Chapter 15
Knapsack Encryption

As a diversion we return in this chapter to another (once) promising public-key encryption scheme using a trap-door function: *Knapsack encryption*. It, too, is based on residue arithmetic, but uses multiplication rather than exponentiation, making it easier to instrument and theoretically more transparent.

The required trap door is obtained from the ancient knapsack puzzle: given the total weight of a knapsack and the weight of individual objects, which objects are in the bag? As it turns out, this problem can be made quite difficult to solve for someone who doesn't have the proper "key".

15.1 An Easy Knapsack

Number-theoretic exponentiation is not the only trap-door function potentially suitable for public-key encryption. Another trap door is opened (and kept almost closed) by *knapsack problems*.

Suppose we have a set of six stones, weighing 1, 2, 4, 8, 16 and 32 kilograms, respectively. Now, if a knapsack containing some of these stones weighs 23 kilograms more than its empty weight, then which stones are in the knapsack? The answer is given by the binary decomposition of 23:

$$23 = 16+4+2+1,$$

i. e., the stones weighing 1, 2, 4 and 16 kilograms are in the knapsack and no others.

This is an example of an *easy* knapsack problem. The problem remains easy if each weight exceeds the sum of the lower weights by at least one measureable unit. The binary sequence of weights 1, 2, 4, 8, etc., fulfills this condition, because $2 > 1$, $4 > 2+1$, $8 > 4+2+1$, etc. Such sequences are called "superincreasing". In fact, the binary sequence is the smallest positive superincreasing sequence if 1 is the just discriminatable weight difference [15.1].

Table 15.1 shows such an easy knapsack embedded in leading and trailing digits. If the clear message in binary form is 110101, then the encrypted message is 21,853,232. It is obtained by multiplying each knapsack row by 0 or 1 in accordance

Table 15.1 Encryption with an easy knapsack using binary "weights" 1, 2, 4, 8, 16, and 32 (see fourth and third columns in centre)

MESSAGE	EASY KNAPSACK	ENCRYPT
1	8301051	8301051
0	202061	
1	7804090	7804090
0	3508049	
1	2416013	2416013
1	3332078	3332078
		$218\underbrace{53}232$

53 = 110101 IN BINARY
WHICH IS THE MESSAGE

with the binary message and adding as shown in Table 15.1. To decrypt this number we only need the two digits 53 and its binary expansion

$$53 = 32+16+4+1,$$

corresponding to 110101 in binary notation, the original clear message.

15.2 A Hard Knapsack

Adding extra digits has not made the knapsack one bit harder to solve – so far. But now look at Table 15.2, in which each row of the simple knapsack is multiplied by $s = 324,358,647$ and reduced modulo $r = 786,053,315$ to form a *hard* knapsack.

Table 15.2 Encryption with a hard knapsack, obtained from the easy knapsack at left by multiplication and residue reduction (After *N. J. A. Sloane* [15.1])

MESSAGE	EASY KNAPSACK	HARD KNAPSACK	ENCRYPT
1	8301051	515862967	515862967
0	202061	679273397	
1	7804090	513438305	513438305
0	3508049	402161783	
1	2416013	427531791	427531791
1	3332078	376052641	376052641
			1832885704
			= CIPHER

MULTIPLY BY s = 324358647
AND REDUCE MOD r = 786053315

Now, if encryption proceeds as before by multiplying the binary message with the rows of the knapsack (the hard knapsack!), the result, after the multiplied rows have been summed, is 1,832,885,704, which is “a little” harder to decrypt if only the public encryption key is available.

To design a trap-door function based on a hard knapsack, proceed as follows [15.2]:

1) Pick a set of easy knapsack weights a_i, i. e., one that forms a superincreasing sequence:

$$a_{i+1} > \sum_{k=1}^{i} a_k. \tag{15.1}$$

2) Pick a modulus r and a coprime multiplier s, i. e., $(s,r) = 1$.
3) Calculate the hard knapsack

$$b_k \equiv sa_k \pmod r \tag{15.2}$$

and publish *only* the b_k.
4) Calculate a decrypting multiplier t such that

$$st \equiv 1 \pmod r \tag{15.3}$$

and *do not* publish it.

Suppose M is a message and its binary digits are m_k:

$$M = \sum_{k=0}^{K} m_k 2^k. \tag{15.4}$$

The encrypted message E is now calculated as follows:

$$E = \sum_{k=0}^{K} m_k b_k. \tag{15.5}$$

To decrypt, the legitimate receiver, knowing t, now forms

$$tE = \sum_k m_k tbk_k. \tag{15.6}$$

But

$$tb_k \equiv tsa_k \equiv a_k \pmod r. \tag{15.7}$$

Thus,

$$tE \equiv \sum_k m_k a_k \pmod r, \tag{15.8}$$

and to recover the message bits m_k is an easy knapsack problem, because the a_k form a superincreasing sequence. Specifically, for $a_k = 2^k$, tE *is* the original message:

$$tE \equiv M \pmod{r}. \tag{15.9}$$

One of the advantages of knapsack encryption is that it does not rely on the supposed difficulty of factoring specially constructed very large numbers. On the other hand, knapsacks are presently under attack because *Shamir* [15.3] and others [15.4] have shown the equivalence of the knapsack problem to a problem in integer programming for which a "fast" algorithm was recently invented by H. W. Lenstra of the University of Amsterdam. Further progress in knapsack ripping has been made by L. Adleman, and by J. C. Lagarias and A. M. Odlyzko [15.5]. Thus knapsacks, as described here, have developed holes through which our "secret" weights can fall for everyone to see. Who will darn the ripped knapsack?