Cryptanalysis of a quadratic compact knapsack public-key cryptosystem

Moon Sung Lee *

National Institute for Mathematical Sciences, KT Daeduk 2 Research Center, 463-1, Jeonmin-dong, Yusung-gu, Daejeon, 305-811, Republic of Korea

A R T I C L E   I N F O

Article history:
Received 10 June 2010
Received in revised form 8 September 2011
Accepted 8 September 2011

Keywords:
Public key cryptanalysis
Knapsack cryptosystem
Quadratic compact knapsack
Lattice attack
Modular equation
Linear equation

A B S T R A C T

Recently, Wang and Hu have proposed a high-density quadratic compact knapsack public-key cryptosystem using the Chinese remainder theorem to disguise two secret cargo vectors. The system is claimed to be secure against certain known attacks; however, it has not been demonstrated to fulfill any provable security goals. In this work, we show that this system is not secure. Exploiting the special structure of system parameters, we first show that a candidate list for the secret modulus can be obtained by solving linear equations with small solutions. Next, we show that with this candidate list, all other secrets can be recovered in succession with lattice-based methods by solving certain modular linear equations with small solutions. As a result, recovering a private key can be done in about 11 h for the proposed system parameter \( n = 100 \). We also discuss a method to thwart the proposed attack.

1. Introduction

Recently, a quadratic compact knapsack public-key cryptosystem has been proposed by Wang and Hu [1]. The scheme uses two secret cargo vectors \( A \) and \( B \) with a special structure such that some entries of \( A \) and \( B \) have only small factors. These two vectors are used to make a public cargo vector \( F \) with a matrix \( C = (c_{ij})_{2 \times 2} \) by using the Chinese remainder theorem and modular multiplication in order to scramble cargo vectors and conceal its special structure. Interestingly, this scheme does not use a binary message. Instead, it uses a message \( M = (m_1, \ldots, m_n) \) with \( m_i \in \{0, 1, 2, \ldots, 15\} \) that is encrypted into a ciphertext \( c \) such that \( c = \sum_{i=1}^{n} f_i m_i^2 \), where \( F = (f_1, \ldots, f_n) \) is a public cargo vector. Using a non-binary message, this scheme seems resistant to low-density attacks. Other known attacks are also considered, including the Diophantine approximation attack and the orthogonal lattice attack; however, it has not been demonstrated to fulfill any provable security goals.

In some sense, this scheme can be viewed as a CRT-variant of a previous knapsack-based probabilistic encryption scheme [2], where only one secret cargo vector is used and some entries have only small factors. Due to the special structure, it is cryptanalyzed by Youssef [3], where it is shown that a short candidate list for the secret modulus can be found in time complexity \( O(n^3) \), and the private key can be retrieved in time complexity \( O(n^7) \) using an attack based on lattice basis reductions. This attack is later improved by Lee [4] where it is shown that the suggested parameters are completely insecure.

In this paper, we show that this CRT-variant is also not secure. That is, scrambling cargo vectors using a matrix and the Chinese remainder theorem is not enough to conceal the special structure. More precisely, we present a heuristic key-recovery attack on the scheme presented in [1] which runs in polynomial time. Our observation is that there exist some relations among the entries of a public cargo vector \( F = (f_1, \ldots, f_n) \) due to the scheme’s special structure. Moreover, these relations are good enough to recover the private key within a reasonable time.

We first show that the candidate list for the secret modulus \( N \) can be obtained using relations among \( f_i \)'s by solving certain multivariate linear equations with small solutions. Next we show that candidate \( N \) values can be checked and other secrets...
can be recovered with lattice-based methods finding small solutions of certain modular multivariate linear equations. We also discuss how to thwart the proposed attack.

We note that Yousef already attacked this scheme in [5] where a heuristic stereotyped message attack is done to recover the plaintext message when partial information about the original message is known. The attack in [5] does not use any internal structure of the secret key. In contrast, we use the special property of secret cargo vectors to recover the secret key, and decryption can be done with the obtained secret key.

The rest of the paper is organized as follows. In the next section, we review some preliminaries for understanding the proposed attack. The key generation algorithm from the scheme presented in [1] is reviewed in Section 3. In Section 4, we present a main observation on which our attack is based, and the proposed attack is presented with experimental results in Section 5. Finally, Section 6 contains a brief conclusion.

2. Preliminaries

In this section, we review lattices and present some facts from number theory that are used in the rest of the paper.

2.1. Notation

Throughout this paper, the following notation is used. The greatest common divisor (GCD) of two integers \( a \) and \( b \) is denoted by \( \gcd(a, b) \). For a set or list \( L \), the cardinality or length of \( L \) is denoted by \( |L| \). We write the binary length of an integer \( a \) as \( |a|_2 \). Finally, the smallest integer greater than or equal to \( r \in \mathbb{R} \) is denoted by \( \lceil r \rceil \).

2.2. Lattices

An \( n \)-dimensional lattice is the set of integer combinations \( \{ \sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z} \} \) of \( n \) linearly independent vectors \( b_1, \ldots, b_n \in \mathbb{R}^n \). In this paper, we are only interested in the integer lattices such that \( b_1, \ldots, b_n \in \mathbb{Z}^n \). The set of vectors \( b_1, \ldots, b_n \) is called a basis for the lattice and can be represented by a matrix \( B \), where the basis vectors are the matrix’s rows. The lattice generated by \( B \) is denoted by \( L(B) \).

One of the most important algorithmic problems in lattices is the shortest vector problem (SVP), which is to find the shortest non-zero vector in a given lattice. Only exponential time algorithms [6-8] are known for this problem, and polynomial time algorithms such as LLL achieve only exponential approximation factors although they behave much better in practice. For a small dimension, such as when \( n < 50 \), SVP can be solved efficiently by algorithms presented in [7,8].

One of the applications of lattices is in finding small solutions of (modular) linear multivariate equations. For a general modular equation \( a_1 x_1 + \cdots + a_n x_n \equiv 0 \pmod{N} \), where all \( a_i \)s and \( N \) are known, lattice-based methods can be used to find a small solution \( (x_1, \ldots, x_n) \) of this equation whenever \( \prod_{i=1}^{n} x_i \leq N \), where \( x_1, \ldots, x_n > 0 \) are integers such that \( |x_1| \leq X_1, \ldots, |x_n| \leq X_n \). This folklore result is justified in [9, Appendix A]. The non-modular case can also be solved under a similar condition [10, Chapter 2].

2.3. An integer linear equation with two variables

In this section, we review how to find solutions of the integer linear equation with two variables, namely,

\[
ax + by = c, \quad a \neq 0, \quad b > 0,
\]

in a bounded region, with \((x, y) \in [0, X] \times [0, Y] \) for given \( X, Y \in \mathbb{N} \).

First, note that there exist integer solutions of (1) if and only if \( d = \gcd(a, b) \) divides \( c \). A particular solution \((x_1, y_1)\) of (1) can be computed using the extended Euclidean algorithm. The general solution is

\[
(x, y) = (x_1 + (b/d)k, y_1 - (a/d)k), \quad k \in \mathbb{Z}.
\]

Next, we find solutions in the bounded region, \([0, X] \times [0, Y] \). Let \( f \) be \( f(x, y) = ax + by - c \). Since \( f(x, y) = 0 \) is a line, for this line to intersect with the region, at least one of \( f(0, 0) \) or \( f(X, 0) \) should be non-positive. In this case, to find solutions in the region, we follow the procedure described below.

Let \( k = \lfloor -(d/b)X_1 \rfloor \). Then \( x_1 + (b/d)k \) is near 0, and \((x_2, y_2) = (x_1 + (b/d)k, y_1 - (a/d)k)\) is a solution near the \( y \)-axis. From this solution, by adding \((b/d, -a/d)\), we obtain more solutions \((x_{i+1}, y_{i+1}) = (x_i + b/d, y_i - a/d)\) for \( i \geq 2 \). When \( x_i \) becomes larger than \( X \), we stop. Clearly, all \((x_i, y_i)\) is a solution to (1), and we can find all solutions in the region \([0, X] \times [0, Y] \).

3. Description of the encryption scheme

In this section, we describe a quadratic compact knapsack public-key cryptosystem [1]. We first describe the key generation algorithm. For the key generation, let \( J \) consist of the following integer pairs and their reverse pairs: \((1, 31), (31, 1)\),...
Moreover, those integers satisfy the following equations:

Algorithm 1

1. Randomly choose \( n - 1 \) integer pairs \( g_i' = (g_{i1}', g_{i2}') \in J \), \( i = 2, \ldots, n \) with repetition permitted.

2. Randomly choose \( 2(n - 1) \) numbers \( s_1, \ldots, s_{n-1} \) and \( t_1, \ldots, t_{n-1} \) satisfying the following requirements. (1) \( \gcd(s_i, g_{ij}) = 1 \), (2) \( \gcd(t_i, g_{ij}) = 1 \), (3) \( \gcd(s_i, s_{i+1}) = 1 \), (4) \( \gcd(t_i, t_{i+1}) = 1 \).

3. Let \( a_1 = s_1, b_1 = t_1, s_n = 1, t_n = 1 \). Compute

\[
 a_i = s_i \prod_{j=n-i+2}^{n} g_{ij}', \quad b_i = t_i \prod_{j=n-i+2}^{n} g_{ij}', \quad \text{for } i = 2, \ldots, n. \tag{2}
\]

4. Output \( A = (a_1, \ldots, a_n), B = (b_1, \ldots, b_n) \), and exit.

It is suggested that \( |s_i|_2 = |s_n \prod_{j=2}^{n} g_{ij}'|_2 - |\prod_{j=n-i+2}^{n} g_{ij}'|_2 \) and \( |t_i|_2 = |t_n \prod_{j=2}^{n} g_{ij}'|_2 - |\prod_{j=n-i+2}^{n} g_{ij}'|_2 \). In this way, \( a_1, \ldots, a_n \) have almost the same binary length. Similarly, \( b_1, \ldots, b_n \) have almost the same binary length. In addition, \( s_i \) and \( t_i \) have the binary length,

\[
|s_i|_2 = |s_n \prod_{j=2}^{n} g_{ij}'|_2 + \epsilon, \quad |t_i|_2 = |t_n \prod_{j=2}^{n} g_{ij}'|_2 + \epsilon' \quad \text{for } \epsilon, \epsilon' \in [0, 1]. \tag{3}
\]

which is used to bound the range of \( s_i \) and \( t_i \) during the attack procedure. Now we describe the key generation algorithm.

Key generation.

1. Randomly choose two cargo vectors \( A = (a_1, \ldots, a_n) \) and \( B = (b_1, \ldots, b_n) \) using Algorithm 1.

2. Randomly choose a matrix \( C = (c_{ij})_{2 \times 2} \) with determinant 1 and the length of its entries upper-bounded by a constant,

that is \( |c_{ij}|_2 = O(1) \) and write the inverse of \( C \) as \( C^{-1} \).

3. Compute

\[
\left( \begin{array}{c}
\hat{A} \\
\hat{B}
\end{array} \right) = \left( \begin{array}{ccc}
a_1 & \cdots & a_n \\
b_1 & \cdots & b_n
\end{array} \right) = C \left( \begin{array}{c}
A \\
B
\end{array} \right).
\]

4. Randomly choose two prime integers \( p \) and \( q \) slightly greater than \( 225 \sum_{i=1}^{n} \hat{a}_i \) and \( 225 \sum_{i=1}^{n} \hat{b}_i \), respectively, and compute \( N = pq \).

5. Use the Chinese remainder theorem to generate a cargo vector \( E = (e_1, \ldots, e_n) \), s.t. \( e_i \equiv \hat{a}_i \mod p, e_i \equiv \hat{b}_i \mod q \).

6. Randomly choose an invertible integer \( v \) over \( \mathbb{Z}_N \).

7. Compute \( F = (f_1, \ldots, f_n) \),

\[
f_i \equiv e_i v \mod N.
\]

The private key is \( N, p, q, C^{-1}, \) and \( v^{-1} \mod N \), and the public key is a (permutated) vector of \( F \), where permutation can be used to increase security. The message \( M = (m_1, \ldots, m_n) \), with \( m_i \in \{0, 1, 2, \ldots, 15\} \), is encrypted into the ciphertext \( c \) such that \( c = \sum_{i=1}^{n} f_i m_i^2 \). For the decryption procedure, we refer to [1]. In the rest of the paper, we assume a public key \( F = (f_1, \ldots, f_n) \) without permutation for the convenience of presentation. Of course, permutation is considered during the complexity analysis.

4. Main observation

We begin with a main observation that there exist small integers \( W_i \) such that \( W_0 f_n + W_1 f_{n-1} + W_2 f_{n-2} + W_3 f_{n-3} = 0 \). This leads to the following lemma, which is used in the first step of the attack.

Lemma 1. Let \( F = (f_1, \ldots, f_n) \) be a public key, and let \( g_i', s_i, \) and \( t_i \) be system parameters generated by the key generation algorithm in Section 3. Then there exist small integers \( W_i(\approx 2s_{n-3}f_{n-3}) \) such that

\[
W_0 f_n + W_1 f_{n-1} + W_2 f_{n-2} + W_3 f_{n-3} = 0. \tag{4}
\]

Moreover, those integers satisfy the following equations:

\[
W_0 g_{12} g_{13} g_{14} + W_1 g_{13} g_{14} s_{n-1} + W_2 g_{14} s_{n-2} + W_3 s_{n-3} = 0, \tag{5}
\]

\[
W_0 g_{22} g_{23} g_{24} + W_1 g_{23} g_{24} s_{n-1} + W_2 g_{24} s_{n-2} + W_3 s_{n-3} = 0. \tag{6}
\]
Proof. From the key generation algorithm, we know that the following equations hold,

\[
\hat{a}_i = c_{11}a_i + c_{12}b_i, \quad \hat{b}_i = c_{21}a_i + c_{22}b_i
\]

\[
e_i \equiv \hat{a}_i \mod p, \quad e_i \equiv \hat{b}_i \mod q
\]

\[
f_i \equiv e_i v \mod N,
\]

where \( C = (c_{ij})_{2 \times 2}, N = pq \). By the Chinese remainder theorem, there exist \( c'_1, c'_2 \in \mathbb{Z}_N \) such that

\[
c'_1 \equiv c_{11} \pmod{p}, \quad c'_2 \equiv c_{12} \pmod{p}, \quad (7)
\]

\[
c'_1 \equiv c_{21} \pmod{q}, \quad c'_2 \equiv c_{22} \pmod{q}. \quad (8)
\]

It follows that \( f_i \equiv (c'_1 a_i + c'_2 b_i)v \pmod{(N)} \). Let \( G_1 = \prod_{j=5}^{n} g_{ij}^{-1}, G_2 = \prod_{j=5}^{n} g_{ij}^{-1} g_{ij}' \), \( a_i = s_i \prod_{j=n+1}^{4} g_{ij}^{-1}, \) and \( b_i = t_i \prod_{j=n+3}^{4} g_{ij}^{-1} g_{ij}' \) for \( i \in \{ n-3, \ldots, n \} \). Note that \( |a_i|_2 \approx |s_{n-3}g_{12} g_{14}'|_2 \) and \( |b_i|_2 \approx |t_{n-3}g_{12} g_{14}'|_2 \).

Using (2), we rewrite \( f_n, \ldots, f_{n-3} \) as

\[
f_n \equiv v(c'_1 s_{n} g_{12} g_{14}' g_{13}' g_{15} g_{16} g_{18} g_{24} g_{22} g_{21} g_{23} g_{24} g_2) = v(c'_1 a_n G_1 + c'_2 b_n G_2) \pmod{N},
\]

\[
f_{n-1} \equiv v(c'_1 s_{n-1} s_{n} g_{12} g_{14}' g_{13}' g_{15} g_{16} g_{18} g_{24} g_{22} g_{21} g_{23} g_{24} g_2) = v(c'_1 a_{n-1} G_1 + c'_2 b_{n-1} G_2) \pmod{N},
\]

\[
f_{n-2} \equiv v(c'_1 s_{n-2} g_{12} g_{14}' g_{13}' g_{15} g_{16} g_{18} g_{24} g_{22} g_{21} g_{23} g_{24} g_2) = v(c'_1 a_{n-2} G_1 + c'_2 b_{n-2} G_2) \pmod{N},
\]

\[
f_{n-3} \equiv v(c'_1 s_{n-3} g_{12} g_{14}' g_{13}' g_{15} g_{16} g_{18} g_{24} g_{22} g_{21} g_{23} g_{24} g_2) = v(c'_1 a_{n-3} G_1 + c'_2 b_{n-3} G_2) \pmod{N}.
\]

To remove \( G_2 \) from the equations, we manipulate the above equations as

\[
\beta_{n-1} f_n - \beta_n f_{n-1} \equiv v c'_1 G_1 (a_{n} \beta_{n-1} - a_{n-1} \beta_n) = v c'_1 G_1 H_{n,n-1} \pmod{N},
\]

\[
\beta_{n-2} f_n - \beta_n f_{n-2} \equiv v c'_1 G_1 (a_{n} \beta_{n-2} - a_{n-1} \beta_n) = v c'_1 G_1 H_{n,n-2} \pmod{N},
\]

\[
\beta_{n-3} f_n - \beta_n f_{n-3} \equiv v c'_1 G_1 (a_{n} \beta_{n-3} - a_{n-1} \beta_n) = v c'_1 G_1 H_{n,n-3} \pmod{N}
\]

where \( H_{i,j} = a_i \beta_j - a_j \beta_i \). Note that \( H_{i,i} = -H_{i,i} \) and \( H_{i,j} H_{k,i} - H_{i,k} H_{j,i} + H_{j,k} H_{i,l} = 0 \). Similarly, by removing \( G_1 \), we obtain

\[
H_{n,n-2} \beta_{n-1} f_n - \beta_n f_{n-1} - H_{n,n-1} (\beta_{n-2} f_n - \beta_n f_{n-2}) = (H_{n,n-2} \beta_{n-1} - H_{n,n-1} \beta_{n-2}) f_n - H_{n,n-2} \beta_n f_{n-1} + H_{n,n-1} \beta_n f_{n-2}
\]

\[
= \beta_n (H_{n-1,n-2} f_n - H_{n,n-2} f_{n-1} + H_{n,n-1} f_{n-2}) \equiv 0 \pmod{N},
\]

Thus,

\[
V_1 = H_{n-1,n-2} f_n - H_{n,n-2} f_{n-1} + H_{n,n-1} f_{n-2} \equiv 0 \pmod{N}. \quad (9)
\]

Similarly, we can obtain the following equations:

\[
V_2 = H_{n-1,n-3} f_n - H_{n,n-3} f_{n-1} + H_{n,n-2} f_{n-2} \equiv 0 \pmod{N}, \quad (10)
\]

\[
V_3 = H_{n-2,n-3} f_n - H_{n,n-3} f_{n-2} + H_{n,n-2} f_{n-3} \equiv 0 \pmod{N},
\]

\[
V_4 = H_{n-2,n-4} f_n - H_{n,n-4} f_{n-3} + H_{n,n-2} f_{n-4} \equiv 0 \pmod{N},
\]

which means that \( V_i = k_i N \) for some small \( k_i (\approx H_{i,i}) \in \mathbb{Z}, i \in \{1, \ldots, 4\} \), and \( k_2 V_1 - k_1 V_2 = 0 \). Now we obtain

\[
k_2 V_1 - k_1 V_2 = (k_2 H_{n-1,n-2} - k_1 H_{n-1,n-3}) f_n - (k_2 H_{n,n-2} - k_1 H_{n,n-3}) f_{n-1} + k_1 H_{n,n-1} f_{n-2} - k_1 H_{n,n-1} f_{n-3}
\]

\[
= k_4 H_{n,n-1} f_n - k_3 H_{n,n-1} f_{n-1} + k_2 H_{n,n-1} f_{n-2} - k_1 H_{n,n-1} f_{n-3}
\]

\[
= H_{n,n-1} (k_4 f_n - k_3 f_{n-1} + k_2 f_{n-2} - k_1 f_{n-3}) = 0,
\]

which means that

\[
k_4 f_n - k_3 f_{n-1} + k_2 f_{n-2} - k_1 f_{n-3} = 0.
\]

since \( H_{n,n-1} \neq 0 \). Note that \( H_{n,n-1} = a_n \beta_{n-1} - a_{n-1} \beta_n = (t_n - s_{n-1} g_{12} g_{14}' g_{13}' g_{15} g_{16} g_{18} g_{24} g_{22} g_{21} g_{23} g_{24} g_2) \). Thus \( H_{n,n-1} = 0 \) implies \( t_n - s_{n-1} g_{12} g_{14}' g_{13}' g_{15} g_{16} g_{18} g_{24} g_{22} g_{21} g_{23} g_{24} g_2 \) which cannot happen due to the fact that \( g_{ij}' \neq g_{ij} \) for all the pairs in \( J \).

Now let \( d \) be the greatest common divisor of \( k_1, \ldots, k_4 \). Then \( W_0 = k_4/d, W_1 = -k_3/d, W_2 = k_2/d, W_3 = -k_1/d \) satisfies (4). The remaining Eqs. (5) and (6) can be checked by simple substitution. Note that the absolute value of \( W_i \) is small, namely \( |W_i| \approx |k_i| \approx |H_{k,i}| \approx 2a_k \beta_i \approx 2s_{n-1} t_{n-3}, \) \( \square \)
Now we are ready to describe the proposed attack.

5. The proposed attack

5.1. Description of the attack

The proposed attack is performed in several steps. Only the public key and restrictions imposed on the system parameters are used in this attack. In brief, we use Lemma 1 to derive a candidate list for $N$. Then we show how to obtain $g'_i$ in succession.

With this obtained information, it is possible to compute all secret values.

Now we describe and explain the attack procedure step by step.

5.1.1. Step 1. $W_i$'s such that $W_0f_n + W_1f_{n-1} + W_2f_{n-2} + W_3f_{n-3} = 0$

With a given public key $F = (f_1, \ldots, f_n)$, the attack begins by finding small integers $W_i$ such that

$$W_0f_n + W_1f_{n-1} + W_2f_{n-2} + W_3f_{n-3} = 0,$$

which are expected to satisfy the following equations:

\begin{align}
W_0g'_1g'_4 + W_1g'_1g'_4g'_{n-1} + W_2g'_1g'_{n-2} + W_3g_{n-3} &= 0, \quad (11) \\
W_0g'_2g'_3g'_4 + W_1g'_2g'_4g'_{n-1} + W_2g'_4g'_{n-2} + W_3g_{n-3} &= 0. \quad (12)
\end{align}

By Lemma 1, such $W_i$'s exist. To identify such integers, lattice reduction algorithms are used with a lattice $L(B)$ generated by the following matrix:

$$B = \begin{pmatrix}
1 & 0 & 0 & 0 & Mf_n \\
0 & 1 & 0 & 0 & Mf_{n-1} \\
0 & 0 & 1 & 0 & Mf_{n-2} \\
0 & 0 & 0 & 1 & Mf_{n-3}
\end{pmatrix},$$

where $M$ is a large integer. The smallest vector in this lattice is expected to be $(W_0, W_1, W_2, W_3, 0)$. We chose $M$ to be $10^{10}$, which seems adequate, and we used the BKZ algorithm\(^1\) implemented in NTL [11] to find the shortest vector in a lattice. Since we only know a permuted vector of a public key, lattice reduction is done for all combinations. Thus, it is done $\binom{n}{3}$ times, and the attack procedure continues with the vector $(W_0, W_1, W_2, W_3, 0)$ that has the smallest norm. Given a vector, all $4!$ permutations are tested in the next steps. If this does not result in the private key, the vector with the next smallest norm should be tried. In experiments, we were able to recover the secret key using the smallest norm vector except one case.\(^2\)

In the next steps, we assume knowledge of $f_n, \ldots, f_{n-3}$ and $W_i$'s, which are expected to satisfy (11) and (12).

5.1.2. Step 2. A set of 6-tuples

In this step, we find solutions for (11). Namely, we find a set $L$ such that $L = \{(g_2, g_3, g_4, u_1, u_2, u_3) | W_0g_2g_3g_4 + W_1g_3g_4u_1 + W_2g_4u_2 + W_3u_3 = 0\}$ where $g_i \in K = \{g | (g, g') \in J\}, 0 < u_1 < U_1 = 2^{5}2^{3/2}2^{5/3}, 0 < u_2 < U_2 = 2^{3}2^{3/2}2^{5/3}$, and $0 < u_3 < U_3 = 2^{3}2^{3/2}2^{5/3}$. The range for $u_i$ comes from the binary length of $s_{n-i}$ in (3).

To find such a set for all possible values $g_2, g_3, g_4 \in K$ and $u_1 (0 < u_1 < U_1)$, the corresponding values $u_2$ and $u_3$ are found by solving the integer linear equation:

$$Au_2 + Bu_3 + C = 0 \quad \text{for } A = W_2g_4, B = W_3, C = W_0g_2g_3g_4 + W_1g_3g_4u_1$$

in a bounded region $(u_2, u_3) \in [0, U_2] \times [0, U_3]$. All integer solutions of this equation can be found using an extended Euclidean algorithm as is explained in Section 2.3. The set $L$ consists of all solutions with corresponding $g_2, g_3, g_4$ and $u_1$ as 6-tuples. Note that (11) and (12) have the same solutions.

5.1.3. Step 3. A set of pairs of 6-tuples

In this step, we identify solutions satisfying both (11) and (12). This means that $(g'_i, g'_j) \in J$. We compute a set of pairs $L' \subset L \times L$ such that $L' = \{(l_1, l_2) | (g_2, g_3, g_4, u_1, u_2, u_3), (g'_i, g'_j, g'_i, u'_1, u'_2, u'_3) | l_1, l_2 \in L \text{ and } (g_i, g'_j) \in J\}$ using $L$ obtained in the previous step. To do that, we make a partition of $L$ indexed by $(g_2, g_3, g_4)$, and obtain pairs for all elements in $L \times L$. Note that these are candidates for $g'_i, s_{n-k}$, and $t_{n-k}$ for $i \in \{1, 2\}, j \in \{2, 3, 4\}$, and $k \in \{1, 2, 3\}$.

5.1.4. Step 4. Candidate $N$ and $g'_j$

Now we compute $N$ and $g'_j$ for $i \in \{1, 2\}, j \in \{2, \ldots, n\}$ from $L'$. Note that we know $f_n, \ldots, f_{n-3}$ from Step 1. With this information, we compute a candidate $N$ from each pair in $L'$. Then we try to compute all $g'_i, s_{n-k}$, and $t_{n-k}$ in succession

---

1 We used LLL\_XD followed by BKZ\_FP with block size 4, $\delta = 0.99$.
2 This case happened when the system parameter $n = 50$. The second smallest norm vector was used to find a secret key.
obtained in Steps 2 and 3. In Step 4, using these candidates, compute $t$ according to
$$\sum_{i \in J} g_i \mod N.$$ 
Correct. Note that two large primes and should be larger than any $f_i$, Thus, for a pair in $L$, let $N$ be the greatest common divisor of $V_1$ and $V_2$, divided by small prime divisors, if it exists. If $N$ is prime or not larger than $(\max_j f_i)$, we move on to the next pair. Otherwise, we try to find the corresponding $g'_i$ for $i \in \{1, 2\}$, $j \in \{2, 3, 4\}$, and we move on to the next pair if we fail. At least for one pair in $L$, we succeed if $W$'s are correct. Note that $g'_i$, $s_n-k$, and $t_n-k$ are already known from $L$ for $i \in \{1, 2\}$, $j \in \{2, 3, 4\}$, and $k \in \{1, 2, 3\}$. In the following, we now explain a method to obtain $g'_y$ in succession.

Let $G'_1 = v c'_1 \prod_{j=3}^n g'_j$ and $G'_2 = v c'_1 \prod_{j=3}^n g'_j$. Then $f_n$ and $f_{n-1}$ can be rewritten as

$$f_n \equiv g'_1 \Bigg( v c'_1 \prod_{j=3}^n g'_j \Bigg) + g'_2 \left( \prod_{j=3}^n g'_j \right) \equiv g'_1 G'_{13} + g'_2 G'_{23} \mod N,$$

$$f_{n-1} \equiv s_{n-1} \left( v c'_1 \prod_{j=3}^n g'_j \right) + t_{n-1} \left( v c'_1 \prod_{j=3}^n g'_j \right) \equiv s_{n-1} G'_{13} + t_{n-1} G'_{23} \mod N.$$ 

Solving this system yields $G'_{13}$ and $G'_{23}$ modulo $N$. By definition, it is clear that

$$G'_{i(j+1)} \equiv g'_i G'_1 \mod N.$$ 

Thus, $G'_{14}$ and $G'_{15}$ can also be computed using $g'_1, g'_4, i \in \{1, 2\}$. Now in order to obtain the rest of $G'_y$, let us rewrite $f_i$ as

$$f_i \equiv v c'_i s_i \left( \prod_{j=3}^n g'_j \right) + v c'_i t_i \left( \prod_{j=3}^n g'_j \right) \equiv s_i G'_{1(n+i-2)} + t_i G'_{2(n+i-2)} \mod N.$$ 

In this equation, $s_i$ and $t_i$ are much smaller than other values that have almost the same binary length as $N$. In fact, $s_i$ and $t_i$ are smaller than $p$ and $q$, and they have the binary length at most $\left| \prod_{j=2}^{n-i+1} g'_j \right|_2 + 1$ and $\left| \prod_{j=2}^{n-i+1} g'_j \right|_2 + 1$, respectively, according to (3). Since $|s_i| < p q = N$, $s_i$ and $t_i$ can be obtained by identifying the small solutions of this modular linear equation using folklore lattice-based methods [9, Appendix A] as briefly reviewed in Section 2.2, if we know $G'_{i(n+i-2)}$ and $G'_{2(n-i-2)}$. We use this modular equation (14) to obtain $g'_y$ in succession.

Computation of $g'_y$ in succession. Let us assume that $G'_{ij}$ is known. By guessing $(g'_1, g'_2) \in J$, we can compute $G'_{i(j+1)}$ using (13). Again assuming $f_{n-j-1}$ in the public key $F$, we try to find a small solution of the equation

$$f_{n-j-1} \equiv s_{n-j-1} G'_{i(j+1)} + t_{n-j-1} G'_{2(j+1)} \mod N$$

satisfying conditions in (3). If such a solution exists, the process continues with obtained values, $G'_{i(j+1)}, (g'_1, g'_2), s_{n-j-1},$ and $t_{n-j-1}$. Beginning with $G'_y$, which can be computed from $L$, we can compute all $G'_y$ and $g'_y$ in succession.

With respect to complexity, note that this step needs lattice reductions at most $|L|/n$ times for each candidate $N$.

5.1.5. Step 5. A private key

In this final step, the private key $(C, C^{-1}, p, q, v^{-1} \mod N)$ is computed. Since all $G'_y$ and $g'_y$ modulo $N$ are obtained during the previous steps, we know $v c'_1$ and $v c'_2$ mod $N$, since $v c'_1 \equiv G'_{1(n+1)} \mod N$. We can also compute $(c'_2/c'_1) \mod N$ as

$$c'_2/c'_1 \equiv (v c'_2)/(v c'_1) \equiv G'_{2(n+1)} / G'_{1(n+1)} \mod N.$$ 

Not that $c'_1 \equiv c_{11} (\mod p)$ and $c'_2 \equiv c_{12} (\mod p)$ from (7). Thus, we can see that $c_{11}(c'_2/c'_1) = c_{12} \equiv 0 (\mod p)$. Noting that $c'_1 \equiv c_{21}$ and $c'_2 \equiv c_{22} (\mod q)$, if we look at the above equation modulo $q$, it should not be zero since the determinant of $C$, $c_{11}c_{22} - c_{21}c_{12}$ is one. Thus we can factor $N$ by computing the greatest common divisor,

$$\gcd(N, c_{11}(c'_2/c'_1) - c_{12}) = p.$$ 

We do not know $c_{11}$ and $c_{12}$, but it is $O(1)$, so an exhaustive search on these values is sufficient. Likewise with an exhaustive search, $c_{21}$ and $c_{22}$ are obtained using $\gcd(N, c_{21}(c'_2/c'_1) - c_{22}) = q$.

With this obtained information, the remaining secrets $c'_1, c'_2, v,$ and $v^{-1} \mod N$ can be computed easily. After testing with encryption and decryption, we can be sure that the private key obtained is the correct key.

In summary, $f_1, \ldots, f_{n-3}$ are found in Step 1, and candidates for $g'_y, s_{n-k}, t_{n-k}$ for $i \in \{1, 2\}, j \in \{2, 3, 4\}, k \in \{1, 2, 3\}$ are obtained in Steps 2 and 3. In Step 4, using these candidates, $N, g'_y, g'_y$, and $f_k$ are computed. In the final step, the correct private key $(p, q, C, v, v^{-1} \mod N)$ is identified.

We briefly analyze the time complexity of this attack in the next section.

---

3 In experiments, we set the bound of $c'_y$ to be 100. For a general bound $c$, this step has time complexity $O(c^2)$. 

With Step 3. In this step, partitioning is done in Step 2. Linear equation is solved for all possible values of \( g_1 \).

Step 4. In this step, a candidate list is generated about estimation.

Step 5. This final step is done in constant time. 

### 5.2. Complexity analysis

- **Step 1.** Lattice reduction of a 4-dimensional lattice is done about \( n^4 \) times in this step. Since \( \|f_1\|_2 \) grows linearly with \( n \) and lattice reduction can be done in quadratic complexity [12], the time complexity of this step is \( O(n^6) \).
- **Step 2.** Linear equation is solved for all possible values of \( g_2, g_3, g_4 \) such that \( 0 < u_1 < 2^{\|g_1\|_2 + 1} \) in this step. Since \( g_2 < 2g_1 \), we must solve linear equations at most \( 256|K|^2 \) times with \( |K| = 42 \). This step can be done in \( O(1) \).
- **Step 3.** In this step, partitioning is done in \( O(|L|) \), and obtaining a pair is done approximately in \( O(|L'|^3) = O(|L|/|K|^2)^2 \). With \( |L| = 100 \) and \( |K| = 42 \), this is \( O(|L|^2/2^{12.4}) \). It is difficult to guess the correct value of \( |L| \). However, it does not depend on \( n \) although it is large and varied from \( 10^5 \) to \( 10^6 \) in experiments. Assuming that this is less than \( 10^6 \), this step can be done in \( O(1) \) which could be \( 2^{60.8} \).
- **Step 4.** In this step, a candidate list \( L_n \) for \( N \) is computed in \( O(|L'|) = O(2^{40.8}) \) operations. For each candidate \( N \), lattice reduction is done \( |J|/n \) times on average. For the only correct \( N \), lattice reduction is done \( |J|/n^2 \) times. Thus, lattice reduction is done \( |J|(|J_n|+n^2) \) times in this step. In experiments, \( |J_n| \) varied from 1 to 33948 and the required time heavily depended on this number. If we assume this number is \( O(n^2) \), which seems to be plausible, this step can be done in time complexity of \( O(n^5) \).
- **Step 5.** This final step is done in constant time.

Overall, our attack works in time complexity \( O(n^5) \) with heuristic assumptions for a practical value of \( n \) (\( \geq 100 \)). One may argue that for the very large key size, the proposed attack might be ineffective since it is \( O(n^5) \); however, the scheme would be inefficient also. For example, one needs very large \( n \approx 2^{80/6} \approx 10321 \) to provide comparable security to 1024-bit RSA.

In the following, we present experimental results which shows the effectiveness of the proposed attack.

### 5.3. Experiments

We implemented the proposed attack with \( n \) from 50 to 200. The required time for each instance is shown in Table 1. Experiments were done on a Q9550 (2.83 GHz) using only one core. As the result shows, our attack is practical. It breaks the cryptosystem in about 11 h for the recommended parameter \( n = 100 \). Even when \( n = 200 \), the proposed attack recovers the secret key within 4 days. Note that the timings heavily depend on the number of candidates for \( N \). However, for random keys, it is not too large and within reach even when \( n = 200 \) which is twice the suggested parameter. This clearly demonstrates that this scheme is completely insecure in the current form.

### 5.4. Discussion

The security failure of this scheme is mainly due to the secret cargo vectors \( A \) and \( B \). In particular, \( a_n \) and \( b_n \) are products of elements only in some small set \( K \), since \( s_n = t_n = 1 \). If we set \( s_n \) and \( t_n \) to be random integers such that they are too large to be guessed, it would be harder to attack this scheme. In fact, if \( s_n \approx t_n \approx 2^n \), our attack fails in Step 2; however, some information about the permutation could be leaked in this case. To avoid such leakage, \( s_n \) and \( t_n \) should be larger than that.

Recall that \( W_i \approx 2s_{n-3}t_{n-3} \) was found satisfying

\[
W_{0n} + W_{1n-1} + W_{2n-2} + W_{3n-3} = 0
\]

in the first step of the proposed attack. The remaining steps cannot be conducted without \( W_i \)'s. Since lattice-based methods allow us to find small solutions of this equation when \( |W_i| \leq (\max_{i=1}^{4}|J_i|)^{1/4} \), we need to make \( |W_i| \) large enough, such that

\[ W_1 \approx 2^n \]

We assumed a uniform distribution in the first three entries of 6-tuples in \( L \). This may not be correct; however, we believe it is reasonable for rough estimation.

### Table 1

Timing results of the proposed attack.

<table>
<thead>
<tr>
<th>( n )</th>
<th># of instances</th>
<th>Average # of candidates for ( N )</th>
<th>Average elapsed time in minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Step 1</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>5001.6</td>
<td>3.13</td>
</tr>
<tr>
<td>60</td>
<td>10</td>
<td>1192.7</td>
<td>8.20</td>
</tr>
<tr>
<td>70</td>
<td>10</td>
<td>1208.7</td>
<td>17.67</td>
</tr>
<tr>
<td>80</td>
<td>10</td>
<td>728.8</td>
<td>33.96</td>
</tr>
<tr>
<td>90</td>
<td>10</td>
<td>709.8</td>
<td>57.66</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>1158.4</td>
<td>96.39</td>
</tr>
<tr>
<td>150</td>
<td>10</td>
<td>2696.3</td>
<td>719.73</td>
</tr>
<tr>
<td>200</td>
<td>10</td>
<td>544.5</td>
<td>2988.81</td>
</tr>
</tbody>
</table>
this inequality does not hold. As is calculated in [1], \( g_1 g_2 \approx 60.86 \) for the randomly chosen key and

\[
\begin{align*}
 f_i & \approx N \approx 225^2 \left( \sum_{i=1}^{n} a_i \right) \left( \sum_{i=1}^{n} b_i \right) \approx 225^2 n^2 a_n b_n \\
 & \approx 225^2 n^2 s_n t_n \prod_{i=2}^{n} g_i g_i' \approx 225^2 n^2 s_n t_n 60.86^{n-1}.
\end{align*}
\]

Thus, by setting \( s_n \approx t_n \approx 2^{2n} \), \( (\max_{i=1}^{n^2} f_i)^{1/3} \) would be smaller than \( s_n t_n \) (\(< |W_i|\)), and the proposed attack would fail. One drawback of this modification is that the ciphertext length increases at least 50%.

We note that random paddings should be used to thwart the stereotyped message attack in [5]. For example, a message \( M \) can be encoded as \( (M', R) \) where \( M' = M \oplus \text{Hash}(R) \).

6. Conclusion

In this paper, we present a heuristic attack on a compact knapsack-based encryption scheme proposed by Wang and Hu [1] that exploits the special property of the scheme’s secret cargo vectors and restrictions on its system parameters. Our heuristic attack works roughly in time complexity \( O(n^6) \), and in our experiment, we obtained a private key in about 11 h for the recommended parameter of \( n = 100 \). We also discuss a method to thwart our attack.

Knapsack-based cryptosystems are quite interesting, particularly for efficiency reasons; however, it is difficult to make secure schemes. We hope to see many secure knapsack-based cryptosystems for the future quantum era.

Acknowledgments

The author would like to thank the anonymous reviewers for their helpful comments. This work was supported by the National Institute for Mathematical Sciences (NIMS) grant funded by the Korea government (No. A21103).

References