COMPACT KNAPSACKS ARE POLYNOMIALLY SOLVABLE

Hamid R. Amirazizi, Ehud D. Karnin and Justin M. Reyneri

Information Systems Laboratory

Stanford University

Stanford, California 94305

Extended Abstract

Given a vector $\mathbf{a}=(a_1, a_2, \ldots, a_n)$ where each a_i is a positive integer, and an integer S, the knapsack problem is to find a binary vector $\mathbf{x}=(x_1, x_2, \ldots, x_n)$ such that

$$\sum_{i=1}^{n} a_i x_i = S , \qquad x_i \in \{0,1\} \tag{1}$$

The associated decision problem, i.e., determining whether (1) has a solution, is in the class of NP-Complete problems. Therefore it is believed that solving (1) is very hard in general.

Merkle and Hellman suggested a public key cryptographic system, which makes use of this difficulty. In their method $\mathbf{a}$ is the public key, $\mathbf{x}$ is the message and S is the cyphertext. The legitimate receiver knows "trapdoor" information embedded in $\mathbf{a}$ and hence can easily recover $\mathbf{x}$. On the other hand the cryptanalyst must try to solve a knapsack problem of the form (1).

The main disadvantage of the system is the size of the public key. Merkle and Hellman [2] recommended some tens of thousands of bits, which turns out to be a problem in terms of the memory required to store the public keys in a many user system.

This work was supported by the National Security Agency under contract no. MDA904-81-C-0414, and by the Joint Services Electronics Program under contract no. DAAG29-81-0057.

A possible way to reduce the size of the public key (while keeping the original message set) is to reduce n and allow non-binary solutions to equation (1).

Suppose the x_i are allowed to take values in $\{0, 1, \ldots, K-1\}$, and let $k=\log_2 K$. A kn bit message can now be encrypted using a key of only n elements. Hence the key size may be reduced by a factor of about k.

To summarize, the compact knapsack problem is: Given positive integers a_i, $i=1, 2, \ldots, n$, and S, find integers $(x_1, x_2, \ldots, x_n)$ such that

$$\sum_{i=1}^{n} a_i x_i = S \tag{2}$$

where

$$0 \le x_i \le K-1 \qquad i=1, 2, \ldots, n \tag{3}$$

We will present an algorithm which, for fixed n, solves the knapsack in time which is bounded by a polynomial in l, the length of the problem, where

$$l = \max_{1 \le i \le n} \{\log_2(a_i+1)\} + k.$$

The method relies heavily on the algorithm recently developed by Lenstra. His algorithm solves the integer programming problem in polynomial time when the number of variables is fixed.

The method can be divided into two parts. In the first, we apply Euclid's greatest common divisor algorithm $n-1$ times to the $\{a_i\}$ to obtain an $n-1$ dimensional, integer, parameterization of the familiy of solutions to (2). That is, we can write

$$x_i = c_{i0} + \sum_{j=1}^{n-1} c_{ij} t_j , \quad i=1,\ldots,n. \tag{4}$$

The size of the resulting c_{ij} is at most nl. In other words, transforming the knapsack to (4) only increases the size of the problem by a constant.

Applying the constraints (3) to equation (4) gives the $2n$ inequalities:

$$\sum_{j=1}^{n-1} c_{ij} t_j \le K-1-c_{i0}$$

and

$$\sum_{j=1}^{n-1} -c_{ij}t_j \le c_{i0}, \quad i=1,\ldots,n.$$

These may be written in matrix notation as

$$C\mathbf{t} \le \mathbf{b}, \tag{5}$$

where C is a $2n$ by $n-1$ matrix, $\mathbf{b}=(K-1-c_{10},\ldots,K-1-c_{1n},c_{10},\ldots,c_{1n})^T$, and $\mathbf{t}=(t_1,\ldots,t_{n-1})^T$.

Finding a solution $\mathbf{t}$ to (5) is exactly the problem treated by Lenstra. Thus, the second part of the solution calls upon his algorithm to find an integer inside the convex region described by (5). The reader is referred to Lenstra's paper for details.

As we mentioned above, the compact knapsack has been suggested as a practical way of reducing the size of the public key vector, while maintaining the same security as the corresponding binary knapsack. While it is known that the compact knapsack problem is NP-Complete, we have just seen that the n dimensional compact knapsack problem can be solved in polynomial time in l for a fixed value of n, and therefore belongs to the class P. However, at this point we do not know a tight upper-bound for the running time of Lenstra's algorithm. Although it is polynomial in l for a fixed n, the degree of the polynomial is exponential in n. Further work is needed to determine whether the "reduction process", which gives rise to this degree, can be simplified. Testing of the algorithm will also be useful to find its *typical* running time, which is important for assessing its ability to break cryptosystems