Knapsack Cryptosystems

Merkle-Hellman Knapsack

This well-known cryptosystem was first described by Merkle and Hellman in 1978. Although this system, and several variants of it, were broken in the early 1980's, it is still worth studying for several reasons, not the least of which is the elegance of its underlying mathematics.

Subset Sum Problem

The underlying mathematical problem is the ***subset sum problem*** closely related to the more famous *knapsack* problem of OR (thus, the "knapsack" in the name of this system is a misnomer). The problem can be described as follows. Suppose that to each element of a given set S there is assigned a (distinct) positive integer. Given a subset of S, the sum of the associated elements of the subset give an integer that corresponds to this subset. The subset sum problem is the inverse of this, that is, given an integer T, is there a subset of S whose sum is T? This decision problem (requiring only a yes-no response) is an NP-complete problem. So is the corresponding search problem, given a T for which the answer is yes, find a subset (or subsets, the answer may not be unique) which gives this sum. As with any NP problem, even though there is no known polynomial time algorithm for solving the general problem, some special cases may be solved easily. This is the case with the subset sum problem, and this permits the creation of a **trapdoor.**

Problem Formulation

We will formulate the problem as follows: Let $s_1, s_2, ..., s_n$ be an ordered set of positive integers (called ***sizes***) and T a positive integer. The search problem is to find a 0-1 vector $(x_1, x_2, ..., x_n)$ so that

$$x_1 s_1 + x_2 s_2 + ... + x_n s_n = T.$$

The simple-minded algorithm of just going through all possibilities is of course of exponential running time in terms of n. However, for certain sets of sizes, there are faster methods.

Super-Increasing Sets

A set of sizes is said to be ***super-increasing*** if each term is greater than the sum of all the previous terms. So, **3, 7, 12, 30, 60, 115** is a super-increasing set while **3, 5, 7, 9, 11** is not. Note that the consecutive powers of any fixed number will always form a super-increasing set of sizes (e.g., **1, 3, 9, 27, 81**, ...). To answer the subset sum problem for a super-increasing set of sizes, just one pass through the set is needed, so the running time is O(n). For example, to decide if T is a subset sum of the super-increasing set **3, 7, 12, 30, 60, 115**, we start by finding the largest value in the set that is less than or equal to T, subtract this value from T to form T' and then repeat the process for T'. Continue in this manner and if we end with 0 then T is a subset sum, and the numbers we subtracted form the subset, otherwise T is not a subset sum.

Examples

3, 7, 12, 30, 60, 115

T = **82**: 82 - 60 = 22;
22 - 12 = 10;
10 - 7 = 3;
3 - 3 = 0

82 is a subset sum and the subset is (1,1,1,0,1,0) corresponding to 3, 7, 12, 60.

T = **80**: 80 - 60 = 20;
20 - 12 = 8;
8 - 7 = 1 80 is not a subset sum.

T = **85**: 85 - 60 = 25;
25 - 12 = 13 and we can stop, this can not lead to a solution.

The Cryptosystem

To use the subset sum problem in a cryptosystem, Bob creates and makes public an ordered list of sizes $s_1, s_2, ..., s_n$. To communicate with Bob, Alice takes her message and writes it as a long binary string. She breaks up the string into blocks of size n (padding the last one with 0's if necessary). Each block is used as the characteristic vector of a subset, and she calculates the subset sum and sends that to Bob (that is, she calculates the sum of the s_i's which correspond to the 1's in the block). Oscar, on intercepting this message, must solve the subset sum problem for each block to recover the message and this is an intractible NP problem if the parameters of the system are large enough. On the other hand, this is exactly what Bob must do as well to decrypt the message. However, Bob knows a **secret !!**

The Cryptosystem

His public list of sizes is really a super-increasing set that has been cleverly disguised, and all he has to do is convert the list back to the original super-increasing set and solve the simple problem.

The Merkle-Hellman Cryptosystem

In the Merkle-Hellman cryptosystem, the "disguise" is done as follows: Let $t_1, t_2, ..., t_n$ be a super-increasing set of sizes. Chose a prime p which is greater than the sum of all the t_i, and a number b with $1 < b < p$. Now Bob calculates his public set of sizes, $s_i \equiv bt_i \mod p$. When Bob receives a block from Alice, he first multiplies it by $b^{-1} \mod p$, which converts the s_i's back to the t_i's and then solves the subset sum problem using the t_i's. Bob keeps the values of p and b secret.

Not so secure!

This system was very popular for a while since it is very fast to implement. However, in the early 1980's, Shamir, using Lenstra's fast linear programming algorithm, was able to peel away this disguise and obtain Bob's super-increasing set (or one that was equivalent to it). Many other methods for disguising the super-increasing set have been tried, but most of these have also been cracked. One method, known as the **Chor-Rivest** cryptosystem, uses finite field manipulations and this is still considered to be secure.