The Rise and Fall of Knapsack Cryptosystems

A. M. Odlyzko

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

1. Introduction

The *knapsack* or *subset-sum problem* is to determine, given positive integers (or weights) $a_1, \ldots, a_n$, and s, whether there is a subset of the a_j that sums to s. This is equivalent to determining whether there are variables $x_1, \ldots, x_n$ such that

$$\sum_{j=1}^{n} x_j a_j = s , \qquad x_j \in \{0, 1\} \text{ for all } j . \tag{1.1}$$

If one thinks of s as the capacity of a knapsack and the a_j as the sizes of various items, then the question is whether the knapsack can be filled exactly by some collection of those items.

The knapsack problem is stated above in its *feasibility recognition form*, namely we ask only whether (1.1) is solvable. However, if this problem can be solved efficiently in general, then actual solutions can also be obtained fast. For example, if we know there is a solution, we can find out whether there is a solution with $x_1 = 1$ by testing whether there is a solution to

$$\sum_{j=2}^{n} x_j a_j = s - a_1 , \qquad x_j \in \{0, 1\} , \quad 2 \le j \le n .$$

If there is no such solution, we know that $x_1 = 0$ for all solutions to the original problem. Once x_1 is determined, we can go on and determine $x_2, x_3, \ldots$, one by one, and thus find at least one solution to (1.1).

The general knapsack problem is known to be *NP*-complete [13], and so it is thought to be quite hard. Being *NP*-complete means that if a polynomial time algorithm existed, there would also be polynomial time algorithms for all problems in the computational complexity class *NP*. This is thought to be unlikely, since this category includes many combinatorial optimization problems that have been investigated intensively over several decades, and for which no fast algorithms are known. This is of course a very subjective evaluation, since no good lower bound proofs are known for any of the problems that are *NP*-complete.

To determine whether (1.1) has a solution, and if so, to find it, one can compute all the $\Sigma\, x_j\, a_j$ with $x_j \in \{0, 1\}$, but that takes on the order of 2^n steps. (We are assuming here that the a_j are not too large, and will not be too precise about counting operations.) A better method is to compute

$$S_1 = \left\{ \sum_{j=1}^{\lfloor n/2 \rfloor} x_j\, a_j : \ x_j = 0 \ \text{ or } \ 1 \ \text{ for all } \ j \right\} ,$$

$$S_2 = \left\{ s - \sum_{j > \lfloor n/2 \rfloor} x_j\, a_j : \ x_j = 0 \ \text{ or } \ 1 \ \text{ for all } \ j \right\} ,$$

(this takes on the order of $2^{n/2}$ operations), sort each of the sets S_1 and S_2 (in about $n 2^{n/2}$ operations), and then scan S_1 and S_2, looking for a common element (about $2^{n/2}$ operations again). An element common to S_1 and S_2 arises precisely when there is a solution to (1.1); if

$$y = \sum_{j=1}^{\lfloor n/2 \rfloor} x_j\, a_j$$

$$= s - \sum_{j > \lfloor n/2 \rfloor} x_j\, a_j \ ,$$

then

$$s = \sum_{j=1}^{n} x_j\, a_j \ .$$

The entire procedure takes on the order of $n 2^{n/2}$ operations (but also about $2^{n/2}$ storage space, which may sometimes be prohibitive!). Surprisingly enough, this is still the fastest algorithm known for the general knapsack problem.

The basic idea of knapsack cryptosystems is to use a public set of weights $a_1, \ldots, a_n$ generated by *A*, say, to encode a message $(x_1, \ldots, x_n)$, $x_j \in \{0, 1\}$ for $1 \leq j \leq n$, as

$$s = \sum_{j=1}^{n} x_j \, a_j \ . \tag{1.2}$$

The message that would be transmitted to A would then be s. An eavesdropper would see s, and would know $a_1, \ldots, a_n$, since those are public, but in order to recover the message $(x_1, \ldots, x_n)$, would have to solve the (apparently intractable) knapsack problem.

The difficulty with the basic scheme outlined above is that while the eavesdropper is faced with having to solve a hard knapsack problem, so is the intended receiver A. That makes the scheme impractical, unless A possesses tremendously more computing power than any possible eavesdropper. (An assumption like that is completely unacceptable, since a basic criterion is that the cryptosystem should be very easy to use, and the cryptanalysis of it ought to be very hard.) Thus to use the knapsack problem for information transmission, it is necessary to make it possible for the intended receiver A to decode messages efficiently. There are some kinds of knapsack problems that are easy to solve. For example, if $a_j = 2^{j-1}$, $1 \le j \le n$, then

$$s = \sum_{j=1}^{n} x_j \, 2^{j-1} \ ,$$

and the x_j are just the digits in the binary representation of s. More generally, if the a_j form a *superincreasing sequence*, so that

$$a_j > \sum_{i=1}^{j-1} a_i \ , \qquad 2 \le j \le n \ , \tag{1.3}$$

then the knapsack problem is easy to solve; $x_n = 1$ if and only if

$$s > \sum_{j=1}^{n-1} a_j \ .$$

Once we find that $x_n = y$, say, we are faced with the smaller knapsack problem of determining $x_1, \ldots, x_{n-1}$ such that

$$s - y a_n = \sum_{j=1}^{n-1} x_j a_j , \qquad x_i \in \{0, 1\} , \qquad 1 \le j \le n-1 ,$$

and thus we can retrieve the entire message $(x_1 , \ldots , x_n)$ recursively.

The use of a superincreasing sequence $a_1 , \ldots , a_n$ of public weights would make it easy for the receiver *A* to read messages, but it would not prevent the eavesdropper from doing so as well. The basic idea of all knapsack public key cryptosystems is to start with a knapsack $b_1 , \ldots , b_n$ that is easy to solve and then transform it into the public knapsack $a_1 , \ldots , a_n$ by a process that conceals the structure of the knapsack, so that the public weights $a_1 , \ldots , a_n$ will appear to have no special structure and will hopefully leave the cryptanalyst baffled. At the same time, the designer of the system will be in a position to reverse the concealing transformation and will only have to solve the easy knapsack.

The most famous transformation of an easy secret knapsack into a seemingly more complicated public one is the modular multiplication used by Merkle and Hellman [21] in their basic knapsack cryptosystem. The receiver, who constructs the system to allow others to send information to her, starts with a superincreasing knapsack $b_1 , \ldots , b_n$ with

$$b_1 \approx 2^n , \quad b_j > \sum_{i=1}^{j-1} b_i , \quad 2 \le j \le n , \quad b_n \approx 2^{2n} . \tag{1.4}$$

(Some of the reasons for this choice of parameters will be presented in Section 2.) She then chooses positive integers *M* and *W* with

$$M > \sum_{j=1}^{n} b_j , \qquad (M, W) = 1 , \tag{1.5}$$

and computes

$$a'_j \equiv b_j W \ (\mathrm{mod}\ M) , \qquad 0 < a'_j < M . \tag{1.6}$$

We cannot have $a'_j = 0$, since $(M, W) = 1$ and $M > b_j$. Then she selects a permutation π of

$\{1, \ldots, n\}$ and defines

$$a_j = a'_{\pi(j)} , \qquad 1 \le j \le n . \tag{1.7}$$

The a_j form the public weights, while the b_j, M, W, and the permutation π are kept secret.

A message $(x_1, \ldots, x_n)$ is encoded as

$$s = \sum_{j=1}^{n} x_j \, a_j . \tag{1.8}$$

The receiver, knowing M and W, computes

$$c \equiv s \, W^{-1} \pmod{M} , \qquad 0 \le c < M , \tag{1.9}$$

where W^{-1} denotes the multiplicative inverse of W modulo M. By (1.6)-(1.8),

$$\begin{aligned} c &\equiv \sum_{j=1}^{n} x_j \, a_j \, W^{-1} \pmod{M} \\ &\equiv \sum_{j=1}^{n} x_j \, a'_{\pi(j)} \, W^{-1} \pmod{M} \\ &\equiv \sum_{j=1}^{n} x_j \, b_{\pi(j)} \pmod{M} . \end{aligned} \tag{1.10}$$

Since $M > \Sigma \, b_j$, the condition $0 \le c < M$ implies

$$c = \sum_{j=1}^{n} x_j \, b_{\pi(j)} . \tag{1.11}$$

After these operations the receiver is faced with the knapsack problem (1.11), which is easy to solve, since the b_j form an increasing sequence.

The paragraphs above describe the basic, or *singly-iterated Merkle-Hellman cryptosystem*. A variation on it is the *multiply-iterated Merkle-Hellman cryptosystem*, in which the easy (secret) knapsack is disguised through a series of modular multiplications; we let $M_1 = M$, $W_1 = W$, $a_j^{(0)} = b_j$, $a_j^{(1)} = a'_j$, and construct a sequence of modulus, multiplier, and weight combinations by choosing iteratively M_k, W_k, to be positive integers such that $(M_k, W_k) = 1$,

$$M_k > \sum_{j=1}^{n} a_j^{(k-1)} ,$$

and define

$$a_j^{(k)} \equiv a_j^{(k-1)} W_k \pmod{M_k} .$$

This kind of scrambling operation, when performed a few times, seems to conceal the original design even more effectively than the basic (singly-iterated) scheme does.

The Merkle-Hellman knapsack cryptosystems, as well as various other ones that were proposed, have the attractive feature that they can be run at the high speeds. Classical secret key cryptosystems, such as the Data Encryption Standard (DES), when implemented with special purpose chips, can be run at speeds of tens of millions of bits per second, and even in software on modest size machines can encrypt on the order of 10^5 bits per second. The RSA system is very slow by comparison. It was thought for a long time that a modulus of about 500 bits was quite secure. (The initial challenge cipher proposed by Rivest, Shamir, and Adleman involved a modulus of about 430 bits.) At that size, though, even the best existing special purpose chips can only encrypt at the rate of 10^4 or 2×10^4 bits per second, and software implementations are limited to something on the order of 10^2 bits per second. Thus the RSA system is about 100 to 1000 times slower than classical cryptosystems.

The Merkle-Hellman knapsack cryptosystems seemed to offer the possibility of much higher speed. For $n \approx 100$ (which seemed a reasonable parameter, since the best algorithms known for solving the knapsack problem require on the order of $2^{n/2}$ operations), the singly-iterated Merkle-Hellman system can be more than 100 times faster than RSA (with modulus of about 500 bits), whether hardware or software implementations were used, and thus can rival classical secret key systems in speed. This speed advantage is due to a small extent to dealing with smaller numbers (200 vs. 500 bits) but mostly to having to do only one modular multiplication, as opposed to over 500 for RSA. There is a slight disadvantage in that twice the communication capacity is needed (m bits is encoded into $\approx 2m$ bits, as against about m bits for RSA), and the size of the public key is larger ($2n^2$ bits, which is about 20,000 for $n \approx 100$, as against 1000 bits for 500-bit RSA).

The major question about knapsack public key systems has always concerned their security. Some of the doubts have been very general, and apply to the RSA cryptosystem as well. What if $P = NP$, and somebody discusses a wonderful new approach that solves all problems in NP efficiently? Even if $P \neq NP$, what if most instances of the knapsack problem are easy to solve? Since the theory of NP-completeness deals with the worst-case situation, there is nothing to forbid this from happening, and many NP-complete problems are easy to solve on average, cf. [27]. Even if most instances of the general knapsack problem are hard, how can one be certain that the cryptanalyst will not be able to deduce from the public knapsack what the construction method was?

In addition to the general doubt about security of all public key systems mentioned above, several other doubts were raised that applied specifically to knapsacks and other systems based on NP-complete problems. On the very abstract level, there was an interesting result of Brassard [2] which says essentially that if breaking a cryptosystem is NP-hard, then $NP = Co\text{-}NP$, which would be a very surprising complexity theory result. Thus if $NP \neq Co\text{-}NP$, then breaking the Merkle-Hellman cryptosystem cannot be NP-hard, and so is likely to be easier than solving the general knapsack problem.

More specific concerns about the security of knapsack cryptosystems were based to the linearity of such schemes. Reading Eq. (1.1) modulo 2, one obtains an equation for the x_j modulo 2, which provides a single bit of information about them. (We can obviously assume that not all of the a_j are even, as in that case, which would correspond to a trivial equation, we could look at the knapsack with weights $a_1/2, \ldots, a_n/2$, etc.) No one ever found a way to take advantage of this bit, but the fact that it could be determined was felt to be suspicious. (Ideal cryptographic systems should reveal no information at all about the plaintext message being transmitted.)

In addition to the general suspicions mentioned above, a number of knapsack cryptosystems were broken in the late 1970's. The final fall of knapsack cryptosystems can be dated to Shamir's announcement in the spring of 1982 of a polynomial time attack on the singly-iterated Merkle-Hellman cryptosystem [26]. This was quickly followed by a string of attacks on other knapsack cryptosystems, culminating in Brickell's attack on the multiply-iterated Merkle-Hellman system [4]. These attacks relied on the fact that the modular multiplication method does not disguise completely the easy knapsack that is the basis of the construction.

In addition to the attacks on specific knapsack systems that have been developed, there are two attacks on so-called *low-density knapsacks*, namely those in which the weights a_j are large. These attacks do not assume any particular structure in the knapsack. They are due to Brickell [3] and Lagarias and Odlyzko [18], respectively. As a result, both of the two basic fears about knapsack cryptosystems have been borne out; their constructions can often be unraveled, and in addition, many cases of the general knapsack problem can be solved efficiently.

A large variety of knapsack cryptosystems have been shown to be insecure, most with the use of tools from the area of diophantine approximation. The paper [6] contains a survey of many of the systems that have been broken as well as descriptions of some of the attacks. For full details, the reader is advised to consult [6] and many of the references contained there, such as [3,4,5,8,11,16,17,18,22,26]. The remainder of this paper is devoted to a description of one each of the two kinds of basic attacks that have been used. Section 2 describes the attack on the singly-iterated Merkle-Hellman cryptosystem. This attack allows the cryptanalyst to read encrypted messages just about as fast as the designer of the system can. Section 3 describes one of the low-density attacks. It is harder to carry out, but applies to a rich variety of knapsacks, not just the cryptographic ones.

While most knapsack cryptosystems have been broken, there are a few that so far have resisted all attacks. One of the most attractive of these is the Chor-Rivest system [7], which involves a combination of number theory ideas and knapsacks. It is described briefly in Section 4. Other recent applications of knapsacks include [15]. The search for secure knapsacks continues both because of the attractively high speed that knapsack systems offer, and because of the desire to have a wide variety of cryptosystems available. After all, factorization and discrete logarithm problems could turn out to be efficiently solvable, and if that happened, it would be nice to have substitute systems available.

2. Basic Merkle-Hellman system

This section shows how to break the basic Merkle-Hellman knapsack cryptosystem. This attack was the first serious cryptanalytic assault on knapsacks, and it led to the breaking of numerous other knapsack systems. It is due to A. Shamir [26]. Many of the crucial observations about the weaknesses of the basic Merkle-Hellman system had been made earlier by Eier and Lagger [10] and by Desmedt, Vandewalle, and Govaerts [9].

Before describing the Shamir attack, we will say a few words about the choice of parameters (1.4). If some of the b_i, and therefore also M, are large, then the knapsack will be inefficient, since n bits of information will be encoded into roughly $\log_2 M$ bits. On the other hand, it is dangerous to let any of the b_j be too small. If we had $b_1 = 1$, say, then $a_j = W$ for some j, and since the knapsack can be shown to be breakable if either W or M is revealed, one could try each of the a_j as a possible W and thus compromise the system. Therefore a reasonable compromise between efficiency and security seemed to be to select the parameters in (1.4).

One could select $b_i = c2^{i-1}$, $1 \leq i \leq n$, for some $c \approx 2^n$. This choice would satisfy (1.4), but it would yield a very insecure system. The reason is that for $1 \leq j \leq n-1$, we would have

$$a'_{j+1} = 2a'_j \quad \text{or} \quad 2a'_j - M ,$$

and each case would usually occur about $n/2$ times. Therefore, by computing

$$a_i - 2a_j , \quad 1 \leq i,\, j \leq n ,$$

we would obtain about $n/2$ occurrences of $-M$ among the n^2 differences, so we could deduce what M is, and this would break the system. Other approaches have also been preposed. For example, one could take $b_i = 2^{i-1}$, but hide this structure by a double iteration of the modular multiplication method [14]. This construction, however, which is more secure than what was presented above, was shown to be insecure by the author (unpublished) by the use of continued fractions. The moral to be drawn from the above examples is that it is easy to construct knapsack cryptosystems that seem attractive, yet are insecure. This explains the generally high level of suspicion that experts have had about knapsacks from the moment they were proposed.

We now proceed to the description of the Shamir attack on the basic Merkle-Hellman system. We assume that the secret knapsack weights $b_1, \ldots, b_n$ satisfy (1.4), and that they are transformed into the public weights $a_1, \ldots, a_n$ via (1.6) and (1.7). We let $U \equiv W^{-1} \pmod M$, $0 < U < M$. By (1.6), we have

$$a_j \equiv b_{\pi(j)}\, W \pmod M , \tag{2.1}$$

or

$$b_{\pi(j)} \equiv a_j\, U \pmod M , \tag{2.2}$$

and this means that for some integer k_j,

$$a_j\, U - k_j\, M = b_{\pi(j)} . \tag{2.3}$$

Hence

$$\frac{U}{M} - \frac{k_j}{a_j} = \frac{b_{\pi(j)}}{a_j\, M} \,. \tag{2.4}$$

What this means is that all of the k_j / a_j are close to U/M. The cryptanalyst does not know U, M, π, the k_j, or the b_j, but does know the a_j. The problem now is to extract some information about all these unknowns from the above remark about (2.4).

From (1.4) we see, for example, that

$$b_1 ,\ b_2 ,\ \ldots ,\ b_5 \lesssim 2^n \,. \tag{2.5}$$

Therefore, if we let $j_i = \pi^{-1}(i)$, then we obtain

$$\left| \frac{U}{M} - \frac{k_{j_i}}{a_{j_i}} \right| \lesssim \frac{2^n}{2^{4n}} = 2^{-3n} \,, \quad 1 \le i \le 5 \,, \tag{2.6}$$

and hence, subtracting the $i = 1$ term from the others,

$$\left| \frac{k_{j_i}}{a_{j_i}} - \frac{k_{j_1}}{a_{j_1}} \right| \lesssim 2^{-3n} \,, \quad 2 \le i \le 5 \,. \tag{2.7}$$

This, in turn, implies that

$$| k_{j_i}\, a_{j_1} - k_{j_1}\, a_{j_i} | \lesssim 2^n \,, \quad 2 \le i \le 5 \,. \tag{2.8}$$

The inequalities (2.8) show just how unusual the a_{j_i} and k_{j_i} are. After all, each of them is on the order of 2^{2n}, so $k_{j_i}\, a_{j_1}$ is on the order of 2^{4n}. For the difference of two such terms to be on the order of 2^n requires some very special structure. It cannot happen often, and (in most cases) the k_{j_i}, $1 \le i \le 5$, are determined uniquely by (2.8). (Remember that the a_{j_i} are public.) The question is: How do we determine the k_{j_i} in practice? Shamir's main contribution was to notice that this could be done in polynomial time by invoking H. W. Lenstra's theorem that the integer programming problem in a fixed number of variables can be solved in polynomial time [20]. This yields the k_{j_i}, $1 \le i \le 5$. Once the k_{j_i} are found, one obtains an approximation to U/M from

(2.6), and that lets one construct a pair (U', M') with U'/M' close to U/M such that the weights c_j obtained by

$$c_j \equiv a_j U' \pmod{M'}, \quad 0 < c_j < M', \quad 1 \le j \le n, \tag{2.9}$$

form a superincreasing sequence (after they are arranged in increasing order). Note that this attack does not recover the U and M that were constructed by the original designer of the system. There are infinitely many pairs U', M' that yield a superincreasing sequence c_j, and any of them can be used to decrypt messages.

One point that was glossed over in the above discussion is that of finding $j_1, \ldots, j_5$. After all, the permutation π is secret, so how can the cryptanalyst be sure she has picked out the right indices? The answer to that is simple; the cryptanalyst considers all possible choices of $j_1, \ldots, j_5$, and since there are only on the order of n^5 of them, this keeps the running time polynomial. (In practice, one would use a different approach, choosing more than 5 weights at a time, but not requiring that they come from the 5 smallest secret weights, but, say, from the half of the secret weights that are smallest.)

The crucial tool that made the above attack succeed was Lenstra's result on integer programming in a fixed number of variables. It was applied to solve (2.6), which is a typical problem in inhomogeneous diophantine approximation. In the case of the basic Merkle-Hellman knapsack with parameters given by (1.4), it turns out that the attack can also be carried out using continued fractions, a much simpler (and more effective) tool than Lenstra's algorithm [6]. Continued fractions, while very useful, have unfortunately only limited applicability to knapsacks. Even for the basic Merkle-Hellman knapsack with the b_i larger than specified in (1.4) (say $b_1 \approx 2^{2n}$, $b_n \approx 2^{3n}$) the continued fraction method fails. The Lenstra result is powerful, but is of mostly theoretical interest, since its running time is given by a high degree polynomial, and so it has never been implemented. However, another tool, the Lovász lattice basis reduction algorithm, which was motivated by Lenstra’s method, was soon applied to knapsacks, and was crucial for the development of the low-density attacks as well as for the breaking of other systems.

3. A low-density attack

This section presents one of the two known approaches to solving general low-density knapsacks, the one due to Lagarias and Odlyzko [18]. Compared to the Brickell low-density attack [3], it appears to break more knapsacks, and it can be rigorously proved to work, at least for extremely low-density knapsacks. Most important for this paper, though, is the fact that this attack is conceptually much simpler, and so is very easy to present.

Before discussing the low-density attack of [18], we introduce lattices and lattice basis reduction algorithms. An *integer lattice* L is an additive subgroup of Z^n that contains n linearly independent vectors over R^n. A basis $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ of L is a set of elements of L such that $L = Z\mathbf{v}_1 \oplus \cdots \oplus Z\mathbf{v}_n$. A basis will be represented by the $n \times n$ basis matrix

$$V = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{bmatrix} .$$

Bases are not unique; if U is a unimodular $n \times n$ matrix (matrix with integer entries and determinant ± 1) then UV is another basis. A very important problem is that of finding the shortest non-zero vector of a lattice, given its basis. This problem appears to be quite hard, although there is no proof that it is.

An important concept related to that of the shortest non-zero vector is that of a *reduced basis*. There are several definitions of reduced bases, but the one we will deal with (but will not define precisely) is that due to Lovász [19]. What we will use is the fact that the first vector in a

Lovász-reduced basis is not too long. The precise result is that if $\mathbf{v}_1, \ldots, \mathbf{v}_n$ is a Lovász-reduced basis of a lattice, then [19; Prop. 1.11]

$$|\mathbf{v}_1|^2 \le 2^{n-1} \min_{\substack{\mathbf{x} \in L \\ \mathbf{x} \neq \mathbf{0}}} |\mathbf{x}|^2 . \tag{3.1}$$

(Actually, one can get a slightly better bound by modifying the definition of Lovász reduction, see [18, 19].) What's most important is that Lovász found a polynomial time algorithm that, given a basis for a lattice, produces a reduced basis. This algorithm thus has a good theoretical running time bound, but in addition turns out to be very fast in practice [18], and usually finds a reduced basis in which the first vector is much shorter than is guaranteed by (3.1). (In low dimensions, it has been observed empirically that it usually finds the shortest non-zero vector in a lattice.)

The low-density attack of [18] is very easy to describe. Suppose that we wish to find a solution to (1.1), where the a_j and s are given. We form the $(n+1)$-dimensional lattice with basis

$$V = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & 0 & \cdots & 0 & -a_2 \\ & & & \vdots & & \\ 0 & 0 & 0 & \cdots & 1 & -a_n \\ 0 & 0 & 0 & \cdots & 0 & s \end{bmatrix} . \tag{3.2}$$

If $\mathbf{v}_1, \ldots, \mathbf{v}_{n+1}$ are the rows of V, and the x_j solve (1.1), then

$$\sum_{j=1}^{n} x_j \mathbf{v}_j + \mathbf{v}_{n+1} = (x_1, x_2, \ldots, x_n, 0) , \tag{3.3}$$

and since the x_j are 0 or 1, this vector is very short. The basic attack of [18] consists of running the Lovász lattice basis reduction algorithm on the basis (3.2) and checking whether the resulting reduced basis contains a vector that is a solution to (1.1).

If the a_j are large, then one could expect that most vectors in the lattice generated by (3.2) would be large, and so the vector (3.3) corresponding to a solution of (1.1) might be the shortest one. It is possible to prove a rigorous result in this direction. If the a_j are chosen at random with

$$a_j \approx 2^{\beta n} , \qquad 1 \leq j \leq n ,$$

where β is any constant > 1.54725 (the precise definition of the critical constant is complicated and is given in [18]), then the vector (3.3) is the shortest non-zero vector in most of these lattices, as is shown in [18]. (Frieze [11] has obtained a simplified proof of this result. Also, the claim above is valid only for $\Sigma\, x_j \leq n/2$, but it is easy to reduce the general problem to this case.) Thus if we could efficiently find shortest non-zero vectors in lattices, we could solve most low-density knapsacks.

The rigorous analysis of [11, 18] applies only to random knapsacks. However, it appears that this same result also applies to knapsacks that arise in cryptographic constructions. Heuristically, there are even technical reasons to think that the above analysis would apply even more strongly to knapsack cryptosystems, since these are less likely to have what are called small linear relations. See [18] for a detailed discussion.

The major deficiency of the rigorous analysis mentioned above is that it applies directly only when we have a way to compute the shortest non-zero vector in a lattice. So far no efficient way to do this has been found. If we have to rely on the rigorous bounds that have been obtained on the performance of the Lovász algorithm, then we can only assert that most knapsacks are solvable whose weights are extremely large, on the order of 2^{n^2}. (See [18] for the precise statement.) In practice, however, the Lovász algorithm performs much better than the worst-case bounds that have been proved for it. Furthermore, some improvements have been suggested, both in the way it is implemented [18, 23] and in the basic construction of the algorithm [24, 25], which yield significant advantages in either speed or success in finding short vectors.

Furthermore, these improvements suggest very strongly that one can obtain even better algorithms. Thus in judging the security of knapsack cryptosystems it is probably prudent to assume that shortest non-zero vectors in lattices can be found efficiently.

4. The Chor-Rivest knapsack

The Chor-Rivest cryptosystem [7] is one of the few knapsack systems that have not been broken, and is among the most attractive ones. Let $GF(p^h)$ be a finite field such that $p^h - 1$ has only moderate prime factors, so that it is fairly easy to compute discrete logarithms in $GF(p^h)$. One possible choice, suggested in [7], is $p = 197$, $h = 24$. Let $f(x)$ be a monic irreducible polynomial of degree h over $GF(p)$, so that $GF(p^h)$ can be represented as $GF(p)[x]/f(x)$. Let t be the residue class of x modulo $f(x)$, so that t is an element of $GF(p^h)$ and $f(t) = 0$. Further, let g be a generator of the multiplicative group of $GF(p^h)$. For $\alpha \in GF(p)$, let a_α be an integer such that

$$g^{a_\alpha} = t + \alpha , \tag{4.1}$$

and let π be a one-to-one map from $\{0, 1, \ldots, p-1\}$ to $GF(p)$. We choose an integer d, and define

$$c_i \equiv a_{\pi(i)} + d \mod p^h - 1 , \quad 0 \le c_i \le p^h - 2 . \tag{4.2}$$

Then $c_0, c_1, \ldots, c_{p-1}$ are the public key. Messages to be encoded are first transformed into p-vectors $(m_0, \ldots, m_{p-1})$ of non-negative integers such that

$$\sum_{i=0}^{p-1} m_i = h . \tag{4.3}$$

(This transformation is easy to perform.) The ciphertext that is transmitted is then

$$s = \sum_{i=0}^{p-1} m_i c_i . \tag{4.4}$$

The decryption is accomplished as follows. First compute

$$r \equiv s - hd \mod p^h - 1 , \quad 0 \le r \le p^h - 2 ,$$

and then we have

$$g^r = \prod_{i=0}^{p-1} g^{m_i a_{\pi(i)}} .$$

Now g^r is represented as a polynomial G in x of degree $< h$, and $g^{a_{\pi(i)}}$ is represented as $x + \pi(i)$. The product

$$\prod_{i=0}^{p-1} g^{m_i a_{\pi(i)}}$$

is then representable as

$$\prod_{i=0}^{p-1} (x + \pi(i))^{m_i} ,$$

and so we must have

$$G + f(x) = \prod_{i=0}^{p-1} (x + \pi(i))^{m_i} ,$$

since the polynomial on the right side above is monic and of degree exactly h. Now one can recover the m_i by factoring $G + f(x)$, and this can be done efficiently.

Chor and Rivest discuss several attacks on this system in this paper. So far, though, none of them have been modified so as to break the full system when its parameters are chosen carefully. This represents a nice challenge for future work.

Acknowledgement. The author thanks T. H. Foregger for his helpful comments on an earlier version of this manuscript.

References

[1] L. M. Adleman, "On Breaking Generalized Knapsack Public Key Cryptosystems," *Proc. of the Fifteenth ACM Symposium on Theory of Computing, 1983*, pp. 402-412.

[2] G. Brassard, "A Note on the Complexity of Cryptography", *IEEE Trans. on Inform. Theory*, vol. IT-25, 1979, pp. 232-233.

[3] E. F. Brickell, "Solving low density knapsacks," *Advances in Cryptology-Proc. Crypto 83*, Plenum Press, New York, 1984, pp. 25-37.

[4] E. F. Brickell, "Breaking Iterated Knapsacks," *Advances in Cryptology-Proc. Crypto 84*, Springer-Verlag, Berlin, 1985, pp. 342-358.

[5] E. F. Brickell, "The Cryptanalysis of Knapsack Cryptosystems," *Applications of Discrete Mathematics*, R. D. Ringeisen and F. S. Roberts, eds., SIAM, 1988, pp. 3-23.

[6] E. F. Brickell and A. M. Odlyzko, "Cryptanalysis: A Survey of Recent Results," *Proc. IEEE*, vol. 76, 1988, pp. 578-593.

[7] B. Chor and R. Rivest, "A Knapsack-Type Public Key Cryptosystem Based on Arithmetic in Finite Fields," *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 54-65. Revised version in *IEEE Trans. Information Theory*, vol. IT-34, 1988, pp. 901-909.

[8] Y. Desmedt, "What Happened with Knapsack Cryptographic Schemes?," *Performance Limits in Communication, Theory and Practice*, J. K. Skwirzynski, ed., Kluwer, 1988, pp. 113-134.

[9] Y. G. Desmedt, J. P. Vandewalle and R. J. M. Govaerts," "A Critical Analysis of the Security of Knapsack Public Key Algorithms," *IEEE Trans. Inform. Theory*, vol. IT-30,

1984, pp. 601-611, also presented at *IEEE Intern. Symp. Inform. Theory*, Les Arcs, France, June 1982, Abstract of papers, pp. 115-116.

[10] R. Eier and H. Lagger, "Trapdoors in Knapsack Cryptosystems," *Cryptography*, Burg Feuerstein, Germany, March 29 - April 2, 1982, Lecture Notes in Computer Science, vol. 149, Springer-Verlag, New York, 1983, pp. 316-322.

[11] A. M. Frieze, "On the Lagarias-Odlyzko Algorithm for the Subset Sum Problem," *SIAM J. Comp.*, vol. 15, no. 2, May 1986, pp. 536-539.

[12] M. L. Furst and R. Kannan, "Succinct Certificates for Almost All Subset Sum Problems," *SIAM J. Comput.*, vol. 18, 1989, pp. 550-558.

[13] M. R. Garey and D. S. Johnson,"*Computers and Intractability: A Guide to the Theory of NP - Completeness*," W. H. Freeman and Company, San Francisco, 1979.

[14] P. S. Henry, "Fast Implementation of the Knapsack Cipher,'' *Bell System Tech. J.*, vol. 60, 1981, pp. 767-773.

[15] R. Impagliazzo and M. Naor, "Efficient Cryptographic Schemes Provably as Secure as Subset Sum," *Proc. 30th IEEE Symposium on Foundations of Computer Science*, IEEE, 1989, pp. 236-241.

[16] J. C. Lagarias, "Knapsack Public Key Cryptosystems and Diophantine Approximation," *Advances in Cryptology, Proc. Crypto 83,* Plenum Press, New York, 1984, pp. 3-23.

[17] J. C. Lagarias, "Performance Analysis of Shamir's Attack on the Basic Merkle-Hellman Knapsack Cryptosystem," *Proc. 11th Intern. Colloquium on Automata, Languages and Programming (ICALP),* Lecture Notes in Computer Science, vol. 172, Springer-Verlag, Berlin, 1984, pp. 312-323.

[18] J. C. Lagarias and A. M. Odlyzko, "Solving Low Density Subset Sum Problems," *J. Assoc. Comp. Mach.*, vol. 32, 1985, pp. 229-246. Preliminary version in *Proc. 24th IEEE Symposium on Foundations of Computer Science*, IEEE, 1983, pp. 1-10.

[19] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, "Factoring Polynomials with Rational Coefficients," *Mathematische Annalen 261*, 1982, pp. 515-534.

[20] H. W. Lenstra, Jr., "Integer Programming with a Fixed Number of Variables,"*Math. Operations Research*, vol. 8, 1983, pp. 538-548.

[21] R. C. Merkle and M. E. Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks," *IEEE Trans. Inform. Theory*, vol. 24, 1978, pp. 525-530.

[22] A. M. Odlyzko, "Cryptanalytic Attacks on the Multiplicative Knapsack Cryptosystem and on Shamir’s Fast Signature System," *IEEE Trans. Inform. Theory*, vol. IT-30, 1984, pp. 594-601.

[23] S. P. Radziszowski and D. L. Kreher, "Solving Subset Sum Problems with the L^3 Algorithm," *J. Combin. Math. Combin. Comput.*, vol. 3, 1988, pp. 49-63.

[24] C. P. Schnorr, "A More Efficient Algorithm for a Lattice Basis Reduction," *Journal of Algorithms*, vol. 9, 1988, pp. 47-62.

[25] C. P. Schnorr, "A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms", *Theoretical Computer Science*, vol. 53, 1987, pp. 201-224.

[26] A. Shamir, "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem," *IEEE Trans. Inform. Theory*, vol. IT-30, 1984, pp. 699-704.

[27] H. S. Wilf, "Backtrack: An $O(1)$ Expected Time Graph Coloring Algorithm," *Inform. Proc. Letters*, vol. 18, 1984, pp. 119-122.

The Rise and Fall of Knapsack Cryptosystems

A. M. Odlyzko

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

Cryptosystems based on the knapsack problem were among the first public key systems to be invented, and for a while were considered to be among the most promising. However, essentially all of the knapsack cryptosystems that have been proposed so far have been broken. These notes outline the basic constructions of these cryptosystems and attacks that have been developed on them.