The combination of nonlinear functions with LFSRs provides a good means for generating running keys (see Chapter 5). The major problem is the difficulty of implementing complex and truly random nonlinear functions. What we would like to have is a simple and fast implementation of a large set of highly nonlinear functions which is readily indexed by the key. Moreover, any function randomly chosen from the large available set should, with virtual certainty, produce a sequence with good distribution properties. As we have seen in the preceding chapter, the knapsack provides an ideal instrument to meet these requirements, except perhaps for a slightly reduced operational speed when compared to direct or table implementations. The maximum order of most sum-bit functions \( f_{j,K}(x) \) assures that large linear complexity may be attained. The knapsack also distributes good statistics in the sense that equally distributed selector variables \( x_1, \ldots, x_N \) will, with virtual certainty, result in equally distributed bits \( s_j \) of the binary representation of the partial sum \( S \) when the modulus is a power of 2 (or close to a power of 2). In the original knapsack problem the weights are public. When the knapsack is used to realize a nonlinear mapping, there is no reason to give this information away; giving it away can only make the attack easier. From the theoretical viewpoint, it is also appealing that the amount of key introduced by keeping the weights secret greatly increases the unicity distance. The weights directly specify the coefficients of the realized nonlinear functions. Thus to allow a truly random key choice, the knapsack should have sufficient size. This suggests an application of the knapsack as a nonlinear state filter (Ruep 85a).
8.1 System Description

The key idea is to replace the nonlinear feedforward function which is applied to the individual stages of the LFSR and which we do not know how to realize efficiently) by a knapsack (which we know how to realize efficiently) whose input vector is taken to be the state of the LFSR and whose output, the integer partial sum, is converted to its binary representation to form the key stream. Fig. 8.1 illustrates the principle.

---

**Fig. 8.1. Conceptual Knapsack Stream Cipher**