In this chapter, we discuss cryptographic hash functions. They are used, for example, in digital signatures. Throughout this chapter, we assume that \( \Sigma \) is an alphabet.

### 11.1 Hash Functions and Compression Functions

By a hash function, we mean a map

\[
h : \Sigma^* \to \Sigma^n, \quad n \in \mathbb{N}.
\]

Thus, hash functions map arbitrarily long strings to strings of fixed length. They are never injective.

**Example 11.1.1**

The map that sends \( b_1 b_2 \ldots b_k \) in \( \{0, 1\}^* \) to \( b_1 \oplus b_2 \oplus b_3 \oplus \cdots \oplus b_k \) is a hash function. It maps, for example, 01101 to 1. In general, it sends a string \( b \) to 1 if the number of ones in \( b \) is odd and to 0 otherwise.
Cryptographic Hash Functions

Hash functions can be generated using compression functions. A compression function is a map

\[ h : \Sigma^m \rightarrow \Sigma^n, \quad n, m \in \mathbb{N}, \quad m > n. \]

It maps strings of fixed length to strings of shorter length.

**Example 11.1.2**
The map that sends the word \( b_1 b_2 \ldots b_m \in \{0, 1\}^m \) to \( b_1 \oplus b_2 \oplus b_3 \oplus \cdots \oplus b_m \) is a compression function if \( m > 1 \).

Hash functions and compression functions are used in many contexts (e.g., for making dictionaries). In cryptography, they also play an important role. Cryptographic hash and compression functions must have properties that guarantee their security. We now describe these properties informally. Let \( h : \Sigma^* \rightarrow \Sigma^n \) be a hash function or \( h : \Sigma^m \rightarrow \Sigma^n \) a compression function. We denote the set \( \Sigma^* \) or \( \Sigma^m \) of arguments of \( h \) by \( D \). If \( h \) is a hash function, then \( D = \Sigma^* \). If \( h \) is a compression function, then \( D = \Sigma^m \).

If \( h \) is used in cryptography, then \( h(x) \) must be easy to compute for all \( x \in D \). We will assume that this is the case.

The function \( h \) is called a one-way function if it is infeasible to invert \( h \); that is, to compute an inverse image \( x \) such that \( h(x) = s \) for a given image \( s \). What does "infeasible" mean? It is complicated to describe this in a precise mathematical way. To do so, we would need the language of complexity theory, which is beyond the scope of this book. Therefore, we only give an intuitive description. Any algorithm that on input of \( s \in \Sigma^n \) tries to compute \( x \) with \( h(x) = s \) almost always fails because it uses too much space or time. It is not known whether one-way functions exist. There are functions, however, that are easy to evaluate but for which no efficient inversion algorithms are known and that therefore can be used as one-way functions.

**Example 11.1.3**
If \( p \) is a randomly chosen 1024-bit prime and \( g \) a primitive root mod \( p \), then the function \( f : \{0, 2, \ldots, p-1\} \rightarrow \{1, 2, \ldots, p-1\}, x \mapsto g^x \mod p \) is easy to compute by fast exponentiation, but an efficient inversion function is not known because it is difficult to compute discrete