they are essentially identical).

There is a simple analogy for symmetric cryptography, as shown in Fig. 6.2. Assume there is a safe with a strong lock. Only Alice and Bob have a copy of the key for the lock. The action of encrypting of a message can be viewed as putting the message in the safe. In order to read, i.e., decrypt, the message, Bob uses his key and opens the safe.

Modern symmetric algorithms such as AES or 3DES are very secure, fast and are in widespread use. However, there are several shortcomings associated with symmetric-key schemes, as discussed below.

Key Distribution Problem The key must be established between Alice and Bob using a secure channel. Remember that the communication link for the message is not secure, so sending the key over the channel directly — which would be the most convenient way of transporting it — can't be done.

Number of Keys Even if we solve the key distribution problem, we must potentially deal with a very large number of keys. If each pair of users needs a separate pair of keys in a network with n users, there are

Fig. 6.2 Analogy for symmetric encryption: a safe with one lock

$$\frac{n \cdot (n-1)}{2}$$

key pairs, and every user has to store $n-1$ keys securely. Even for mid-size networks, say, a corporation with 2000 people, this requires more than 4 million key pairs that must be generated and transported via secure channels. More about this problem is found in Sect. 13.1.3. (There are smarter ways of dealing with keys in symmetric cryptography networks as detailed in Sect. 13.2; however, those approaches have other problems such as a single point of failure.)

No Protection Against Cheating by Alice or Bob Alice and Bob have the same capabilities, since they possess the same key. As a consequence, symmetric cryptography cannot be used for applications where we would like to prevent cheating by either Alice or Bob as opposed to cheating by an outsider like Oscar. For instance, in e-commerce applications it is often important to prove that Alice actually sent a certain message, say, an online order for a flat screen TV. If we only use symmetric cryptography and Alice changes her mind later, she can always claim that Bob, the vendor, has falsely generated the electronic purchase order. Preventing this is called *nonrepudiation* and can be achieved with asymmetric cryptography, as discussed in Sect. 10.1.1. Digital signatures, which are introduced in Chap. 10, provide nonrepudiation.

Fig. 6.3 Analogy for public-key encryption: a safe with public lock for depositing a message and a secret lock for retrieving a message

Principles of Asymmetric Cryptography

In order to overcome these drawbacks, Diffie, Hellman and Merkle had a revolutionary proposal based on the following idea: It is not necessary that the key possessed by the person who *encrypts* the message (that's Alice in our example) is secret. The crucial part is that Bob, the receiver, can only *decrypt* using a secret key. In order to realize such a system, Bob publishes a public encryption key which is known to everyone. Bob also has a matching secret key, which is used for decryption. Thus, Bob's key k consists of two parts, a public part, k_{pub}, and a private one, k_{pr}.

A simple analogy of such a system is shown in Fig. 6.3. This systems works quite similarly to the good old mailbox on the corner of a street: Everyone can put a letter in the box, i.e., encrypt, but only a person with a private (secret) key can retrieve letters, i.e., decrypt. If we assume we have cryptosystems with such a functionality, a basic protocol for public-key encryption looks as shown in Fig. 6.4.

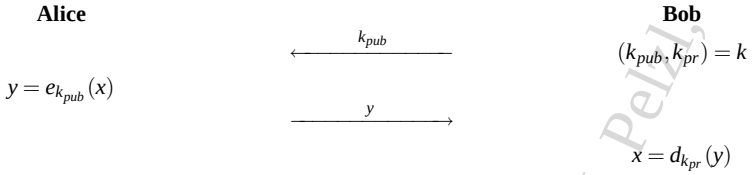

Fig. 6.4 Basic protocol for public-key encryption

By looking at that protocol you might argue that even though we can encrypt a message without a secret channel for key establishment, we still cannot exchange a key if we want to encrypt with, say, AES. However, the protocol can easily be modified for this use. What we have to do is to *encrypt a symmetric key*, e.g., an AES key, using the public-key algorithm. Once the symmetric key has been decrypted by Bob, both parties can use it to encrypt and decrypt messages using symmetric ciphers. Figure 6.5 shows a basic key transport protocol where we use AES as the symmetric cipher for illustration purposes (of course, one can use any other symmetric algorithm in such a protocol). The main advantage of the protocol in Fig. 6.5 over the protocol in Fig. 6.4 is that the payload is encrypted with a symmetric cipher, which tends to be much faster than an asymmetric algorithm.

From the discussion so far, it looks as though asymmetric cryptography is a desirable tool for security applications. The question remains how one can build public-key algorithms. In Chaps. 7, 8 and 9 we introduce most asymmetric schemes of practical relevance. They are all built from one common principle, the one-way function. The informal definition of it is as follows:

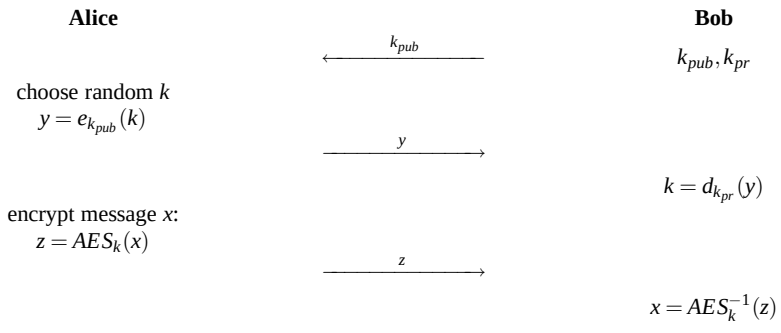

Fig. 6.5 Basic key transport protocol with AES as an example of a symmetric cipher

Definition 6.1.1 One-way function
A function $f()$ is a one-way function if:

1. *$y = f(x)$ is computationally easy, and*
2. *$x = f^{-1}(y)$ is computationally infeasible.*

Obviously, the adjectives "easy" and "infeasible" are not particularly exact. In mathematical terms, a function is easy to compute if it can be evaluated in polynomial time, i.e., its running time is a polynomial expression. In order to be useful in practical crypto schemes, the computation $y = f(x)$ should be sufficiently fast that it does not lead to unacceptably slow execution times in an application. The inverse computation $x = f^{-1}(y)$ should be so computationally intensive that it is not feasible to evaluate it in any reasonable time period, say, 10,000 years, when using the best known algorithm.

There are two popular one-way functions which are used in practical public-key schemes. The first is the integer factorization problem, on which RSA is based. Given two large primes, it is easy to compute the product. However, it is very difficult to factor the resulting product. In fact, if each of the primes has 150 or more decimal digits, the resulting product cannot be factored, even with thousands of PCs running for many years. The other one-way function that is used widely is the discrete logarithm problem. This is not quite as intuitive and is introduced in Chap. 8.

6.2 Practical Aspects of Public-Key Cryptography

Actual public-key algorithms will be introduced in the next chapters, since there is some mathematics we must study first. However, it is very interesting to look at the principal security functions of public-key cryptography which we address in this section.