Information Sciences 179 (2009) 3116–3121

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Cryptanalysis of a knapsack-based probabilistic encryption scheme

Amr M. Youssef *

*Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, Canada H3G 1M8*

ARTICLE INFO

*Article history:*
Received 5 March 2008
Received in revised form 23 April 2009
Accepted 27 May 2009

*Keywords:*
Public key cryptography
Cryptanalysis
Knapsack cryptosystem
Lattice basis reduction
Diophantine equations

ABSTRACT

Wang et al. [B. Wang, Q. Wu, Y. Hu, A knapsack-based probabilistic encryption scheme, Information Sciences 177(19) (2007) 3981–3994] proposed a high density knapsack-based probabilistic encryption scheme with non-binary coefficients. In this paper, we present a heuristic attack that can be used to recover the private key parameters from the known public key parameters. In particular, we show that the restrictions imposed on the system parameters allow the attacker to recover a short list of candidates for the first half of the public key. The second half of the public key can then be recovered using an attack based on lattice basis reduction. Finally, by encrypting an arbitrary plaintext using the known public key then decrypting the resulting ciphertext using these estimated candidate solutions, the right private key can be uniquely determined.



## 1. Introduction

Knapsack-based cryptosystems [6,7] were among the first public key systems to be invented. These seemingly NP-hard systems attracted considerable interest because of their high encryption/decryption rates. However, it was soon realized that the underlying knapsacks often have a low density and hence they are vulnerable to lattice reduction attacks [8,10].

Let $S = \{a_1, a_2, \ldots, a_n\}$ be a knapsack set. The density of $S$ is defined as [7]

$$d = \frac{n}{\max\{log_2(a_i) | 1 \leqslant i \leqslant n\}},$$

where $log_2(a_i)$ denotes the base 2 logarithm of $a_i$. It was noted in [2] that the density of the knapsack vector must be larger than a critical value (0.9408) in order to avoid low density attacks. The reader is referred to [4,9] for some recent extended definitions of knapsack density and its relation to cryptanalysis of knapsack-based cryptosystems.

Several designers tend to increase the density beyond the critical density by allowing non-binary coefficients. Recently, Wang et al. [12] proposed a knapsack-based probabilistic encryption scheme with non-binary coefficients. For properly chosen parameters, the underlying knapsack problem enjoys a high density larger than 1.06 in the worst case, which makes it secure against the low density subset-sum attacks.

In this paper, we show that the security level of this cryptosystem was overestimated by its designers. In particular, we present a heuristic attack that can be used to recover the private key parameters from the known public key parameters.

Let $(m, w)$ denote the private key of the proposed system with a parameter $n$ where $n$ denotes the length of the plaintext block $P = (p_1, p_2, \ldots, p_n), 0 \leqslant p_i \leqslant 3$. In [12], it was shown that exhaustive search for the plaintext corresponding to a given ciphertext requires $n2^{3n/2}$ encryption steps. In here, we show that, because of the restrictions imposed on the system parameters, a short list $L_m$, $m \in L_m$ can usually be derived in $O(n^3)$ steps. For each $m' \in L_m$, we show that obtaining a candidate

* Tel.: +1 514 848 2424.
*E-mail address:* youssef@ciise.concordia.ca

0020-0255/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2009.05.015

solution for the other half of the public key, $w'$, is equivalent to solving a system of linear Diophantine equations with lower and upper bounds on the variables. Although this is an NP-complete problem, Aardal et al. [1] have developed a relatively efficient heuristic algorithm for solving this problem based on lattice basis reduction [5]. Using this heuristic algorithm allows us to recover $w'$ corresponding to a given $m'$ in a relatively very efficient way.

Finally, by encrypting an arbitrary plaintext using the known public key then decrypting the resulting ciphertext using these $(m\prime, w\prime)$ candidate solutions and checking whether the decrypted plaintext matches the original plaintext, the right private key, $(m, w)$, can be uniquely identified.

The rest of the paper is organized as follows. In the next section, the features of the public key system that are relevant to our attack are reviewed. In Section 3, we briefly describe the Aardal's algorithm. The proposed attack is described in Section 4. Finally, Section 5 is the conclusion.

## 2. Description of the knapsack-based scheme

In this section we present the key generation steps of the proposed public key scheme because of its relevance to our attack. Further details about the encryption and decryption operations as well as justification for the choice of the different parameters can be found in [12].

The key generation steps can be summarized as follows:

1. Randomly choose $n-1$ numbers $C = (c_1, \ldots, c_{n-1})$, where $c_i \in K = \{10, 11, 14, 15, 16, 17, 20, 21, 22\}$.
2. Randomly choose $n-1$ numbers $E = (e_2, \ldots, e_n)$ such that $gcd(c_i, e_j) = 1$, $2 \leqslant j \leqslant n$, $1 \leqslant i \leqslant n-1$.
3. Compute $a_1 = \prod_{j=1}^{n-1} c_j$.
4. For $i = 2, \ldots, n-1$, compute $a_i = e_i \prod_{j=i}^{n-1} c_j$. Set $a_n = e_n$.
5. Randomly select two large numbers $m > 27\sum_{i=1}^{n} a_i$, and $w$ such that $gcd(w, m) = 1$.
6. Compute the vector $B = (b_1, \ldots, b_n)$ where $b_i = wa_i \bmod m$.

From [12], the components of $E = (e_2, \ldots, e_n)$ are chosen such that $a_i$ have the same length for $1 \leqslant i \leqslant n$.

The public key of the above system is the permuted set corresponding to $B$ and the private key is $(m, w)$.

## 3. Solving linear diophantine equations with bounds on the variables

A Diophantine equation is an indeterminate polynomial equation that allows the variables to be integers only. In [1], Adral et al. developed an efficient heuristic algorithm for solving a system of linear Diophantine equations with lower and upper bounds on the variables. Let

$$\mathbf{x}^T = (x_1, x_2, \ldots, x_n), \quad \mathbf{u}^T = (u_1, u_2, \ldots, u_n).$$

We use the notation $\mathbf{x} \leqslant \mathbf{u}$ to denote the fact that $x_i \leqslant u_i$ for all $0 \leqslant i \leqslant n$.

Let $\mathbf{A}$ be an $M \times N$ matrix with integer coefficients. Then Adral's algorithm solves for $\mathbf{x} \in Z^N$ such that $\mathbf{Ax} = \mathbf{d}, \mathbf{0} \leqslant \mathbf{x} \leqslant \mathbf{u}$.

The algorithm is based on lattice basis reduction [8,5]. It starts by first finding a short vector satisfying the system of Diophantine equations, and a set of vectors belonging to the null space of the constraint matrix. All these vectors are relatively short because of the lattice reduction. Then using a branch and bound [13] on linear combinations of the null space vectors, the algorithm yields a vector that satisfies the bound constraints or provides no proof that such algorithm exists.

In what follows we briefly describe the steps involved. For further details about the lattice reduction and a proof that the algorithm will find the solution if it exists, the reader is referred to [5,1], respectively.

### 3.1. Algorithm 1

We formulate an initial basis $\mathbf{B}$ as follows:

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}^{(N)} & \mathbf{0}^{N\times 1} \\ \mathbf{0}^{(1\times N)} & N_1 \\ N_2\mathbf{A} & -N_2\mathbf{d} \end{pmatrix}$$

Let $\widehat{\mathbf{B}}$ denote the reduced lattice corresponding to $\mathbf{B}$. The integers $N_1$ and $N_2$ are chosen in order to guarantee that the reduced lattice $\widehat{\mathbf{B}}$ will contain a vector in the form $\left(\mathbf{x}_d^T, N_1, \mathbf{0}^{(1\times M)}\right)^T$ and $N-M$ vectors of the form $\left(\mathbf{x}_0^T, 0, \mathbf{0}^{M\times 1}\right)^T$ (cf. Theorem 4 in [1]), where the vector $\mathbf{x}_d$ satisfies $\mathbf{Ax}_d = \mathbf{d}, \mathbf{0} \leqslant \mathbf{x} \leqslant \mathbf{u}$. Since the basis reduction finds a short vector belonging to a given lattice, we hope that $\mathbf{x}_d$ is short in the sense that it satisfies the bounds. If not, we can add integer linear combinations of the vectors satisfying $\mathbf{Ax} = \mathbf{0}$ to $\mathbf{x}_d$ in order to obtain a vector that satisfies the bound as well.

To illustrate the above steps, consider the following numerical example.

**Example 1.** Suppose we want to find a solution $\mathbf{x} = (x_1 \ldots x_5)^T$ that satisfies the following system of linear Diophantine equations

$$20037x_1 - 24111x_2 - 20x_3 = 0$$
$$8242x_1 - 24111x_4 - 20x_5 = 0$$

with the additional constraints that $(1,0,0,0,0)^T \leqslant \mathbf{x}^T < (24111, 24111, 1205, 24111, 1205)$.

We form the following matrix

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & N1 \\ 20037N_2 & -24111N_2 & -20N_2 & 0 & 0 & 0 \\ 8242N_2 & 0 & 0 & -24111N_2 & -20N_2 & 0 \end{pmatrix}$$

where $N1 = 1000$ and $N2 = 10000$ are two arbitrary constants satisfying the conditions given by Theorem 4 in [1].

After we run the lattice reduction (column wise) on the matrix above, we get

$$\widehat{\mathbf{B}} = \begin{pmatrix} 603 & -133 & 253 & 0 & -193 & -295 \\ 501 & -111 & 211 & 0 & -160 & -245 \\ 135 & 570 & -903 & 0 & -469 & -186 \\ 206 & -46 & 86 & 0 & -66 & -101 \\ 153 & 646 & 584 & 0 & 31 & 191 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & -10000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10000 \end{pmatrix}$$

It is easy to verify that the fourth column in the matrix above is in the form $\left(\mathbf{x}_d^T, N_1, \mathbf{0}^{(1\times M)}\right)^T$ where $\mathbf{x}_d = (0,0,0,0,0)^T$ satisfies the set of equations but it does not satisfy the lower bound. Also, any linear integer combination of the first $N - M(= 5 - 2 = 3)$ columns satisfy the equation $\mathbf{Ax} = \mathbf{0}$.

Then we use the heuristic algorithm in Section 5 of [1] to find a solution the satisfies the set of constraints. If this heuristic algorithm fails, we branch on a linear combination of these 3 columns. For our example above, it is easy to check that the first column of the matrix satisfies the set of constraints. Hence we obtain $\mathbf{x} = (603, 501, 135, 206, 153)^T$ as a valid solution for our problem.

## 4. The proposed attack

### 4.1. Recovering m

In this section, we show that given the known permuted list $(b_{i_1}, b_{i_2}, \ldots, b_{i_n})$, the attacker can derive a relatively short list $L_m = \{m_1, \ldots, m_L\}$ where $m \in L_m$.

The following Lemma [3] will be used in the attack.

**Lemma 1.** *If $x_1 \equiv x_2 \bmod m$, then $m|x_1 - x_2$.*

Before we proceed further, it should be noted that if $c_i$ is uniformly distributed over $K$ then $c_i \approx \sqrt[9]{10 \cdot 11 \cdot 14 \cdots 22} \approx 15.7$. It should also be noted that the components of $E = (e_2, \ldots, e_n)$ generated by the key generation above are chosen such that $a_i$ have the same length for $1 \leqslant i \leqslant n$ [12]. Thus we have

$$a_1 \approx a_2 \Rightarrow c_1 \prod_{j=2}^{n-1} c_j \approx e_2 \prod_{j=2}^{n-1} \Rightarrow e_2 \approx c_1 \Rightarrow e_2 \approx 15.7,$$
$$a_2 \approx a_3 \Rightarrow e_2 \cdot c_2 \prod_{j=3}^{n-1} c_j \approx e_3 \prod_{j=3}^{n-1} \Rightarrow e_3 \approx e_2 c_2 \approx 15.7^2.$$

Similarly, one can show that $e_i \approx 15.7^{i-1}$.

By noting that

$$\frac{b_2}{b_1} \equiv \frac{a_2}{a_1} \equiv \frac{e_2}{c_1} \bmod m,$$

thus we have $m|t_{12} = b_2 c_1 - b_1 e_2$.

Similarly, from the relation between $b_1, b_3$ and $b_2, b_3$ we have

$$\frac{b_3}{b_1} \equiv \frac{a_3}{a_1} \equiv \frac{e_3}{c_1 c_2} \mod m \Rightarrow m | t_{13} = b_1 e_3 - b_3 c_1 c_2$$

and

$$\frac{b_3}{b_2} \equiv \frac{a_3}{a_2} \equiv \frac{e_3}{e_2 c_2} \mod m \Rightarrow m | t_{23} = b_2 e_3 - b_3 e_2 c_2.$$

Using the above relations, and from Lemma 1, one can guess $m$ as $gcd(t_{12}, t_{13}, t_{23})$. A large number of wrong candidates can be eliminated by noting that $m > max_{i=1}^{n} b_i$.

If we use $l$ $b_i$'s in this step, and taking into account the fact that we only know a permuted version of $B$, then the number of steps is approximately given by

$$T_1 \approx \binom{n}{l} \times l! \times 9^{l-1} \times 15.7 \times \cdots \times 15.7^{l-1} = \binom{n}{l} \times l! \times 9^{l-1} \times 15.7^{\frac{(l-1)l}{2}}. \quad (1)$$

The term $9^{l-1}$ in the equation above accounts for the fact that we have 9 different choices for each $c_i \in K$.

Because of the restrictions we impose on the size of $e_i$'s, and since $gcd(c_i, e_j) = 1$, it is easy to show that wrong choices of $b_1, \ldots b_l$ are very unlikely to add a large number of entries to the list $L_m$. In other words, most of the candidates for $m$ will correspond to the right choice of $b_1, \ldots, b_l$. Our experimental results confirmed the above observation. It should be noted, however, that the success of our attack does not depend on this claim. Let $\#\{\cdot\}$ denotes the cardinality of the enclosed set. Our experimental results show that $\#\{L_m\}$ is very moderate for $l = 3$ and it does not tend to increase as $n$ increases. In fact, the filtering process above becomes stricter for larger values of $n$ and hence assuming that $\#\{L_m\} \approx O(n)$ is a very conservative estimate. For $l = 3$, it is easy to show that $T_1 \approx O(n^3)$. The gcd calculations can be performed efficiently using the Euclidian algorithm [7] in $O(log^2(m))$ bit operations. Also, the gcd calculations should not be performed through out all these step. For example, iff the choice of $b_1, b_2$ produce a valid $m$, then we need to consider the rest of the relations. We should also note that the search should only consider $e_i$'s and $c_j$'s for with $gcd(e_i, c_j) = 1$.

**Example 2.** Consider the example given in [12] with $B$ = (1611671814, 2484057721, 762578407, 1615332064, 2498998196, 1623000439, 1608556339, 1640312439). The filtering process above (with $l = 3$ where we searched the range $1 \leqslant e_i \leqslant 22^{i-1},\ i = 2, 3$) produces

$L_m$ = {5807226349, **2583882721**, 7485448163, 19066625861, 17454954047, 14231610419, 11008266791, 9396594977, 6173251349, 2949907721, 17588054047, 15976382233, 14364710419, 11141366791, 9529694977, 7918023163, 10841891791, 6006876349}. Note that $m = 2583882721 \in L_m$.

### 4.2. Known m attack: recovering w

Before we present our $w$-recovery attack, we should note that the authors in [12] have also considered the security of their system under the assumption that $m$ becomes known to the attacker. Let $x = w^{-1} \mod m$. By noting that $a_i = x b_i \mod m$, then there exists a set of integers $k_i, i = 1, \ldots, n$ such that $x b_i - k_i m = a_i$. Thus we have $\frac{b_i}{m} - \frac{k_i}{x} = \frac{a_i}{mx} \approx \frac{a_i}{27 n a_i x} \approx \frac{1}{27nx}$ which is very small. Hence, the set $\left\{\frac{k_i}{x}\right\}$ with a denominator $x$ approximates the fractions $\left\{\frac{b_i}{m}\right\}$. The authors in [12] argued that if one can find an efficient algorithm to solve the above simultaneous Diophantine approximation problem, then one can recover the secret parameter $w$. However, the authors in [12] stopped short of providing such an algorithm. Our experimental results (using Algorithm 3.108 in [7]) proved not to be successful. This can be explained by noting that traditional simultaneous diophantine approximation algorithms are concerned with approximating a vector $\left(\frac{q_1}{q}, \frac{q_2}{q}, \ldots \frac{q_l}{q}\right)$ of rational numbers by a vector $\left(\frac{p_1}{p}, \frac{p_2}{p}, \ldots \frac{p_l}{p}\right)$ of rational numbers with a small denominator $p$ which is not the case for the above system where the denominator is very large ($p = x = w^{-1} \mod m = O(m)$).

In what follows, we present a heuristic algorithm that can be used to recover the secret parameter $w$ from the public key parameters and $m$. From the key generation algorithm, we have

$$a_i \mod c_{n-1} = 0 \Rightarrow (b_i w^{-1} \mod m) \mod c_{n-1} = 0, \quad i = 1, \ldots n-1.$$

Then the public key parameters $b_i$, $i - 1, \ldots n-1$ should satisfy following system of linear Diophantine equations

$$\begin{aligned} &b_i \times x_1 - m \times x_{2i} - c_{n-1} \times x_{2i+1} = 0, \\ &0 \leqslant x_{2i} < m, 0 \leqslant x_{2i+1} < \lfloor m / c_{n-1} \rfloor, \quad i = 1, \ldots n-1. \end{aligned} \quad (2)$$

The above system of equations can be solved using Algorithm 1 above (with $M = n - 1$ and $N = 2M + 1 = 2n - 1$).

Let $\mathbf{x} = (x_1, \ldots x_{2n-1})^T$ denote a vector that satisfies Eq. (2) above. Then, as indicated by our experiments, another vector

$$\mathbf{x}' = \pm(x'_1, q x_2, q x_3, \ldots, q x_{2n-1})^T,$$

where $q$ is a relatively small integer may also satisfy this equation. If this vector happens to be shorter than the original vector $\mathbf{x}$ (i.e., $x'_1 \ll x_1$ and hence $\|\mathbf{x}\prime\| < \|\mathbf{x}\|$), then it will be returned by the lattice reduction step instead of the vector $\mathbf{x}$. In this

case, the solution recovered by our algorithm will be in the form $a'_i = q \times m \times x_{2i} + q \times c_{n-1} x_{2i+1} \bmod m = a_i \times q$, $i = 1, \ldots n-1$. To overcome this problem, we utilize the fact that $gcd(a_1, \ldots, a_{n-1}) = c_n$. Hence we can evaluate $q = gcd(a'_1, \ldots, a'_{n-1})/c_n$. Then we can estimate $a_i = \frac{a'_i}{q} = \frac{a'_i \times c_{n-1}}{gcd(a'_1, \cdots a'_{n-1})}$. Finally, $w$ can be recovered by evaluating $w = b_i a_i^{-1} \bmod m$.

The following examples illustrates the above steps.

**Example 3.** Consider a toy example with $n = 3$, $c_1 = 15$, $c_2 = 20$, $E = (17, 253)$, $A = (300, 340, 253)$, $m = 24111$, $w = 2719$, $B = (20037, 8242, 12799)$.

If $m$ is known, then the cryptanalyst knows that $x_1 = w^{-1} \bmod m$ must satisfy the following system of linear Diophantine equations

$$20037x_1 - 24111x_2 - 20x_3 = 0$$
$$8242x_1 - 24111x_4 - 20x_5 = 0$$

where $(1, 0, 0, 0, 0)^T \leqslant \mathbf{x}^T < (24111, 24111, 1205, 24111, 1205)$.

Using the results of Example 1, we have $x_1 = 603 \Rightarrow a'_1 = b_1 \times x_1 \bmod m = 2700, a'_2 = b_2 \times x_1 \bmod m = 3060$. Then, for $i = 1, 2$ we have $a_i = \frac{a'_i \times 20}{gcd(2700, 3060)} = \frac{a'_i \times 20}{180} \Rightarrow a_1 = 300, a_2 = 340$. Hence $w \equiv \frac{b_i}{a_i} \bmod 24111 = 2719$ and $a_3 = b_3 \times x_1 \bmod 24111 = 253$ which completes our example.

Once the set A is recovered, the effect of the re-indexing can be eliminated by utilizing the fact $a_i$ should satisfy $a_i = e_i \prod_{j=i}^{n-1} c_j$, $gcd(e_j, c_k) = 1$.

Since we don't know which value in $K$ corresponds to $c_{n-1}$ and which value in $B$ corresponds to $b_{n-1}$, then the algorithm above has to be repeated for the 9 possible choices of $c_{n-1} \in K$ and $\binom{n}{1} = n - 1$ choices for $b_n$, i.e., for $9n$ times.

Assuming the number of steps required to run one instant Algorithm 1 is $T_2$, then the total number of steps required to break this system is approximately given by

$T_1 + \#\{L_m\} \times (9n \times T_2)$ where $T_1$ is given by Eq. (1).

### 4.3. Computational experiments

For relatively small values of $n$, the branch and bound step at the end of Algorithm 1 runs very efficiently. However, for large values of $n \gtrapprox 30$, because the integer coefficients of the corresponding integer optimization problem become very large, the truncation errors associated with our branch and bound implementation prevents us from implementing this step successfully.

On the other hand, we observed that, with relatively good probability, the first vector in the reduced lattice which corresponds to the shortest non-zero choice of $\mathbf{x}$ tends to satisfy the required bounds and hence provides a valid solution for our systems of Diophantine equations. So, we decided to run Algorithm 1 without the branch and bound step and declare a failure when none of the $(N - M)$ short nonzero vectors in the form $(\mathbf{x}^T, 0, \mathbf{0}^{(1\times M)})^T$ in the reduced lattice satisfies the required constraints.

Since we have decided not to run the branch and bound step, then we ignore its associated complexity in calculating the number of steps required by Algorithm 1. In other words, the number of arithmetic operations needed by Algorithm 1 would now be equivalent to that required by the lattice reduction steps. Let $\mathbf{c}_i$ denote the $i$th column of the matrix $\mathbf{B}$ that is used as an input to Algorithm 1. Let $C$ be an arbitrary constant satisfying $\|\mathbf{c}_i\|^2 \leqslant C$ for $i = 1, \ldots N + 1$. Then $T_2 = O(N^4 logC)$ on integers of size $O(NlogC)$ [7]. From [1], $N_1$ and $N_2$ used in the formation of the matrix $\mathbf{B}$ are chosen such that $N_1 > N_{01}$ and $N_2 > 2^{N+M} N_1^2 + N_{02}$ where the sizes of $N_{01}$ and $N_{02}$ are polynomially bounded by the size of $\mathbf{A}$ and $\mathbf{d}$ (see Section 2.2 in [11] for the definition of the size of a matrix). By simple mathematical manipulation, we obtain $T_2 = O(n^5)$. Thus, the overall complexity of the attack grows as $O(n^7)$.

Since the complexity of our attack grows as $O(n^7)$, one may argue that we can choose $n$ large enough to render our attack impractical. However, we should note that the size of the public key also grows as $O(n^2 log(n))$ which would also limit the practical applications of this system with very large $n$.

For $n = 20, 30, 40, 50$ and 60, the estimated security level [12] to recover only a single plaintext from its known ciphertext $\approx 2^{34}, 2^{50}, 2^{65}, 2^{80}$ and $2^{96}$, respectively. Algorithm 1, implemented without the branch and bound step, successfully recovers the secret key in $\approx 70\%$, 70%, 60%, 60% and 60% of the times for $n = 20, 30, 40, 50$ and 60, respectively. Providing a theoretical estimate for the success probability of the attack, without the branch and bound step, seems to be an intractable problem because of the heuristic nature of the lattice reduction algorithm. On the other hand, the above experimental results clearly shows that the security level of the considered cryptosystem has been overestimated by its designers.

We emphasis that the main reason for not being able to recover the right key in all the failed instances above is due to the elimination of the branch and bound step in Algorithm 1, which as we have already stated, means that we declare a failure when the first vector in the reduced lattice does not satisfy the required constraints. A more careful and dedicated implementation which includes the branch and bound step can certainly increase the success probability of the attack. However, we believe that overcoming the truncation errors associated with very large integer coefficients ($\approx m \approx 27n22^{n-1}$) may not be an easy task.

## 5. Conclusions

The security level of the probabilistic knapscak-based encryption scheme proposed by Wang et al. is overestimated. Our heuristic attack also provides a clarifying example that shows that high density do not always prevent efficient reductions to lattice problems.

## Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant N00930. The author would like to thank the anonymous reviewers for their comments that helped improve the presentation of the paper.

## References

[1] K. Aardal, C. Hurkens, A. Lenstra, Solving a system of linear diophantine equations with lower and upper bounds on the variables, Mathematics of Operations Research 25 (3) (2000) 427–442.
[2] M.J. Coster, B.A. LaMacchia, A.M. Odlyzko, C.P. Schnorr, An improved low-density subset-sum algorithm, Advances in Cryptology – EUROCRYPT 91, LNCS 547, Springer-Verlag, 1991, pp. 54–67.
[3] G.H. Hardy, E.M. Wright, An Introduction to the Theory of Numbers, fifth ed., Oxford University Press, 1979.
[4] N. Kunihiro, New Definition of Density on Knapsack Cryptosystems, Progress in Cryptology AFRICACRYPT 2008, LNCS 5023, Springer-Verlag, 2008, pp. 156–173.
[5] A.K. Lenstra, H.W. Lenstra, L. Lovász, Factoring Polynomials with Rational Coefficients, Mathematische Annalen 261 (1982) 515–534.
[6] R.C. Merkle, M.E. Hellman, Hiding information and signatures in trapdoor knapsacks, IEEE Transactions on Information Theory 24 (1978) 525–530.
[7] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptographic Research, CRC Press, 1996.
[8] P. Nguyen, J. Stern, Lattice reduction in cryptology: an update, algorithmic number theory, in: Proceedings of ANTS-IV, Springer-Verlag, LNCS 1838, 2000, pp. 85–112.
[9] P. Nguyen, J. Stern, Adapting Density Attacks to Low-Weight Knapsacks, Advances in Cryptology – ASIACRYPT 2005, LNCS 3788, Springer-Verlag, 2005, pp. 41–58.
[10] A.M. Odlyzko, The rise and fall of knapsack cryptosystems, in: Cryptology and Computational Number Theory, Proceedings of Symposia in Applied Mathematics, vol. 42, A.M.S., 1990, pp. 75–88.
[11] A. Schrijver, Theory of Linear and Integer Programming, first ed., Wiley, 1998.
[12] B. Wang, Q. Wu, Y. Hu, A knapsack-based probabilistic encryption scheme, Information Sciences 177 (19) (2007) 3981–3994.
[13] L.A. Wolsey, Integer Programming, first ed., Wiley-Interscience, 1998.