Fall 2005
Chris Christensen
Cryptology notes

The Knapsack Problem

In 1978, Merkel and Hellman published a public-key encryption system based upon the knapsack problem ("Hiding information and signatures in trapdoor knapsacks," *IEEE Transactions on Information Theory* IT-24(5), 525 – 530).

One way of thinking of the knapsack problem is to imagine having a collection of items for which the weight of each item is known. Out of your sight, another person selects some of the items and places them into a knapsack, and you are told the total weight of the knapsack. The knapsack problem is: knowing the weight of the knapsack, determine which items have been placed into it. (Of course, some conditions must be placed upon the weights of the items for there to be a unique solution.)

Here is a set of weights for a small collection of items.

{2, 5, 13, 21, 42, 84, 168, 336, 672, 1344}

Let us assume that after the items have been placed in the knapsack that the weight of the filled knapsack is 889. Which items were placed inside?

This is actually an easy problem because of the special nature of the set of weights. Notice that the item that weighs 1344 is too heavy to be in the knapsack, and the item that weighs 672 must be in the knapsack because the total of the remaining weights

$$2+5+13+21+42+84+168+336=671$$

is less than 672. We have accounted for 672 of the weight of the knapsack; items weighing 217 remain in the knapsack. We must use the item of weight 168; that leaves a weight of 49 in the knapsack. The item of weight 84 is too heavy. We must use the item of weight 42; that leaves a weight of 7 in the

knapsack. The items of weights 21 and 13 are too heavy. We must use the items of weights 2 and 5.

In the chart below, the weights of the items are in the second row, and the first row indicates which items are in the knapsack -- 1 denotes that the item is in the knapsack and 0 denotes that the item is not in the knapsack.

1	1	0	0	1	0	1	0	1	0
2	5	13	21	42	84	168	336	672	1344

The special nature of the set of weights – that each weight is larger than the sum of all the weights that are less than it – permitted us to easily solve the knapsack problem. Such a set of weights is called **superincreasing**.

The first row in the chart above gives us a clue as to how we may use a sequence of weights to encipher a message. Let us represent the letters of the alphabet by binary numbers (strings of `0`s and `1`s) using the following:

_	00000
a	00001
b	00010
c	00011
d	00100
e	00101
f	00110
g	00111
h	01000
i	01001
j	01010
k	01011
l	01100
m	01101
n	01110
o	01111
p	10000
q	10001
r	10010
s	10011
t	10100
u	10101
v	10110
w	10111
x	11000
y	11001
z	11010

where _ denotes a space.

The first row of the chart is a binary string of length 10 -- `1100101010`. By splitting it into two 5-bit (a bit is a **bi**nary dig**it**; i.e., a 0 or a 1) strings `11001` and `01010`, we could think of it as representing a digraph – in this case, `yj`. Notice that given the weight 889 we were led to a unique set of items in the knapsack – the items corresponding to the binary string

`1100101010`. We could, therefore, use the weight 889 to represent the digraph `yj`.

Reversing the process, given a digraph, say `th`, we could convert it to the 10-bit binary string `1010001000`. The string determines the items in the knapsack.

1	0	1	0	0	0	1	0	0	0
2	5	13	21	42	84	168	336	672	1344

which determines the weight $2+13+168=183$.

So, we have a correspondence between digraphs and weights of knapsacks. Each digraph could be encrypted as a weight; and, given a valid weight, a digraph can be determined.

The Key

Of course, this encryption is not secure because the superincreasing sequence can be solved. We need to transform this into a one-way trapdoor function.

Think back to multiplicative ciphers. There were not too practical (because there were only 12 of them), but the purpose of the multiplication and the modulus was to decimate the alphabet – to destroy the usual frequency patterns. We can play the same game here.

We choose for our modulus a prime that is larger than the maximum possible weight of the knapsack

$$2+5+13+21+42+84+168+336+672+1344=2687$$

2689 works.

Now choose a multiplier that has a multiplicative inverse modulo 2689. Recall that any integer that is relatively prime to 2689 would work. Say, 696. The multiplicative inverse of 696 modulo 2689 is 1387.

We convert the original superincreasing set of weights to a non-superincreasing set by multiplying each weight by 696 modulo 2689.

Superincreasing set of weights	Non-superincreasing set of weights
2	1392
5	791
13	981
21	1171
42	2342
84	1995
168	1301
336	2602
672	2515
1344	2341

The non-superincreasing set of weights is the public key; it is made known to all senders, The superincreasing set of weights, the modulus 2689 and the multiplicative inverse 1387 modulo 2689 form the private key; they are known only to the receiver.

Encryption Example

Say, we want to send the message `sean`.

First, we convert the message to a binary string.

s	e	a	n
1 0 0 1 1	0 0 1 0 1	0 0 0 0 1	0 1 1 1 0

Next, we split the string into two strings of ten bits each, and use the zeros and ones to select the items for the knapsack from the (public key) non-increasing set of weights.

		s						e		
1	0	0	1	1	0	0	1	0	1
1392	791	981	1171	2342	1995	1301	2602	2515	2341

which has weight 1392 + 1171 + 2342 + 2602 + 2341 = 9848.

		a						n		
0	0	0	0	1	0	1	1	1	0
1392	791	981	1171	2342	1995	1301	2602	2515	2341

which has weight 2342 + 1301 + 2602 + 2515 = 8760.

The ciphertext message is the weights `9848 8760`.

Decryption Example

What about decryption?

If the set of weights were superincreasing, decrypting the ciphertext would be easy. But, remember that we began with a superincreasing set and decimated it by multiplying by 696. Knowing the private key 1387 (the inverse of the multiplier), allows the receiver to convert back to a superincreasing set and decrypt the message.

For our example, we multiply each of the blocks in the ciphertext message by 1387 modulo 2689:

$$9848 \times 1387 \bmod 2689 = 1745$$

$$8760 \times 1387 \bmod 2689 = 1218$$

and use the superincreasing set of weights {2, 5, 13, 21, 42, 84, 168, 336, 672, 1344}.

		s					e			
1	0	0	1	1	0	0	1	0	1	
2	5	13	21	42	84	168	336	672	1344	

		a					n			
0	0	0	0	1	0	1	1	1	0	
2	5	13	21	42	84	168	336	672	1344	

Cryptanalysis

The cryptographer hopes that the security of this method depends upon the cryptanalyst being unable to break the message except by brute force – by trying all possible collections of objects in the knapsack. For two 5-bit strings, brute force would require trying $2^{10} = 1024$ ten-bit strings; for each object there are two choices -- either it is in the knapsack or not in the knapsack – therefore, there are $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^{10} = 1024$ collections of objects that could be in the knapsack. Of course, in practice, rather than a ten-bit string, a larger collection of objects (i.e., a longer string) would be used. For a 100-bit string, there would be

$$2^{100} = 1,267,650,600,228,229,401,496,703,205,376$$

collections of objects that could be in the knapsack.

The receiver has a trapdoor into the one-way knapsack problem; the receiver knows the private-key and can covert the knapsack problem to one with a set of weights that is superincreasing. In that case, the knapsack problem is easily solvable, and the message can be decrypted.

The system was thought to be secure because no algorithm is known to solve the public-key knapsack problem other than exhaustive search.

Unfortunately, the receiver's trapdoor also provided a trapdoor for the cryptanalyst. This system was shown not to be secure by Shamir in 1982 ("A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem," *Proceedings of the IEEE 23rd Annual Symposium on*

Foundations of Computer Science, 145 – 151). Shamir was able to convert the public-key weights to a superincreasing sequence -- not necessarily the private-key superincreasing sequence – but, still, the ciphertext could be broken using Shamir's superincreasing sequence.

Cryptanalysis of the original knapsack encryption system exhibits one of the problems faced by encryption systems that are based upon difficult mathematical procedures. It turned out that the security of the knapsack cryptosystem was not equivalent to the solution of the knapsack problem; there was an unexpected cryptanalysis based upon the solution of an easier problem.

Exercises

The following exercises use the public-key and private-key of the example above.

1. Decrypt the ciphertext `6455 3674`.

2. Ecnrypt the message `purdue`