数理解析研究所講究録 1325 巻 2003 年 128-133

別の数え上げ符号を用いたナップザック暗号
Density Attack and Enumerative Source Encoding on Knapsack Cryptosystems

大村 慶二
Keiji Oomura
東京工業大学大学院情報理工学研究科数理・計算科学専攻
Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology

田中 圭介
Keisuke Tanaka
東京工業大学大学院情報理工学研究科数理・計算科学専攻
Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology

1 Introduction

Several knapsack-type public-key cryptosystems have been shown to be insecure. In particular, Brickell [2] found a way to break the general Merkel–Hellman scheme. However A different attack is the *low-density* attack of Lagarias and Odlyzko [8]. The density of a knapsack is defined as the ratio of the number elements in it to the size in bits of these elements. There are several schemes based on discrete logarithms, which are secure against low-density attack. Typical examples are the Chor-Rivest system [4] and the Okamoto–Tanaka–Uchiyama scheme [12, 13]. We propose a new density attack which can be applied to the Chor–Rivest scheme and the Okamoto–Tanaka–Uchiysma Scheme. According to this attack, these schemes become no longer secure if we use these schemes naively. We also propose a new scheme, which makes the Chor–Rivest cryptosystem and the Okamoto–Tanaka–Uchiyama cryptosystem secure against the Lagarias–Odlyzko attack.

2 Previous Low-Density Attack

Subset Sum Problem

Given: $A = \{a_i \in Z : 1 \leq i \leq n\}$, $M \in Z$.
Question: Is the sum of the elements in some subset of A equal to M?

0-1 Integer Programming Problem

Given: $A = \{a_i \in Z : 1 \leq i \leq n\}$, $M \in Z$.
Find x: $\sum_{i=1}^{n} a_i x_i = M$, $x_i = \{0, 1\}$.

The subset sum problem is to decide whether or not the 0-1 integer programming problem has a solution. This problem is NP-complete, and the difficulty of solving it is the basis of public-key cryptosystems of knapsack type. It converts the problem to one of finding a particular short vector v in a lattice, and then to attempt to find v we uses a lattice basis reduction algorithm[10] due to A. K. Lestra, H. W. Lenstra,Jr., and L. Lovász.

We briefly review two algorithms which would solve the subset-sum problem with some densities in polynomial time by finding the shortest non-zero vectors in lattices. One is the Lagarias–Odlyzko algorithm and the other is its improved algorithm.

definition 1 *A lattice $L \subset \boldsymbol{R}^n$ such that*

$$L = \left\{ \sum_{i=1}^{n} x_i b_i \middle| x_i \in \boldsymbol{Z}, i = 1, \ldots, n \right\},$$

$b_1, \ldots, b_n \in \boldsymbol{R}^n$ is a liner independent. $B = (b_1, \ldots, b_n) \subset \boldsymbol{R}^{n \times n}$ is the basis of $L = L(B)$.

The general subset sum problem is NP-complete. However, there are two algorithms, one due to Brickell [1] and the other to Lagarias and Odlyzko [8], which in polynomial time solve almost all subset-sum problems of sufficiently low-density. Both methods rely on basis reduction algorithms to find short non-zero vectors in special lattices. The Lagarias and Odlyzko algorithm would solve almost all subset sum problems of density $< 0.6463\ldots$ in polynomial time if it could invoke a polynomial time algorithm for finding the shortest non-zero vector in a lattice. Coster and Joux and so on [5] improved the Lagarias–Odlyzko algorithm, which would solve almost all subset sum problem of density $< 0.9408\ldots$ if it exists a lattice oracle which can find the shortest non-zero vectors in a lattice.

definition 2 *The density of weights $a_1, \ldots, a_n$ is defined by $d(a) = \frac{n}{\log_2 \max_i a_i}$.*

2.1 Lagarias–Odlyzko Algorithm

Lagarias and Odlyzko show that if the density is bounded by $0.6463\ldots$, the lattice oracle is guaranteed to find the solution vector with high probability.

theorem 1 (***Lagarias–Odlyzko***[8]) *Let A be a positive integer, and let $a_1, \dots, a_n$ be random integers with $0 < a_i \leq A$ for $1 \leq i \leq n$. Let $\mathbf{e} = (e_1, \dots, e_n) \in \{0,1\}^n$ be arbitrary, and let $s = \sum_{i=1}^{n} e_i a_i$. If the density $d < 0.6463\dots$, then the subset sum problem defined by $a_1, \dots, a_n$ and s may 'almost always' be solved in polynomial time with a single call to a lattice oracle.*

theorem 2 *For all $n \geq 1$, $S_n(\beta n) \leq 2^c$,*

$$c = \min_{u \in R} (\log_2 e)\delta(u, \beta),$$

$$\delta(u, \beta) = u\beta + \ln \theta(e^{-u}), \theta(z) = 1 + 2\sum_{k=1}^{\infty} z^{k^2}.$$

Proof. From [8], $S_n(\beta n) \leq e^{n\delta} = 2^{(\log_2 e)\delta(\beta,u)n}$. If the minimum value of $e^{n\delta} = 2^{(\log_2 e)\delta(\beta,u)n}$ is taken, c can be found. ◇

The low-density attack of Lagrias and Odlyzko is performed by the following Algorithm.

algorithm 1 Logarias-Odlyzko Algorithm$(a_1, ..., a_n, s)$

1 Input : $a_1, ..., a_n, s$; Output : $e_1, ..., e_n$.

2 Choose $N > \sqrt{n}$.

3 Make the lattice with the following vectors,

$$\begin{aligned} b_1 &= (1, 0, ..., 0, -Na_1), \\ b_2 &= (0, 1, ..., 0, -Na_2), \\ &\dots \\ b_n &= (0, 0, ..., 1, -Na_n), \\ b_{n+1} &= (0, 0, ..., 0, Ns). \end{aligned}$$

4 We want to find the shortest non-zero vector out of this lattice. Using LLL algorithm, find the vector $v = (v_1, ..., v_{n+1})$. If $s = \sum_{i=1}^{n} a_i v_i$, it will become $v = e$.

2.2 Improved Low-Density Subset Sum Algorithms

Improved of Lagarias–Odlyzko algorithm,almost all subset sum problem of density < 0.9408... can be solved.

theorem 3 [5] *Let A be a positive integer, and let $a_1, ..., a_n$ be random integers with $0 < a_i \leq A$ for $1 \leq i \leq n$. Let $\mathbf{e} = (e_1, ..., e_n) \in \{0,1\}^n$ be arbitrary, and let $s = \sum_{i=1}^{n} e_i a_i$. If the density $d < 0.9408\cdots$, then the subset sum problem defined by $a_1, ..., a_n$ and s may 'almost always' be solved in polynomial time with a single call to a lattice oracle.*

The improved low density attack is performed by changing the vector b_{n+1} of algorithm 1 to $b_{n+1} = (\frac{1}{2}, ..., \frac{1}{2}, Ns)$.

In almost all subset sum problems of density $d < 0.9408\dots$, the solution vector $v = e$ we searched for is the shortest nonzero vector in the lattice. One way to improve the bound presented above would be to show that it is possible to cover the vertices of the n-cube with a polynomial number of n–spheres of radius $\sqrt{\alpha n}$ with $\alpha < \frac{1}{4}$. But, according to proposition 1, any n-sphere of radius $\sqrt{\alpha n}$ with $\alpha < \frac{1}{4}$ can cover only an exponentially small fraction of the vertices of the n-cube. So, it is impossible to improve the bound of density.

proposition 1 [5] *Any sphere of radius $\sqrt{\alpha n}, \alpha < \frac{1}{4}$, in R^n contains at most $(2 - \delta)^n$ points of $\{0,1\}^n$, for some $\delta = \delta(a) > 0$.*

3 More Precise Analysis on Density Attack

In the cases where the subset sum problem to be solved is known to have $\sum_{i=1}^{n} e_i$ small (as occurs in some knapsack cryptosystems, such as the Chor–Rivest[4]), it is possible to improve Lagarias–Odlyzko algorithm [8]. If we know that $\sum_{i=1}^{n} e_i \leq \beta n$ for $0 < \beta \leq \frac{1}{4}$, we can solve the subset-sum problem of density above 0.9408... using the Lagrias–Odlyzko algorithm.

proposition 2 *One n-sphere beyond a radius $\sqrt{\beta n}$ centered at $c = (0, 0, ..., 0)$ can cover the points of $e \in \{0,1\}^n$ for $\sum_{i=1}^{n} e_i \leq \beta n$.*

Proof. The distance h of $c = (0, 0, ..., 0)$ and the points $e = (e_1, ..., e_n)$ with $\sum_{i=1}^{n} e_i \leq \beta n$ is,

$$\begin{aligned} h &= \|c - e\| \\ &= \|e\| \\ &\leq \sqrt{\beta n}. \end{aligned}$$

◇

theorem 4 *Let A be a positive integer, and let $a_1, ..., a_n$ be random integers with $0 < a_i \leq A$ for $1 \leq i \leq n$. $e = (e_1, ..., e_n) \in \{0,1\}^n$ is set to $\sum_{i=1}^{n} e_i \leq \beta n$ for $0 < \beta \leq \frac{1}{4}$, and s is set to $s = \sum_{i=1}^{n} e_i a_i$. If density $d < d_0$, which is is described below, then the subset sum problem defined by $a_1, ..., a_n$ and s may 'almost always' be solved in polynomial time with a single call to a lattice oracle. The value of d_0 is,*

$$d_0 = \max_{u \in R} \frac{1}{(\log_2 e)\delta(u, \beta)},$$

$$\delta(u, \beta) = u\beta + \ln \theta(e^{-u}), \quad \theta(z) = 1 + 2\sum_{k=1}^{\infty} z^{k^3}.$$

Proof. We can prove this theorem by improving theorem 1. Now, we are interested in vectors $\hat{x} = (x_1, ..., x_{n+1})$ which satisfy,

$$\begin{cases} \|\hat{x}\| \leq \|e\|, \\ \hat{x} \in L, \\ \hat{x} \notin \{0, e, -e\}. \end{cases} \qquad (1)$$

We know $\sum_{i=1}^{n} e_i \leq \beta n$, so $\|e\| \leq \beta n$. We show that probability P that a lattice L contains a short vector which satisfies equation 1 is,

$$
\begin{aligned}
P &= \Pr(\exists x \text{ which satisfies equation (1)}) \\
&\le \Pr\Big(\exists x, y \ s.t. \|x\| \le \|e\|, |y| \le n\sqrt{\frac{1}{2}}, \\
&\qquad x \notin \{0, e, -e\}, \sum_{i=1}^{n} x_i a_i = ys\Big) \\
&\le \Pr\left(\sum_{i=1}^{n} x_i a_i = ys \,|\, 0 < \|x\| \le \|e\|, |y| \le n\sqrt{\frac{1}{2}}, x \notin \{0, e, \cdot\right. \\
&\quad \cdot \left|\{x \mid \|x\| \le \|e\| \ \le \sqrt{\beta n}\}\right| \cdot \left|\left\{y \,|\, |y| \le n\sqrt{\frac{1}{2}}\right\}\right| \\
&\le n\Big(2n\sqrt{\frac{1}{2}n} + 1\Big)\frac{1}{A} \cdot \left|\{x \mid \|x\| \le \|e\| \ \le \sqrt{\beta n}\}\right|.
\end{aligned} \tag{2}
$$

From theorem 2, $\left|\{x \mid \|x\| \le \|e\| \ \le \sqrt{\beta n}\}\right| \le 2^{c_1}$. such that

$$c_1 = \min_{u \in R}(\log_2 e)\left\{u\beta + \ln\left\{1 + 2\sum_{k=1}^{\infty}(e^{-u})^{k^2}\right\}\right\}.$$

Then,

$$P \le n\Big(2n\sqrt{\frac{1}{2}n} + 1\Big)\frac{2^{c_1}}{A}.$$

When $A > 2^{c_1 n}$, we have $\lim_{n \to \infty} P = 0$, and the density d is as follows,

$$d = \frac{n}{\log_2 \max_{1 \le i \le n} a_i} = \frac{n}{\log_2 A} < \frac{n}{\log_2 2^{c_1 n}} = \frac{1}{c_1} = d_0.$$

$\diamond$

The density d_k of the multiplicative knapsack scheme based on discrete logarithm is dependent on the number n of the public key and β which is the value so that $\sum_{i=1}^{n} e_1 \le \beta n$ for encoded text $e = (e_1, ..., e_n)$. So,

$$d_k = \frac{n}{\beta n \log n} = \frac{1}{\beta \log n}.$$

Moreover from theorem 4, about the subset sum problem which $\sum_{i=1}^{n} e_i \le \beta n$, the density d_a for which it can be solved is decided by β. Then,

$$d_a = \max_{u \in R} \frac{1}{(\log_2 e)\delta(u, \beta)}.$$

If $d_a - d_k > 0$, the attack of the multiplicative knapsack scheme based on discrete logarithm can be succeeded.

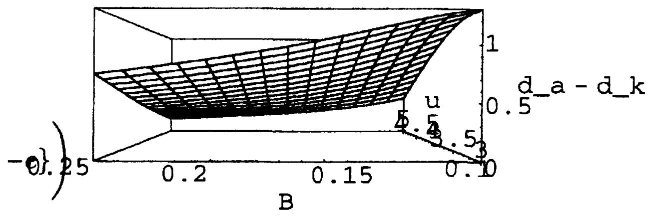
theorem 5 *For* $0 < \exists n_0 \le n$, $0 < \exists \beta_0 \le \beta \le \frac{1}{4}$,

$$d_a - d_k > 0.$$

Proof. The value of d_k depends on n and β, and d_a depends on β. We have

$$d_k = \frac{1}{\beta \log n} < \frac{1}{\beta \log n_0} \le \frac{1}{\beta_0 \log n_0 g}.$$

So, if $d_a - \frac{1}{\beta \log n_0} > 0$, then $d_a - d_k > 0$. For example, setting $n_0 = 64$, $\beta_0 = \frac{1}{16}$, we have the graph $z = d_a - d_k = \frac{1}{(\log_2 e)\delta(u,\beta)} - \frac{1}{\beta \log n_0}$ as follows.



This graph is drawn in the range of $0.0625 \le B \le 0.25$, $2 \le u \le 4.7$, $0 \le z \le 1.3$. If $n = 256, \beta_0 = 0.0125$, the subset sum problem of density $d \le 10$ become insecure since $b_a - b_k > 0$. $\diamond$

4 Proposed Scheme for Enumerative Source Encoding

Some knapsack scheme based on discrete logarithm use the coding translating unconstrained binary text into 0-1 vectors with length p and weight h. We propose the coding translating a text into vectors with length p and weight h. This coding is introduced by powerline system [11, 3], however the explanation is not given clearly about the coding scheme. Moreover we propose the coding which translating a text into the vectors with the length p and weight from k to $h (> k)$. We show that using the vectors whose weights has from k to h, we can make knapsack schemes more secure than the previous schemes.

4.1 Enumerative Source Encoding

This section describes a simple procedure for translating an unconstrained binary text into the $0-1$ vectors. Given a binary text, we first break it into blocks of $\lfloor \log_2 \binom{p}{h} \rfloor$ bits each. Each such block is viewed as the binary representation of a number n $(0 \le n < \binom{p}{h})$. To map these numbers into binary vectors with weight h, we use the order preserving mapping induced by the lexicographic order of the vectors and the natural order of the integers. If n is larger than $\binom{p-1}{h-1}$, the first bit in the corresponding vector is set to 1. Otherwise, the first bit is set to 0. We then update p and h, and iterate p times, until all p bits are determined.

algorithm 2 Transforming a number n into a binary vector M

```
Input s, p, h; Output: M = (m_1, m_2, ..., m_p)
for i ← 1 to p do
    if i ← 1 then
        m_i ← 1
        s ← s − C(p−1, h−1)
        h ← h − 1
    else then m_i ← 0
Return M
```

The inverse transformation, which is the last step in decryption, is as follows.

algorithm 3 Transforming a binary vector M into a number s

Input: $M = (m_1, m_2, ..., m_p), p, h$; Output:$s$
$n \leftarrow 0$ **do**
for $i \leftarrow 1$ **to** p **do**
  if $m_i = 1$ **then**
    $s \leftarrow s + \binom{p-i}{h}$
    $h \leftarrow h - 1$
Return s

For efficient implementation, the $\frac{p \cdot h}{h}$ binomial coefficients preceding $\binom{p}{h}$ (in the Pascal triangle) will be precomputed and permanently stored. The above indexing scheme is well known in the literature [6].

4.2 Proposed Scheme for Enumerative Source Encoding

This section describes a procedure[9] for translating an unconstrained binary text into the form which is vectors $M = (m_p, ..., m_1)$ for $m_i \in \{0, h\}(1 \leq i \leq p)$ and $\sum_{i=1}^{p} m_i = h$.

fact 1

$$\binom{n}{m} = \sum_{i=0}^{m} \binom{n-l-i}{m-i}.$$

fact 2 *The number of vectors* $m = (m_p, ..., m_1)$ *with* $\sum_{i=1}^{p} m_i = k$ *is*

$$\binom{p+k-1}{k}.$$

Given a binary text, we first break it into blocks of $\lfloor \log_2 \binom{p+h-1}{h} \rfloor$ bits each. Each such block is viewed as the binary representation of number n $(0 < n \leq \binom{p+h-1}{h})$. To map these numbers into non-negative integer vectors with weight h, we use the following algorithm [7].

algorithm 4 EncodingWithTheWeight(s, p, h)

Input: (s, p, h); Output: $M = (m_p, m_{p-1}, \ldots, m_1)$
$M \leftarrow (0, 0, \ldots, 0)$
$h' \leftarrow h$
$stage \leftarrow p$
while (stage ≥ 2) **do**
  $t \leftarrow s - \binom{stage-2+h'}{stage-2}$
  if $t > 0$ **then**
    $m_{stage} \leftarrow m_{stage} + 1$
    $s \leftarrow t$
    $h' \leftarrow h' - 1$
  else if $t = 0$ **then**
    $m_1 \leftarrow h'$
    $stage \leftarrow 1$
  else if $t < 0$ **then**
    $stage \leftarrow stage - 1$
Return M

The algorithm for transforming a non-negative integer vector M into a number s is as follows.

algorithm 5 DecodingWithTheWeight(M, p, h)

Input: $M = (m_p, m_{p-1}, \ldots, m_1), p, h$; Output: s
$h' \leftarrow h$
$s' \leftarrow 0$
for $stage \leftarrow p$ **downto** 2 **do**
  if $m_{stage} \neq 0$ **then**
    $s' \leftarrow s' + \sum_{i=0}^{m_{stage}-1} \binom{stage-2+h'-i}{stage-2}$
    $h' \leftarrow h' - m_{stage}$
Return s'+1

In algorithms 2, 3 for transforming 0-1 vectors, the number of calculation of binomial coefficient is at most $p-1$. Otherwise, in these algorithms 4, 5, the number of calculation of binomial coefficient is at most $p - 1 + h$. Therefore, these algorithms are quite fast. little become bad.

example 1 *In the case of* $p = 3, h = 3$, *we have*

$$\begin{array}{ll} 1 = (0,0,3) & 6 = (1,1,1) \\ 2 = (0,1,2) & 7 = (1,2,0) \\ 3 = (0,2,1) & 8 = (2,0,1) \\ 4 = (0,3,0) & 9 = (2,1,0) \\ 5 = (1,0,2) & 10 = (3,0,0). \end{array}$$

Next, we propose the coding which improves the algorithm above. We change the number into the vector of the length p and the weight k to h. In order to encode a text, we divide it for $\lfloor \log_2 \sum_{i=k}^{h} \binom{p-1+i}{p-1} \rfloor$ bits of every blocks. This algorithm to encode $(m_p, m_{p-1}, ..., m_1)$ of $k \leq \sum_{i=1}^{p} m_i \leq h$ is as follows:

$$\begin{aligned} (0,0,...,0,h) &= 1, \\ (0,0,...,1,h-1) &= 2, \\ &\cdots \\ (h,0,...,0,0) &= \binom{p-1+h}{p-1}, \\ (0,0,...,0,h-1) &= \binom{p-1+h}{p-1} + 1, \\ (0,0,...,1,h-2) &= \binom{p-1+h}{p-1} + 2, \\ &\cdots \\ (h-1,0,...,0,0) &= \binom{p-2+h}{p-1} + \binom{p-1+h}{p-1}, \\ (0,0,...,0,h-2) &= \binom{p-2+h}{p-1} + \binom{p-1+h}{p-1} + 1, \\ &\cdots \\ &\cdots \\ (k,0,...,0,0) &= \sum_{i=k}^{h} \binom{p-2+i}{p-1}. \end{aligned}$$

Next **ImprovedEncoding(s, p, h)** algorithm finds the value of $\sum_{i=1}^{p} m_i$ which can be encoded from input s. If $\sum_{i=1}^{p} m_i$ are known to input s, **EncodingWithTheWeight(s, p, h)** algorithm encodes input s.

algorithm 6 ImprovedEncoding(s, p, h)

Input: s, p, h; Output: $M = (m_p, m_{p-1}, \ldots, m_1)$
$t \leftarrow s - \binom{p-1+h}{p-1}$
while $t > 0$ **then**
  $h \leftarrow h - 1$

```
    s ← t
    t ← s − C(p−1+h, p−1)
EncodingWithTheWeight(s, p, h)
```

When decoding $M = (m_p, ..., m_1)$, $\sum_{i=1}^{p} m_i$ is calculated. It asks for total number of encoded messages which the sum of weights of M is from $1 + \sum_{i=1}^{p} m_i$ to h.

algorithm 7 ImprovedDecoding(M, p, h)

$\quad$Input: $M = (m_p, m_{p-1}, \dots, m_1), p, h$; Output: s

$\quad h' \leftarrow \sum_{i=1}^{p} m_i$

$\quad$**if** $h' = h$ **then**

$\qquad s \leftarrow$ DecodingWithTheWeight(M, p, h')

$\quad$**else then**

$\qquad s \leftarrow \sum_{i=h'+1}^{h} \binom{p-1+i}{p-1}$+DecodingWithTheWeight($M, p, h'$)

$\quad$**Return** s

example 2 *In the case of* $p = 3, h = 3, k = 1$, *we have*

$1 = (0,0,3)$ $6 = (1,1,1)$ $11 = (0,0,2)$ $16 = (2,0,0)$
$2 = (0,1,2)$ $7 = (1,2,0)$ $12 = (0,1,1)$ $17 = (0,0,1)$
$3 = (0,2,1)$ $8 = (2,0,1)$ $13 = (0,2,0)$ $18 = (0,1,0)$
$4 = (0,3,0)$ $9 = (2,1,0)$ $14 = (1,0,1)$ $19 = (1,0,0)$
$5 = (1,0,2)$ $10 = (3,0,0)$ $15 = (1,1,0)$.

4.3 Application To Chor–Rivest Cryptosystem

We introduce that the Chor–Rivest Cryptosystem using our proposed coding. At this time, the changed part is as follows.

- At the time of encryption, we use the message M which is transformed by EncodingWithTheWeight Algorithm 4 or ImprovedEncoding Algorithm 6 instead of a binary message M.
- When decrypting, multiple roots may exist.

1. System Generation
   (a) Let p be a prime power, $h \leq p$ an integer such that discrete logarithms in $GF(p^h)$ can be efficiently computed.
   (b) Pick a random $t \in GF(p^h)$ that is algebraic of degree h over $GF(p)$.
   (c) Pick $g \in GF(p^h)$, g a multiplicative generator of $GF(p^h)$, *at random.*
   (d) Construction following Bose-Chowla theorem: Compute $a_i = \log_g^{t+\alpha_i}$ for all $\alpha_i \in GF(p)$.
   (e) Scramble the a_i's: Let π:$\{0, 1, ..., p-1\} \rightarrow \{0, 1, ..., p-1\}$ be a randomly chosen permutation. Set $b_i = a_{\pi(i)}$
   (f) Add some noise: Pick $0 \leq d \leq p^h - 2$ at random. Set $c_i = b_i + d$.
   (g) Public key–to be published: $c_0, c_1, ..., c_{p-1}$; p, h.
   (h) Private key–to be kept secret: t, g, π^{-1}, d.
2. Encryption
   - **Encoding a plain text using EncodingWithTheWeight Algorithm.** We encrypt a **message** $M = (x_0, x_1, ..., x_{p-1})$ of length p and weight $h (= \sum_{i=0}^{p-1} x_i)$, and send $E(M) = \sum_{i=0}^{p-1} x_i c_i \pmod{p^h - 1}$.
   - **Using the ImprovedEncoding Algorithm**. We encrypt a **meaasage** $M = (x_0, x_1, ..., x_{p-1})$ of length p and weight below h, and send $E(M) = \sum_{i=0}^{p-1} x_i c_i \pmod{p^h - 1}$.
3. Decryption
   (a) Let $r(t) = t^h \mod f(t)$, a polynomial of degree $\leq h - 1$ (computed once at system generation).
   (b) Given $s = E(M)$, computes $s' = s - hd \pmod{p^h - 1}$.
   (c) Compute $q(t) = g^{s'} \mod f(t)$, a polynomial of degree $h - 1$ in the formal variable t.
   (d) Add $t^h - r(t)$ to $q(t)$ to get $s(t) = t^h + q(t) - r(t)$, a polynomial of degree h in $GF(p)[t]$.
   (e)
   - **Encoding a plain text using EncodingWithTheWeight.** We now have $s(t) = (t+\alpha_{i_1})(t+\alpha_{i_2}) \cdots (t+\alpha_{i_h})$ namely $s(t)$ factors to *liner terms* over $GF(p)$. (There is the possibility of $\alpha_{i_j} = \alpha_{i_k} (1 \leq j \neq k \leq h)$).By successive substitutions, we find the h roots α_{i_j}'s (at most p substitutions needed). Apply π^{-1} to recover the coordinates of the original M having the bit 1.
   - **Using the ImprovedEncoding Algorithm.** We now have $s(t) = (t+\alpha_{i_1})(t+\alpha_{i_2}) \cdots (t+\alpha_{i'_h})$. h' is below h. (There is the possibility of $a_{i_j} = a_{i_k} (1 \leq j \neq k \leq h')$). By successive substitutions, we find the h' roots α_{i_j}'s (at most p substitutions needed). Apply π^{-1} to recover the coordinates of the original M having the bit 1.

4.4 Information Rate

Using the coding scheme for transforming into 0-1 vectors, the message space is $|M_b| = \binom{p}{h}$ and the information rate R_b is

$$R_b = \frac{\log \binom{p}{h}}{\log p^h}.$$

When we use the coding scheme for mapping a numbers into non-negative vectors with weight h, the message space $|M_h|$ is $\binom{p+h-1}{h}$ and the information rate R_h is

$$R_h = \frac{\log \binom{p+h-1}{h}}{\log p^h}.$$

Moreover using the coding scheme for transforming a text into the vectors with weights of k to h, the message space is $|M_{k-h}| = \sum_{i=k}^{h} \binom{p+i-1}{i}$ and the information rate R_{k-h} is

$$R_{k-h} = \frac{\log \sum_{i=k}^{h} \binom{p+i-1}{i}}{\log p^h}.$$

Each information rate is not asymptotically different. In Chor–Rivest[4], the recommendation parameter p is 256 and h is 25. k is taken as 1 to $h-1$. Each message space and information rate is compared as follow.

The comparison table of message space and information rate in $p = 256, h = 25$		
$p = 256, h = 25$	message space	information rate
$0-1$vectors	M_b	0.572856
weight$= 25$	$10.457709 \times M_b$	0.573594
weight$\in \{24, 25\}$	$11.391433 \times M_b$	0.573666
weight$\in \{23, 25\}$	$11.475100 \times M_b$	0.573673
weight$\in \{22, 25\}$	$11.482624 \times M_b$	0.573674
weight$\in \{21, 25\}$	$11.483303 \times M_b$	0.573674
weight$\in \{20, 25\}$	$11.483364 \times M_b$	0.573674
weight$\in \{19, 25\}$	$11.483370 \times M_b$	0.573674
weight$\in \{1, 25\}$	$11.483371 \times M_b$	0.573674

4.5 Security Consideration

Although some knapsack scheme using 0-1 vectors become insecure against our proposed scheme, some knapsack schemes using the vectors whose weight is from k to h may be secure. For the encoding text $e = (e_1, ..., e_n)$ of $\sum_{i=1}^{n} e_i = h$, the value of $\|e\|$ changes from $\sqrt{h}$ to h.

lemma 1 *As for non-negative integer vector $e = (e_1, ..., e_n)$ with $\sum_{i=1}^{n} e_i = h$ and $e_i \in \{0, h\}$,*

$$\sqrt{h} \leq \|e\| \leq h.$$

Proof. We assume that $(e_1, ..., e_n)$ such that $\|e\| = t$ and $\sum_{i=1}^{n} e_i = h$. The following substitution is carried out to e_i, e_j $(i \neq j)$ for $e_i \geq e_j > 0$,

$$e_i \leftarrow e_i + 1, \ e_j \leftarrow e_j - 1.$$

Then $\|e\|$ is as follow,

$$\begin{aligned} \|e\| &= (t - e_i^2 - e_j^2) + (e_i + 1)^2 + (e_j - 1)^2 \\ &= t + 2 + 2(e_i - e_j) > t \end{aligned}$$

So, when $\forall i (1 \leq i \leq n)$, $e_i = h$, $e_j = 0$ $(1 \leq j \leq n, i \neq j)$, the maximum of $\|e\|$ is

$$\|e\| = \sqrt{h^2} = h.$$

When becoming the minimum of $\|e\|$, it is necessary is just to consider $0 < e_i < e_j$ $(i \neq j)$. So the minimum of $\|e\|$ is

$$\|e\| = \sqrt{1^2 \times h} = \sqrt{h}.$$

$\diamond$

When the subset sum problem change to the shortest vector problem, the solution vector in the lattice will become big. If we want to solve it using our proposed scheme, the solution vector would not be found. Some multiplicative knapsack schemes based on discrete logarithm using these coding algorithms 4, 5 is secure against our proposed scheme. Other considering several possible attacks have written to the paper of [4] and [12].

References

[1] E F. Brickell. Solving low density knapsacks. In *Advances in Cryptology: Proceedings of Crypto '83*, pages 25–37. Plenum Press, 1983.

[2] E F. Brickell. Breaking iterated knapsacks. In *Advances in Cyrptology: Proceedings of Crypto '84*, pages 342–358. Springer, 1985.

[3] P. Camion and H. Chabanne. On the powerline system. In *Applicable Algebra in Engineering Communication and Computing*, volume 9, pages 405–432, 1999.

[4] B. Chor and R L. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. In *Advances in Cyrptology: Proceedings of Crypto '84*, volume 196, pages 54–65. Springer, 1984.

[5] M J. Coster, B A. LaMacchia, A M. Odlyzko, and C P. Schnorr. An improved low-density subset sum algorithm. In *Advances in Cryptology: Proceedings of Eurocrypt '91*, volume 547, pages 54–67, 1991.

[6] T M. Cover. Enumerative source encoding. In *IEEE Trans. Inform. Theory*, volume IT-19, pages 73–77, 1973.

[7] B E. Dom. A systematic enumeration of contingency tables. Technical report, IBM, 2000.

[8] J C. Lagarias and A M. Odlyzko. Solving low-density subset sum problems. *jacm*, 32:229–246, 1985.

[9] Ming Kin Lai. Knapsack Cryptosystems: The Past and the Future, 2001.

[10] A K. Lenstra, H W. Lenstra Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[11] H W. Lenstra Jr. On the Chor–Rivest knapsack cryptosystem. *Journal of Cryptology*, 3:149–155, 1991.

[12] T Okamoto. Quantum public-key cryptosystems with higher information rates, 2000.

[13] T. Okamoto, K. Tanaka, and S. Uchiyama. Quantum public-key cryptosystems. In *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*. Springer, 2000.