Knapsack Public-Key Crypto-System 1978

Ralph Merkle
Berkeley → Stanford University

Martin Hellman
Stanford University

Published similar concept to Diffie-Hellmann system as a student at Berkeley University

"Secure communications over insecure channels"
Commun ACM, April 1978 (Berkeley Univ.), submitted Aug. 1975

Based on: Knapsack problem as a One-Way Function

Merkle-Hellman Knapsack

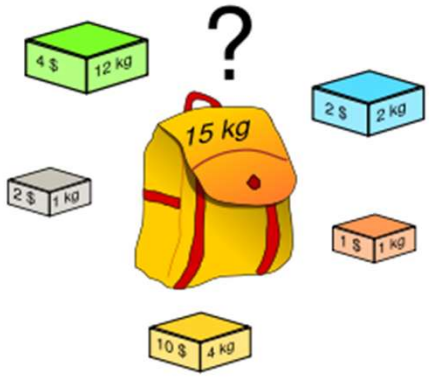
- One of first public key systems
- Based on NP-complete problem
- Original algorithm is weak
  - Lattice reduction attack
- Newer knapsacks are more secure
  - But nobody uses them…
  - Once bitten, twice shy

Merkle-Hellman Knapsack – Variations

- Merkle-Hellman additive knapsack cryptosystem
- Merkle-Hellman multiplicative knapsack cryptosystem
- Merkle-Hellman multipy-iterated knapsack cryptosystem

The Original Knapsack Problem

- Problem Definition
  - Want to carry essential items in one bag
  - Given a set of items, each has
    - A cost (i.e., 12kg)
    - A value (i.e., 4$)



- Goal
  - To determine the # of each item to include in a collection so that
    - The total cost is less than some given cost
    - And the total value is as large as possible

Types

- 0/1 Knapsack Problem
  - restricts the number of each kind of item to zero or one
- Bounded Knapsack Problem
  - restricts the number of each item to a specific value
- Unbounded Knapsack Problem
  - places no bounds on the number of each item

0/1 Knapsack Problem

- Problem: John wishes to take n items on a trip
  - The weight of item *i* is w_i & items are all different (0/1 Knapsack Problem)
  - The items are to be carried in a knapsack whose weight capacity is *c*
    - When sum of item weights ≤ c, all n items can be carried in the knapsack
    - When sum of item weights > c, some items must be left behind
- Which items should be taken/left?

0/1 Knapsack Problem (2)

- John assigns a profit p_i to item i
  - All weights and profits are positive numbers

- John wants to select a subset of the n items to take
  - The weight of the subset should not exceed the capacity of the knapsack (constraint)
  - Cannot select a fraction of an item (constraint)
  - The profit of the subset is the sum of the profits of the selected items (optimization function)
  - The profit of the selected subset should be maximum (optimization criterion)

- Let $x_i = 1$ when item i is selected and $x_i = 0$ when item i is not selected
  - Because this is a 0/1 Knapsack Problem, you can choose the item or not choose it.

Knapsack Problem

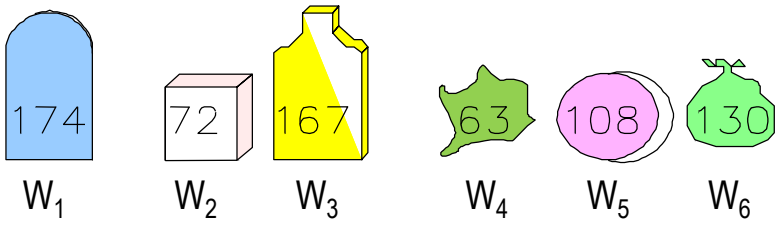
- Given a set of n weights $W_0, W_1, \ldots, W_{n-1}$ and a sum S, is it possible to find $a_i \in \{0,1\}$ so that

  $$S = a_0 W_0 + a_1 W_1 + \ldots + a_{n-1} W_{n-1}$$

  (technically, this is "subset sum" problem)

- **Example**
  - Weights $(62,93,26,52,166,48,91,141)$
  - Problem: Find subset that sums to $S = 302$
  - Answer: $62+26+166+48 = 302$
- The (general) knapsack is NP-complete

Example: Given the following 6 items, each with its own weight:



Problem: Given the total weight of the knapsack = $\sum_{i=1}^{n} w_i x_i$

Find the binary vector X = [x_1, x_2 .....], where x_i = {0,1}

Solution : X = [1 0 1 0 1 0]

- There is no algorithm known for finding X !!! (in the public literature)
- The solution is easy if the knapsack is superincreasing

Super- increasing

Each weight is greater than the sum of the weights of all of its predecessors in order

$w_2 > w_1$
$w_3 > w_1 + w_2$
$w_4 > w_1 + w_2 + w_3$ and so on ...

Activity - Is it super increasing ?

a. {1, 2, 4, 9, 20, 38}

b. {1, 2, 3, 9, 10, 24}

TWO KEYS

When the Knapsack Algorithm is used in public key cryptography, the idea is to create two different knapsack problems.

One is easy to solve, the other not.

Using the easy knapsack, the hard knapsack is derived from it.

The hard knapsack becomes the public key.

The easy knapsack is the private key.

The public key can be used to encrypt messages, but cannot be used to decrypt messages.

The private key decrypts the messages.

Procedure used to construct a knapsack cryptosystem

1. Generate a super increasing knapsack.

2. Convert the super increasing knapsack into a general knapsack.

3. The public key is the general knapsack.

4. The private key is the super increasing knapsack together with the conversion factors.

Example

Choose the super increasing knapsack
(2,3,7,14,30,57,120,251)

To convert the superincreasing knapsack into a general knapsack, we choose a multiplier m and a modulus n so that m and n are relatively prime and n is greater than the sum of all elements in the superincreasing knapsack.

For this example, we choose m=41 and n=491.

secret key is u = (m, u) = (41,491)
Where gcd (41,491)=1

The general knapsack is computed from the super-increasing knapsack by modular multiplication:

$2m = 2 \cdot 41 = 82 \bmod 491$

$3m = 3 \cdot 41 = 123 \bmod 491$

$7m = 7 \cdot 41 = 287 \bmod 491$

$14m = 14 \cdot 41 = 83 \bmod 491$

$30m = 30 \cdot 41 = 248 \bmod 491$

$57m = 57 \cdot 41 = 373 \bmod 491$

$120m = 120 \cdot 41 = 10 \bmod 491$

$251m = 251 \cdot 41 = 471 \bmod 491$

The public key is the general knapsack

Public key:

(82,123,287,83,248,373,10,471)

The private key is the super increasing knapsack together with the modular inverse of the conversion factor m, that is,

Private key:

(2,3,7,14,30,57,120,251)

Encrypt the message M=150

converts 150 to binary, 10010110.

Then she uses the 1 bits to select the elements of the general knapsack that are summed to give the cipher text.

82,123,287,83,248,373,10,471
1 ,0, 0, 1, 0, 1, 1, 0

C=82+83+373+10=548

Decrypt

The receiver of the cipher text uses *m* and *n* to calculate the *multiplicative inverse* of *m*, m^{-1}.

$m * m^{-1} \bmod n = 1$

Example:

$m^{-1} = 41^{-1} \bmod 491 = 12$

548 X 12 mod 491
193 mod 491

193 -120 = 73
73 – 57 = 16
16- 14 = 2
2-2 =0
2 3 7 14 30 57 120 251 = 10010110

Activity :

{1,2,4,10,20,40}

With multiplier 31
Modulo 110

Find public key and private key
Step 1: check the number to super increasing
Step 2: convert super increasing to general knapsack
- multiplier and modulo

a. conditions : both should be co-prime
modulo should be > sum of super increasing

b. Si_0 * multiplier mod n

Step 3: Encrypt using general knapsack
Step 4: Decrypt using super increasing knapsack

The normal knapsack sequence would be:

$1 \times 31 \bmod(110) = 31$

$2 \times 31 \bmod(110) = 62$

$4 \times 31 \bmod(110) = 14$

$10 \times 31 \bmod(110) = 90$

$20 \times 31 \bmod(110) = 70$

$40 \times 31 \bmod(110) = 30$

the public key is: {31, 62, 14, 90, 70, 30}
and
the private key is {1, 2, 4, 10, 20.40}.

Send a message that is in binary code:

100100111100101110

The knapsack contains six weights so we need to split the message into groups of six:

100100
111100
101110

This corresponds to three sets of weights with totals as follows

100100 = 31 + 90 = 121
111100 = 31+62+14+90 = 197
101110 = 31+14+90+70 =205

So the coded message is 121 197 205.

All you then have to do is multiply each of the codes 71 mod 110 to find the total in the knapsack which contains {1, 2, 4, 10, 20, 40} and hence to decode the message.
The coded message is 121 197 205:

121×71 mod(110) = 11 = 100100
197×71 mod(110) = 17 = 111100
205×71 mod(110) = 35 = 101110

The decoded message is:
100100111100101110.
Just as I thought!
Simple and short knapsack codes are far too easy to break to be of any real use. For a knapsack code to be reasonably secure it would need well over 200 terms each of length 200 bits.