Seminar: IT Security and Cryptography

Knapsack Public Key Cryptosystem

Dinara Barshevich

July 7, 2005

Abstract

In this work we present so-called Knapsack-Based Public Key Cryptosystems and in particular Merkle - Hellman Cryptosystem .

Contents

Introduction

We present the Knapsack Public Key Cryptosystems that were first implemented in 1978 by R. Merkle and Hellmann and at once became quite popular because of its high speed and elegance.

However, several attacks were made to break it that finally resulted in crash of Merkle - Hellman Cryptosystem and almost all Knapsack-Based Cryptosystems.

The paper consists of 4 main sections. In Section 1 we get acquainted with Knapsack problem itself and find out how it can be used in constructing Public Key Knapsack Cryptosystem. Section 2 deals with the Knapsack Cryptosystem and describes the method used in constructing it. Section 3 studies the algorithm analysis and consists of 2 subsections. In Subsection 3.1 we perform the main features of the comparison between RSA and Merkle - Hellman Cryptosystem. In Subsection 3.2 we discuss the problem of its security. In the last Section 4 we perform two basic attacks that were made on this Cryptosystem and led to its fall.

1 Knapsack Problem: setting, comlexity and basic analyses

1.1 Problem Setting

The Knapsack Problem is one of the well - known discrete mathematics problems. The problem that is used in constructing Cryptosystem being discussed is closely realted to it but still is exactly called SubSet - Sum Problem.

The problem is, given an input of positive integers $a_1, \dots, a_n$ and s, to determine boolean integers $x_1, \dots, x_n$ such that the following holds:

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

If one looks at the $a_1, \dots, a_n$ as at the weights of items and at s as the capacity of the knapsack then the problem is to find such a subset of these items that exactly fills the knapsack.

1.2 Possible Algorithms of Solving Knapsack

The main observation on the problem that helps to use it in Cryptography is that general knapsack problem is known to be NP-complete and so it is believed to be hard to solve.

The trivial algorithm of solving the general Knapsack problem of n items is to consider all possiblities that will obviously take $O(2^n)$ time.

The better algorithm for solving the general Knapsack problem is the following.

1. Compute
   $S_1 = \left\{ \sum_{j=1}^{\lfloor n/2 \rfloor} x_j a_j \,|x_j \in \{0,1\} \right\}$
   $S_2 = s - \left\{ \sum_{j=\lfloor n/2 \rfloor+1}^{n} x_j a_j \,|x_j \in \{0,1\} \right\}$
2. Sort them and scan for a common member. The common member y exists iff the corresponding $x_j, j = 1 \dots n$ give the solution:
   if $y = \left\{ \sum_{j=1}^{\lfloor n/2 \rfloor} x_j a_j \,|x_j \in \{0,1\} \right\} = s - \left\{ \sum_{j=\lfloor n/2 \rfloor+1}^{n} x_j a_j \,|x_j \in \{0,1\} \right\}$ then $s = \sum_{j=1}^{n} x_j a_j$

It can be easily seen that the algorithm above demands $O(n2^{n/2})$ time and $O(2^{n/2})$ storage space.

Surprisingly enough, it's still the fastest existing algorithm for solving general Knapsack Problem.

1.3 Easy-solvable Knapsacks

Though as we have seen the general Knapsack is believed to be hard to solve, if we restrict the problem somehow, we can get an easy solvable problem.

Let us consider the so-called super-increasing Knapsack that is the Knapsack with super-increasing sequence of weights:

$$a_j > \sum_{i=1}^{j-1} a_i, 2 \leq j \leq n.$$

In this case the problem turns out to be linear-time solvable if we consider the following algorithm:

```
for j = n downto 1 do
if $s \geq a_j$ then $\{x_j = 1; s = s - a_j\}$ else $x_j = 0$;
return $(x_1, \ldots, x_n)$.
```

It is easy to see that the algorithm always gives a unique solution if it exists.

2 Description of the Knapsack Public Key Cryptosystem

Every Public - Key Cryprosystem consists of 3 main procedures:

- Public and Private Key Generation that is made by the receiver
- Encoding Procedure that is made by the sender by means of Public Key
- Decoding Procedure that is made by the receiver by means of Private Key

Merkle - Hellmann Algorithm determines the details of these procedures.

- Key Generation.
  1. Start with a super-increasing sequence of weights $b_1, \ldots, b_n$, where n is a security parameter such that
  $$b_1 \sim 2^n, b_j > \sum_{i=1}^{j-1} b_i, 2 \leq j \leq n, b_n \sim 2^{2n}$$
  2. Choose M and W such that $M > \sum_{i=1}^{j-1} b_i$ and $\gcd(M, W) = 1$ by modulus M
  3. Compute $a_j^1 \equiv b_j \dot{W} \bmod M, 0 < a_j^1 < M$. It is easy to see that a_j^1 will be nonzero.
  4. Select arbitrary permutation π of $\{1, \ldots, \}$ and define $a_j = a_{\pi(j)}^1 \forall j = 1, \ldots, n$

  Conclusively, the public key is $\{a_1, \ldots, a_n\}$ and the private key is $\{b_1, \ldots, b_n, M, W, \pi\}$
- Encoding Procedure:

  If the sender wishes to send a message, he first of all shifts it to the pieces of length n and encodes every piece $\{x_1, \ldots, x_n\}$ as $s = \sum_{j=1}^{n} x_j a_j$

- Decoding Procedure:

  The receiver computes:

  $c \equiv s\dot{W}^{-1} \bmod M,\ 0 \le c < M \Longrightarrow c \equiv \sum_{j=1}^{n} x_j a_j W^{-1} \equiv \sum_{j=1}^{n} x_j a^1_{\pi(j)} W^{-1} \equiv$

  $\equiv \sum_{j=1}^{n} x_j b_{\pi(j)} \bmod M.$

  Since $0 \le c < M$ and $0 < \sum_{j=1}^{n} x_j b_j < M$, we get $c = \sum_{j=1}^{n} x_j b_{\pi(j)}$.

  The trick is that since $\{b_1, \dots, b_n\}$ is a auper-increasing sequence, the receiver faces the knapsack that is easy to solve.

The algorithm presented above is called singly iterated Merkle - Hellmann algorithm comparing to multiply iterated algorithm. It is constructed similarly but uses several iterations of public and private key generation. That was considered to make it more secure, but was as well broken several years after its construction.

The Key Generation in this case includes the following steps:

Let $M_1 = M, W_1 = W, a_j^{(0)} = a^1_{\pi(j)}$.

Then $M_k, W_k : M_k > \sum_{j=1}^{n} a_j^{(k-1)}, \gcd(M_k, W_k) = 1$ and $a_j^{(k)} = a_j^{(k-1)} W_k \bmod M_k$.

3 Algorithm Analyses

3.1 Comparing to RSA

Merkle - Hellmann algorithm was proposed just a little later after the well-known Public - Key RSA algorithm. So it is often compared with it. The basic results of comparison are the following:

- Merkle - Hellman algorithm with n as a security parameter is about 100 times faster than RSA with $m = p\dot{q}$ when n is about 100 bits and m is about 500 bits.
- In Merkle - Hellman algorithm n bits are encoded in $2n$ bits whlie in RSA n bits are encoded into the same n bitswith n.
- Public Key of Merkle - Hellman algorithm is of size $2n^2$ bits while RSA's Public Key is of $2m$ bits.
- Merkle - Hellman algorithm assumes $P \neq NP$, while RSA assumes factorization is in NP

3.2 Algorithm Security

From the very beginning of its existence many doubts were claimed about the security of Merkle - Hellman algorithm. These are the basic doubts:

- What if $P = NP$?

  It is clear that the crucial moment in security of the algorithm is the suggestion that $P \neq NP$
- What if most instances of knapsacks used by the algorithm are easy to solve?

  The algorithm assumes that general knapsack is hard to solve but it is concerned the worst cases and not the average. So if one can just solve the encrypted knapsack, any eavesdropper can read the plaintext.

- What if one can deduce from the public Knapsack what the construction method is? When one faces the public knapsack, it is necessary for the security that one cannot find the way to decrypt an easy solvable knapsack from it.
- This doubt is based on the interesting result of Brassard who proved that if breaking a cryptosystem is NP-hard, then $NP = Co-NP$. Using it we get that if $NP \neq Co-NP$, then breaking Merkle - Hellman algorithm cannot be NP-hard. So it is likely to be easier to break it than to break the general Knapsack problem.
- Linearity of the equation always causes some suspicions about the method security. For example, considering the equation by modulus 2:
  $s = \sum\limits_{j=1}^{n} x_j a_j \mod 2,$
  we get a single bit of information about the plaintext. Nobody could ever makes use of it but still this knowledge is considered doubtful for security.

As a result of all these general suspicions, several attacks were proposed to break Merkle - Hellman Cryptosystem that finally led to its total crash.

4 Attacks to Merkle - Hellman Cryptosystem

There are two basic types of attacks used to break the system. The first of them relies on the proposal that modular multiplication used to convert public knapsack into private one, doesn't disguise the private knapsack that is easy to solve securely enough. The second one tries to solve the general knapsack under certain conditions so that one can solve public knapsack without knowing private one.

4.1 Shamir polynomial algorithm for the singly-iterated Merkle-Hellman, 1982

Here we will discuss the attck to singly iterated Merkle - Hellman algorithm as the attack to the multiply iterated is rather similar.

Let $U \equiv W^{-1} \mod M$. Then $b_{\pi(j)} \equiv a_j U \mod M$ for $\forall j = 1, \ldots, n$.

So for some $k_j \in Z$ $a_j U - k_j M = b_{\pi(j)}$ for $\forall j = 1, \ldots, n$. Hence $\frac{U}{M}$ - $\frac{k_j}{a_j} = \frac{b_{\pi(j)}}{a_j M}$.

This means that all of the $\frac{k_j}{a_j}$ are close to $\frac{U}{M}$. Due to the choice of public knapsack weights in Merkle - Hellman algorithm we see $b_1, \ldots, b_q \sim 2^n$ when q is small enough.

Thus, if we put $j_i = \pi_i^{-1}$ we can get $|k_{j_i} - a_{j_1} - k_{j_1} a_{j_i}| \sim 2^n$. Now if we apply Lenstra's theorem that claims, that the integer programming problem in a fixed number of variables can be solved in polynomial time, we can get k_{j_i} for $i = 1, \ldots, q$. By means of these integers we can construct a pair (U^1, M^1): $\frac{U^1}{M^1}$ close to $\frac{U}{M}$ such that if we compute the weights c_j by the equation: $c_j \equiv a_j U^1 \mod M^1$, $0 < c_j < M^1$, $j = 1, \ldots, n$, they will form a super increasing sequence of weights when arranged in increasing order.

As it can be easily seen, these c_j can be used to decrypt the message by solving an easy knapsack.

The last difficulty is to find secret $j_1, \ldots, j_q$ as permutation π is private. But it can be done polynomially by considering all possible variants of them by using the fact q is a constant.

4.1.1 Difficulties of Shamirs method

The crucial tool in the attack was Lenstras result on integer programming in a fixed number of variables. However, Lenstras algorithm running time is given by a high degree polynomial.

Thus, it has never been implemented in practice. Continued fraction method can be used instead of Lenstras result, but when the b_j are large enough, it fails.

4.2 Lagarias and Odlyzko Attack, 1983

This attack provides a method to solve the general knapsack problem when weights are large enough. This kind of knapsacks are also called low- density knapsacks. The attack uses the lattica theory, so we will discuss the basic notions of it.

An integer lattice L is an additive subgroup of Z^n that contains n linearly independent vectors over R^n. A basis $(v_1 \dots, v_n)$ of L is a set of elements of L such that $L = \{z_1v_1 + \dots + z_nv_n : z_i \in Z\}$

The problem of the Shortest Lattice Vector is given a basis of the lattice L as an input to find the shortest non-zero vector of L.

The problem is believed to be quite hard, yet it was not proved.

Lovasz proposed an algorithm of constructing the so-called Lovasz- reduced basis $(v_1, \dots, v_n)$ from some basis where $||v_1|| \leq 2^{n-1} \min\{||x||^2 | x \in L, x \neq 0\}$.

The low density attack itself is the following.

Given the a_i as the public key and s as a cipher text, we form the $(n+1)$-dimensional lattice with basis $\begin{matrix} 1 & 0 & \dots & 0 & -a_1 \\ 0 & 1 & \dots & 0 & -a_2 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & \dots & 0 & -a_1 \\ 0 & 0 & \dots & 1 & -a_n \\ 0 & 0 & \dots & 0 & s \end{matrix}$.

Denoting the matrix by $\begin{matrix} v_1 \\ v_2 \\ \dots \\ v_n \\ v_{n+1} \end{matrix}$, we get the following equation: $\sum_{i=1}^{n} x_i a_i + v_{n+1} = (x_1, x_2, \dots, x_n, 0)$ if $\{x_j | j = 1, \dots, n\}$ form the solution of the public knapsack problem.

The miracle is that since $x_j = 0$ or 1 for $j = 1, \dots, n\}$, the vector is very short.

Therefore, if we run the Lovasz lattice basis reduction algorithm on the basis and check if the resulting reduced basis contains a vector that is a solution or not we can presume that we will find the right solution to the public knapsack.

It is proved that we can solve knapsacks with $a_j \sim 2^{n^2}$ that is obviously extremely large! But in practice it turns out much better.

Conclusion

As we have seen, doubts about the security of the knapsack cryptosystems turn out real. Most cryptosystems were broken but still some of them (e.g. Chor - Rivest Cryptosystem) remain unbroken up to now.

Nevertheless, due to the high speed of the algorithm, the fact that factorization and logarithm procedures can turn out efficiently solvable someday and elegance of the algorithm itself search of the cryptosystem that use knapsack problem inside is going on.

References

1 L.M. Adleman "On Breaking Generalized Public Key Cryptosystem" Proc of the 15 ACM, 1983,pp. 402-412

2 E.F. Brickell "Solving low density knapsacks" Crypto,83, 1984, pp.25-37

3 E.F. Brickell "The Cryptoanalysis of Knapsack Cryptosystems" SIAM, 1988, pp.3-23

4 A. Shamir "A Polynomial time Algorithm for Breaking the Basic Merkle - Hellman Cryptosystem" IEEE, v.IT-30, 1984 pp.699-704