Chapter 8
Public-Key Cryptosystems Based on the Discrete Logarithm Problem

In the previous chapter we learned about the RSA public-key scheme. As we have seen, RSA is based on the hardness of factoring large integers. The integer factorization problem is said to be the *one-way function* of RSA. As we saw earlier, roughly speaking a function is *one-way* if it is computationally easy to compute the function $f(x) = y$, but computationally infeasible to invert the function: $f^{-1}(y) = x$. The question is whether we can find other one-way functions for building asymmetric crypto schemes. It turns out that most non-RSA public-key algorithms with practical relevance are based on another one-way function, the discrete logarithm problem.

In this chapter you will learn:

- The Diffie–Hellman key exchange
- Cyclic groups which are important for a deeper understanding of Diffie–Hellman key exchange
- The discrete logarithm problem, which is of fundamental importance for many practical public-key algorithms
- Encryption using the Elgamal scheme

The security of many cryptographic schemes relies on the computational intractability of finding solutions to the *Discrete Logarithm Problem (DLP)*. Well-known examples of such schemes are the Diffie–Hellman key exchange and the Elgamal encryption scheme, both of which will be introduced in this chapter. Also, the Elgamal digital signature scheme (cf. Section 8.5.1) and the digital signature algorithm (cf. Section 10.2) are based on the DLP, as are cryptosystems based on elliptic curves (Section 9.3).

We start with the basic Diffie–Hellman protocol, which is surprisingly simple and powerful. The discrete logarithm problem is defined in what are called *cyclic groups*. The concept of this algebraic structure is introduced in Section 8.2. A formal definition of the DLP as well as some illustrating examples are provided, followed by a brief description of attack algorithms for the DLP. With this knowledge we will revisit the Diffie–Hellman protocol and more formally talk about its security. We will then develop a method for encrypting data using the DLP that is known as the Elgamal cryptosystem.

8.1 Diffie–Hellman Key Exchange

The *Diffie–Hellman key exchange (DHKE)*, proposed by Whitfield Diffie and Martin Hellman in 1976 [58], was the first asymmetric scheme published in the open literature. The two inventors were also influenced by the work of Ralph Merkle. It provides a practical solution to the key distribution problem, i.e., it enables two parties to derive a common secret key by communicating over an insecure channel[1]. The DHKE is a very impressive application of the discrete logarithm problem that we'll study in the subsequent sections. This fundamental key agreement technique is implemented in many open and commercial cryptographic protocols like Secure Shell (SSH), Transport Layer Security (TLS), and Internet Protocol Security (IPSec).The basic idea behind the DHKE is that exponentiation in $\mathbb{Z}_p^*$, p prime, is a one-way function and that exponentiation is commutative, i.e.,

$$k = (\alpha^x)^y \equiv (\alpha^y)^x \bmod p$$

The value $k \equiv (\alpha^x)^y \equiv (\alpha^y)^x \bmod p$ is the joint secret which can be used as the session key between the two parties.

Let us now consider how the Diffie–Hellman key exchange protocol over $\mathbb{Z}_p^*$ works. In this protocol we have two parties, Alice and Bob, who would like to establish a shared secret key. There is possibly a trusted third party that properly chooses the public parameters which are needed for the key exchange. However, it is also possible that Alice or Bob generate the public parameters. Strictly speaking, the DHKE consists of two protocols, the set-up protocol and the main protocol, which performs the actual key exchange. The set-up protocol consists of the following steps:

Diffie–Hellman Set-up

1. Choose a large prime p.
2. Choose an integer $\alpha \in \{2, 3, \ldots, p-2\}$.
3. Publish p and α.

These two values are sometimes referred to as *domain parameters*. If Alice and Bob both know the public parameters p and α computed in the set-up phase, they can generate a joint secret key k with the following key-exchange protocol:

[1] The channel needs to be authenticated, but that will be discussed later in this book.

Diffie–Hellman Key Exchange

Alice		Bob
choose $a = k_{pr,A} \in \{2,\ldots,p-2\}$		choose $b = k_{pr,B} \in \{2,\ldots,p-2\}$
compute $A = k_{pub,A} \equiv \alpha^a \bmod p$		compute $B = k_{pub,B} \equiv \alpha^b \bmod p$
	$\xrightarrow{k_{pub,A}=A}$	
	$\xleftarrow{k_{pub,B}=B}$	
$k_{AB} = k_{pub,B}^{k_{pr,A}} \equiv B^a \bmod p$		$k_{AB} = k_{pub,A}^{k_{pr,B}} \equiv A^b \bmod p$

Here is the proof that this surprisingly simple protocol is correct, i.e., that Alice and Bob in fact compute the same session key k_{AB}.

Proof. Alice computes

$$B^a \equiv (\alpha^b)^a \equiv \alpha^{ab} \bmod p$$

while Bob computes

$$A^b \equiv (\alpha^a)^b \equiv \alpha^{ab} \bmod p$$

and thus Alice and Bob both share the session key $k_{AB} \equiv \alpha^{ab} \bmod p$. The key can now be used to establish a secure communication between Alice and Bob, e.g., by using k_{AB} as key for a symmetric algorithm like AES or 3DES. □

We'll look now at a simple example with small numbers.

Example 8.1. The Diffie–Hellman domain parameters are $p = 29$ and $\alpha = 2$. The protocol proceeds as follows:

Alice		Bob
choose $a = k_{pr,A} = 5$		choose $b = k_{pr,B} = 12$
$A = k_{pub,A} = 2^5 \equiv 3 \bmod 29$		$B = k_{pub,B} = 2^{12} \equiv 7 \bmod 29$
	$\xrightarrow{A=3}$	
	$\xleftarrow{B=7}$	
$k_{AB} = B^a \equiv 7^5 = 16 \bmod 29$		$k_{AB} = A^b = 3^{12} \equiv 16 \bmod 29$

As one can see, both parties compute the value $k_{AB} = 16$, which can be used as a joint secret, e.g., as a session key for symmetric encryption.

◇

The computational aspects of the DHKE are quite similar to those of RSA. During the set-up phase, we generate p using the probabilistic prime-finding algorithms discussed in Section 7.6. As shown in Table 6.1, p should have a similar length as the RSA modulus n, i.e., 1024 or beyond, in order to provide strong security. The integer α needs to have a special property: It should be a primitive element, a topic which we discuss in the following sections. The session key k_{AB} that is being computed in the protocol has the same bit length as p. If we want to use it as a symmetric key for algorithms such as AES, we can simply take the 128 most significant bits. Alternatively, a hash function is sometimes applied to k_{AB} and the output is then used as a symmetric key.