Knapsack cryptosystems built on NP-hard instances

Laurent Evain (laurent.evain@univ-angers.fr)

Abstract:
We construct three public key knapsack cryptosystems. Standard knapsack cryptosystems hide easy instances of the knapsack problem and have been broken. The systems considered in the article face this problem: They hide a random (possibly hard) instance of the knapsack problem. We provide both complexity results (size of the key, time needed to encypher/decypher...) and experimental results. Security results are given for the second cryptosystem (the fastest one and the one with the shortest key). Probabilistic polynomial reductions show that finding the private key is as difficult as factorizing a product of two primes. We also consider heuristic attacks. First, the density of the cryptosystem can be chosen arbitrarily close to one, discarding low density attacks. Finally, we consider explicit heuristic attacks based on the LLL algorithm and we prove that with respect to these attacks, the public key is as secure as a random key.

Introduction

The principle

It is natural to build cryptosystems relying on NP-complete problems since NP-complete problems are presumably difficult to solve. There are several versions of knapsack problems, all of them being NP-complete. Several cryptosystems relying on knapsack problems have been introduced in the eighties [9].

We are interested in the bounded version of the knapsack problem. Let $s, M, v, v_1, \ldots, v_s \in \mathbb{N}$. The problem is to determine whether there are integers $\epsilon_i, 0 \leq \epsilon_i < M$ such that $\sum_{i=1}^{s} \epsilon_i v_i = v$. In case $M = 2$, the problem is to fill a knapsack of volume $v$ with objects of volume $v_i$.

Knapsack cryptosystems are built on knapsack problems. Alice constructs integers $v_i$ (using some private key $q$) such that the cyphering map $C$ is injective: $C : \{0, \ldots, M-1\}^s \rightarrow \mathbb{N}, (\epsilon_i) \mapsto \sum \epsilon_i v_i$. The sequence $v_i$ is the public key. When Bob has a plaintext message $m \in \{0, \ldots, M-1\}^s$ for Alice, he sends the ciphertext $C(m)$. Alice decodes using her private key.

Strength and weakness of knapsack cryptosystems

The main advantage of knapsack cryptosystems is the speed. These systems attain very high encryption and decryption rates. The knapsack cryptosystem proposed by Merkle-Hellman [7] seemed to be 100 times faster than RSA for the same level of security at the time it was introduced [9].

The main weakness of knapsack cryptosystems is security. All standard knapsack cryptosystems have been broken: the Merkle-Hellman cryptosystem by Shamir and Adleman [11], the iterated Merkle-Hellmann by Brickell [3], the Chor-Rivest cryptosystem by Vaudenay in 1997 [12], ...

Two main reasons explain the fragility of knapsack cryptosystems.

First, most of these cryptosystems start with an easy instance. The knapsack problem is NP-complete and no fast algorithm to solve it is known in general. However, the knapsack problem is easy
to solve for some instances \((v_i)_{i \leq s}\): if \((v_i)\) is a superincreasing sequence in the sense that \(v_i > \sum_{j<i} v_j\), there is a very fast algorithm to solve the knapsack problem, depending linearly on the size of the data. For knapsack cryptosystems, the public key is usually a hard instance \((v_i)\) obtained as a function \(v_i = f(q, w_i)\) of an easy instance \((w_i)\) using a private key \(q\). When Alice receives the message \(C_{v_i}(m)\) encrypted with the hard instance \(v_i\), she can compute with her private key the message \(C_{w_i}(m)\) encrypted with the easy instance \(w_i\). Then she decodes easily.

One could hope that if the private key \(q\) is chosen randomly, it is impossible to recover \(q\) and the message. This intuition is wrong. As an easy instance of the knapsack problem, the initial sequence \(w_i\) carries information and this information is still present in the ciphertext in a hidden form. This makes it possible to break the system. For instance, in the Merkle-Hellmann scheme, \(w_i\) is a superincreasing sequence and Shamir has shown that it is possible to recover the initial message \(m\), even if the private key \(q\) remains unknown.

Thus, starting from an easy instance and hiding it with a random private key is structurally weak. Information can leak, whatever the random choice of the private key.

Another potential weakness of knapsack cryptosystems is the possibility of low density attacks. Usually the numbers \((v_i)_{i \leq s}\) used as the public key are large numbers and the density \(d = s/\max \log_2(v_i)\) is low. In this case, the elements \((\epsilon_i)\) of the translated lattice \(L\) defined by the equation \(\sum \epsilon_i v_i = C(m)\) are expected to be large, and the plaintext message \(m\) sent by Bob to Alice is expected to be the smallest element in \(L\). Besides this heuristic argument, this circle of ideas yields a provable reduction of the knapsack problem to the closest vector problem CVP (CVP consists in finding the closest point to a fixed point \(P\) in a lattice). In particular, using polynomial time algorithms to approximate CVP [1], the knapsack problem is solvable in polynomial time when the density is low enough and the knapsack is sufficiently general: most knapsacks of density roughly less than \(2/s\) are solvable in polynomial time [8].

When the density is low but not less than \(2/s\), there is no known polynomial time algorithm to solve knapsack problems. However, one can still reduce knapsack problems to CVP. The embedding method reduces CVP to the shortest vector problem SVP with high probability when the density \(d\) of the knapsack is low enough, explicitly when \(d \leq 0.9408...\) (SVP consists in finding the shortest vector in a lattice). Although CVP is NP hard and SVP is NP-hard under randomized reductions [8], there are algorithms which solve efficiently CVP and SVP in low dimension, notably LLL based-algorithms. In practical terms, a knapsack cryptosystem should have dimension \(s\) at least 300 to avoid such attacks.

**Aim of the article**

Summing up, Alice constructs a cryptosystem starting from an instance \((w_i)_{i \leq s}\) and hides it with a private key \(q\). The public key \(v_i = v_i(q, w_i)\) is a function of \(q\) and \(w_i\). The above analysis shows that a knapsack cryptosystem is potentially weak if one starts with an easy instance \((w_i)_{i \leq s}\). To construct a robust cryptosystem, one should start with a hard instance \((w_i)_{i \leq s}\), i.e. the \(w_i\)’s should have no structure (chosen randomly). The dimension \(s\) should be at least 300. Under these conditions, breaking the cryptosystem should be as difficult as recovering the private key \(q\) since the existence of the private key is the only reason which makes the message received by Alice decipherable. In particular, the difficulty to find the private key is expected to be a measure of the security of the system.

The goal of this paper is to construct such cryptosystems which start with a random instance \((w_i)_{i \leq s}\) in high dimension \(s\) and such that finding the private key is as difficult as factorising a product of two primes.

Unlike the other knapsack cryptosystems, our construction does not include modular multiplications.
Differences and similarities between the three cryptosystems

The first of our three systems is the most natural. It is a fast system, both for encryption and decryption. The drawback is the size of the public key which goes from 0.1MB to 4.9MB depending on the level of security considered.

The size of the public key is subject to debate. Some authors want a short key. Other authors (see [4]) think that the concept of a small key should be questioned, and that, in view of the transmission rates on the Internet today, it is preferable to have a fast and secure system than a system with a small public key.

The sizes of the keys considered in the first system are large. Though they could be compatible with the transmission rates on the internet or the size of the memory of modern computers, it is nevertheless desirable to shorten the keys. We thus construct a second system based on the same ideas with a shorter key. The size of the key starts from 0.03MB for a reasonably secure system (corresponding to a knapsack problem with \( s = 500 \) elements), and is around 0.1MB in dimension \( s = 1000 \).

Our third cryptosystem is a hybrid between the two first cryptosystems. The key is not much longer than in the second cryptosystem, but the private key has been hidden more carefully and the system is more secure.

Our three cryptosystems have in common the same underlying one-way function based on the following remark: it is fast to produce divisions \( n_i = qx_i + r_i \) with small rests \( r_i \ll q \) (choose \( q, x, r_i \) and compute \( n_i \)) but it takes more time to recover the divisions once the numbers \( n_i \) are given. For instance, if there is one number \( n \) and we look for the smallest rest \( r = 0 \) in a division \( n = qx + r \), it means that we try to find a factorisation of \( n \). The security of the RSA system relies on the difficulty to factorize a product of two primes \( n = qx \). Thus our one way function can be seen as a generalisation of the one way function used in the RSA system. Section 1.2 explains this one-way function with more details.

The results

We provide complexity results, experimental results, and security results for the cryptosystems.

Complexity results

There are various possible choices for the parameters. There are two base parameters \( s, p \), with \( s = o(p) \) and the other parameters depend on \( s \) and \( p \). The complexity results for the first system are as follows, where \( \epsilon \) is an arbitrarily small positive number.

**Theorem 1.**

Size of the public key \( x_s \): \( O(s^2 \log_2(p)) \)

Size of the private key \( \epsilon, q_i, \sigma, \tau \): \( O(s^2 \log_2(p)) \)

Encryption time: \( O(s^2 \log_2(p)) \)

Decryption time: \( O(s^2 \log_2(p))^{1+\epsilon} \)

Creation time of the public key: \( O(s^3 \log^2(p)^{1+\epsilon}) \)

Density of the knapsack associated with \( x_s \): \( 1/ \log_2(p) \).

The complexity results for the second system are the following:

**Theorem 2.** Size of the public key \( x_1 \): \( O(s^2 + s \log_2(p)) \)

Size of the private key: \( O(s^2 + s \log_2(p)) \)

Encryption time: \( O(s^2 + s \log_2(p)) \)

Decryption time: \( O(s^2 + \log_2(p)^{1+\epsilon}) \)
Density of the knapsack associated with \( x_s = \frac{1}{1 + \frac{1}{2} \log_2(p)} \).

For the parameters chosen as in variant 2, we have:

**Theorem 3.** Size of the public key \( x_1 : O(s^2 \log_2(p)) \)
Size of the private key : \( O(s^2 + s \log_2(p)) \)
Encryption time: \( O(s^2 + s \log_2(p)) \)
Decryption time: \( O(s^2 + \log_2(p)^{1+\epsilon}) \)
Time needed to create the public key: \( O(s^2 + s \log_2(p)) \)
Density of the knapsack associated with \( x_s = \frac{1}{1 + \frac{1}{2} \log_2(p)} \).

By construction, the third system is a hybrid mixing the first and second system. For brevity, we have not included its complexity results which can be computed as for the previous two systems.

**Experimental results for the first system**

We report experiments to show that encryption/decryption time is acceptable in high dimension. The processor used is an Intel Xeon at 2GHz. The programs have been written with the software Maple (slow high level language manipulating natively arbitrarily large integers).

| s \( \times \) \( 10^6 \) | \( 10^9 \) | \( 10^{12} \) | \( 10^{15} \) | \( 10^{18} \) |
|-----------------|--------|--------|--------|--------|
| Encryption time in seconds |
| 200 | 0.002 | 0.001 | 0.001 | 0.001 | 0.001 |
| 400 | 0.001 | 0.001 | 0.001 | 0.002 | 0.002 |
| 600 | 0.001 | 0.002 | 0.002 | 0.002 | 0.144 |
| 800 | 0.003 | 0.002 | 0.003 | 0.004 | 0.261 |
| Decryption time in seconds |
| 200 | 0.150 | 0.152 | 0.166 | 0.178 | 0.209 |
| 400 | 0.480 | 0.481 | 0.587 | 0.872 | 0.872 |
| 600 | 1.019 | 1.025 | 1.182 | 2.343 | 2.099 |
| 800 | 1.597 | 1.602 | 1.809 | 3.813 | 3.314 |
| Time for generating the key in seconds |
| 200 | 0.543 | 0.602 | 0.713 | 0.850 | 0.965 |
| 400 | 3.121 | 3.707 | 4.984 | 9.933 | 11.155 |
| 600 | 12.127 | 14.164 | 18.500 | 46.012 | 52.045 |
| 800 | 25.940 | 31.376 | 37.769 | 113.364 | 118.746 |

| s \( \times \) \( 10^6 \) | \( 10^9 \) | \( 10^{12} \) | \( 10^{15} \) | \( 10^{18} \) |
|-----------------|--------|--------|--------|--------|
| Size of the key in MegaBytes |
| 200 | 0.107 | 0.157 | 0.207 | 0.257 | 0.307 |
| 400 | 0.430 | 0.628 | 0.828 | 1.027 | 1.226 |
| 600 | 0.966 | 1.413 | 1.864 | 2.312 | 2.760 |
| 800 | 1.720 | 2.514 | 3.312 | 4.111 | 4.908 |

**Experimental results for the second system.**

| s \( \times \) \( 10^6 \) | \( 10^9 \) | \( 10^{12} \) | \( 10^{15} \) | \( 10^{18} \) |
|-----------------|--------|--------|--------|--------|
| Encryption time in seconds |
| 500 | 0.002 | 0.001 | 0.001 | 0.000 | 0.001 |
| 800 | 0.001 | 0.002 | 0.002 | 0.001 | 0.001 |
| 1100 | 0.001 | 0.001 | 0.001 | 0.008 | 0.002 |
| 1400 | 0.002 | 0.001 | 0.002 | 0.002 | 0.001 |
| 1700 | 0.001 | 0.002 | 0.002 | 0.002 | 0.002 |
| 2000 | 0.002 | 0.002 | 0.003 | 0.003 | 0.002 |
Security results

We now come to the security analysis of the cryptosystems. Among the three cryptosystems described, it is easier to attack the second cryptosystem (shortest key, built to be fast, no special care to hide the private key). Thus we concentrate our analysis for this second system.

First, we remark on the above formulas that the density can be as close to 1 as possible with a suitable choice of the parameters. Thus the parameters can be chosen to avoid low density attacks.

We show that finding the private key \( q \) is as difficult as factorising a number \( n \) which is a product of two primes: if it is possible to find the private key \( q \) in polynomial time, then \( \forall \eta > 0 \), it is possible to factorise \( n = pq \) in polynomial time with a probability of success at least \( 1 - \eta \) (theorem 22).

In fact, our result is a little more precise. The private key \( q \) is an integer with suitable properties. One could use a “pseudo-key” \( q' \), i.e. an integer with the same properties as \( q \), to cryptanalyse the system. Our result says that finding a pseudo-key \( q' \) with the help some extra-information is as difficult as factorising a product of primes (i.e. there is a polynomial probabilistic reduction as above). Moreover, the system is more secure if \( q \) is the only integer with the required properties. We give evidences in section 4.1 that one can construct with high probability a cryptosystem with \( q \) as the only pseudo-key.

The above results express that it is difficult to find a pseudo-key. But the cryptosystem could still be attacked by heuristic attacks. Since most heuristic attacks rely on the LLL-algorithm and its improvements, we consider the standard attack relying on the LLL-algorithm and the embedding method.

NP-completness and many experiments lead to the conclusion that the knapsack problem is not solvable for a random instance \( x_0 = (v_1, \ldots, v_s) \) in high dimension \( s \). The public key is not a random instance \( x_0 \) but a slight deformation \( x_1 \) of \( x_0 \). A weakness appears if the heuristic attacks perform better when the random \( x_0 \) is replaced by \( x_1 \).

Our result (theorem 29) says in substance that, if \( x_0 \) is very general, replacing \( x_0 \) by a suitable \( x_1 \) is not dangerous: both the number of steps to perform the algorithm and the probability of success

| \( s \backslash p \) | \( 10^6 \) | \( 10^7 \) | \( 10^8 \) | \( 10^9 \) | \( 10^{10} \) | \( 10^{11} \) |
|---|---|---|---|---|---|---|
| 500 | 0.003 | 0.007 | 0.001 | 0.002 | 0.002 | 0.002 |
| 800 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 |
| 1100 | 0.004 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 |
| 1400 | 0.005 | 0.004 | 0.005 | 0.005 | 0.004 | 0.004 |
| 1700 | 0.014 | 0.005 | 0.005 | 0.006 | 0.006 | 0.006 |
| 2000 | 0.015 | 0.006 | 0.006 | 0.007 | 0.007 | 0.007 |

| \( s \backslash p \) | \( 10^6 \) | \( 10^7 \) | \( 10^8 \) | \( 10^9 \) | \( 10^{10} \) | \( 10^{11} \) |
|---|---|---|---|---|---|---|
| 500 | 0.056 | 0.056 | 0.057 | 0.058 | 0.057 | 0.057 |
| 800 | 0.091 | 0.092 | 0.094 | 0.094 | 0.094 | 0.094 |
| 1100 | 0.129 | 0.127 | 0.133 | 0.127 | 0.125 | 0.125 |
| 1400 | 0.166 | 0.168 | 0.165 | 0.169 | 0.169 | 0.169 |
| 1700 | 0.199 | 0.198 | 0.205 | 0.203 | 0.210 | 0.210 |
| 2000 | 0.239 | 0.237 | 0.244 | 0.245 | 0.254 | 0.254 |

| \( s \backslash p \) | \( 10^6 \) | \( 10^7 \) | \( 10^8 \) | \( 10^9 \) | \( 10^{10} \) | \( 10^{11} \) |
|---|---|---|---|---|---|---|
| 500 | 0.034 | 0.035 | 0.036 | 0.037 | 0.039 | 0.039 |
| 800 | 0.084 | 0.086 | 0.088 | 0.090 | 0.092 | 0.092 |
| 1100 | 0.157 | 0.159 | 0.162 | 0.165 | 0.168 | 0.168 |
| 1400 | 0.252 | 0.255 | 0.259 | 0.262 | 0.266 | 0.266 |
| 1700 | 0.370 | 0.374 | 0.378 | 0.382 | 0.387 | 0.387 |
| 2000 | 0.510 | 0.515 | 0.520 | 0.525 | 0.530 | 0.530 |
are unchanged. In other terms, with respect to LLL-attacks, the system is as secure if the message is
cyphered with $x_0$ or with a suitable $x_1$.

Acknowledgments
Nice surveys on knapsack cryptosystems made the subject accessible to me. I am in particular grateful
to the authors of [8], [9] and [2].

1 First system

1.1 Description of the system

We denote by $M_{p\times q}(A)$ the set of $p \times q$ matrices with coefficients in the set $A$.

- **List of parameters:** $M, s \in \mathbb{N}$, $\epsilon \in M_{s\times s}(\mathbb{N})$, $p_1, \ldots, p_s, q_1, \ldots, q_s \in \mathbb{N}$, $x_0 \in M_{1\times s}(\mathbb{N})$,
- **Message to be transmitted:** a column vector $m \in \{0, 1, \ldots, M - 1\}^s = M_{s\times 1} \{0, \ldots, M - 1\}$.
- **Private key:**
  - An invertible matrix $\epsilon \in M_{s\times s}(\mathbb{N})$ with rows $\epsilon_1, \ldots, \epsilon_s$. We let $||\epsilon_i||_1 = \sum_{j=1}^{s} \epsilon_{ij}$ the norm of
    the $i^{th}$ row.
  - A $s$-tuple of positive rational numbers $\lambda_i = \frac{a_i}{q_i}, i = 1, \ldots, s$ such that $(M - 1)\lambda_i ||\epsilon_i||_1 < 1$.
- **Recursive Construction:** Choose a random row vector $x_0 \in \mathbb{N}^s$. Define the row vector $x_1$, $i = 1 \ldots s$ by $x_i = q_i x_{i-1} + p_i \epsilon_i$.
- **Public key:** $x_s$
- **Cyphered message:** $x_s m \in \mathbb{N}$.

**Notation 4.** We denote by $C$ the cyphering function $\{0, 1, \ldots, M - 1\}^s \rightarrow \mathbb{N}$, $m \mapsto N_s = x_s m$

**Proposition 5.** The function $C$ is injective.

It suffices to explain how to decrypt to prove the proposition. We define $N_i$, $0 \leq i \leq s$ and $O_i$, $1 \leq i \leq s$ by decreasing induction:

- $N_s = C(m) = x_s m$
- $N_{i-1} = \lfloor \frac{N_i}{q_i} \rfloor$, where $\lfloor \rfloor$ denotes the integer part
- $O_i = (N_i - q_i N_{i-1})/p_i$.
- Let $N \in M_{s+1\times 1}(\mathbb{N})$ be the column vector with entries $N_0, \ldots, N_s$
- Let $O \in M_{s\times 1}(\mathbb{Q})$ be the column vector with entries $O_1, \ldots, O_s$.
- Let $X \in M_{s+1\times s}(\mathbb{N})$ be the matrix with rows $x_0, \ldots, x_s$.

**Proposition 6.** The message $m$ verifies $X m = N$, $em = O$. In particular, the coefficients of $O$ are integers.

**Proof.** We prove that $x_i m = N_i$ by decreasing induction on $i$. The case $i = s$ is true by definition. If $x_i m = N_i$, then $(x_{i-1} + \lambda_i \epsilon_i) m = N_{i-1}$. Since $x_{i-1} m \in \mathbb{N}$ and $0 < \lambda_i \epsilon_i m \leq \lambda_i ||\epsilon_i||_1 (M - 1) < 1$ by hypothesis, we obtain $x_{i-1} m = [N_i/q_i] = N_{i-1}$, as expected. Thus $\epsilon_i m = (x_i - (q_i x_{i-1})) m/p_i = (N_i - q_i N_{i-1})/p_i = O_i$.

**Corollary 7.** To decrypt the message,

- Compute $N_{s-1}, \ldots, N_1$ with the formula $N_{i-1} = \lfloor \frac{N_i}{q_i} \rfloor$.
- Compute $O_i = (N_i - q_i N_{i-1})/p_i$.
- Solve the system $em = O$. 

6
1.2 Analysis of the system

The underlying one way function

We make a quick analysis of the system.

The couple \( (q_s, \epsilon_s) \) in the private key satisfies \( x_s = q_s x_{s-1} + p_s \epsilon_s \) with \( q_s > p_s ||\epsilon_s||_1(M - 1) \). Componentwise, \( p_s \epsilon_{si} \) is the rest of the division of \( x_{si} \) by \( q_s \). These rests are small. The rest of the division of \( x_{si} \) by \( q_s \) is at most \( q_s \), and the sum of the rests \( p_s \epsilon_{si} \) for \( 1 \leq i \leq s \) is at most \( sq_s \) in general.

In the present situation, the sum \( \sum_{i=1}^{s} p_s \epsilon_{si} = p_s ||\epsilon_s||_1 \) of all the rests is at most \( \frac{q}{M-1} \).

In other words, an eavesdropper who tries to break the system looks for an integer \( q_s \) such that the rests of the divisions of the \( x_{si} \) by \( q_s \) are unusually small. The sum of the \( s \) rests is at most \( \frac{q}{M-1} \).

There is hopefully a one way function here. It is easy to construct a couple of integers \( (x, q) \) such that the rest of the division of \( x \) by \( q \) is small. But once \( x \) is given, it is not easy to find back an integer \( q \) such that the rest of the division of \( x \) by \( q \) is small.

For instance, to obtain a rest which is at most \( \frac{1}{10^n} \) of the divisor \( q \), choose any \( y, q \in \mathbb{N}, 0 \leq \epsilon \leq q/10^n \) and put \( x = qy + \epsilon \). As a function of \( q \), the number of operations to compute \( x \) is \( O(\log_2(q)) \). If \( x \) is given and Eve knows that there is a \( q \) satisfying \( x = qy + \epsilon, 10^n \epsilon < q \), trying successively all possible divisors \( 1, \ldots, q \) requires \( O(q) \) operations.

Thus, in the absence of a quick algorithm to find \( q \), there is a gain of an exponential factor here. In our choice of parameters, the numbers \( q_i \) will be large to make the most of this advantage.

Construction of the matrix \( \epsilon \)

The matrix \( \epsilon \) of the private key should be quickly invertible, for instance triangular, to facilitate decryption (see corollary 7). But a triangular matrix \( \epsilon \), or any matrix with a lot of null coefficients, would be a bad choice. Indeed, \( \epsilon \) is sparse, there are two components \( c, c' \) of \( x_s = q_s x_{s-1} + p_s \epsilon_s = (\ldots, c, \ldots, c', \ldots) \) whose gcd is a multiple of \( q_s \), or \( q_s \) itself. After several attempts, the eavesdropper could find \( q_s \).

The same problem occurs if the components of \( \epsilon_s \) are too small or well localised by a law of repartition. If \( x_s = (\ldots, c, \ldots, c', \ldots) \), there is a natural attempt to find \( q_s \): test for the gcd of \( (c - c', c' - c'') \) for several values of \( c', c'' \).

Summing up, the matrix \( \epsilon \) should satisfy the two following conditions:
- its coefficients are difficult to localize,
- solving \( cm = O \) is fast.

If the coefficients of the matrix \( \epsilon \) are chosen randomly, it takes time to solve \( \epsilon m = O \). If we choose a lower triangular matrix \( L \), an upper triangular matrix \( U \) with random uniform coefficients, and choose \( \epsilon = LU \), then it is easy to solve the system but the coefficients of \( \epsilon \) are not random uniform and this non uniformity could be used to cryptanalyse the system as explained above.

Thus there is a compromise to find between the amount of time required to compute and invert \( \epsilon \) and the uniformity in the coefficients of \( \epsilon \). Our approach to find the compromise is to consider an upper triangular matrix \( U \) with random coefficients and to deform it using elementary operations (proposition 8).

Let \( L, N \in M_{s \times s}(\mathbb{N}) \) be the lower triangular matrices defined by \( L_{ii} = N_{ii} = 1 \), \( L_{i,1} = 1 \), \( N_{n,i} = 1 \) and all other coefficients equal to zero. If \( \sigma \) is a permutation of \( \{1, \ldots, s\} \), we denote by \( M_\sigma \) the permutation matrix defined by \( M_{i,\sigma(i)} = 1 \) and \( M_{ij} = 0 \) otherwise.

**Proposition 8.** Let \( U \in M_{s \times s}(\mathbb{N}) \) be an upper invertible triangular matrix with coefficients \( u_{ij}, i \leq j \) chosen randomly in \( \{1, \ldots, x\} \) and \( \sigma, \tau \) be permutations of \( \{1, \ldots, s\} \). Then every entry \( e \) of the matrix \( \epsilon(s, x) = M_\sigma L U N M_\tau \) verifies \( 0 \leq e \leq 4x \). In particular, the norm of the lines \( \epsilon_i \) satisfy \( ||\epsilon_i||_1 \leq 4sx \).

**Proof.** The action of the permutations \( \sigma, \tau \) permute the coefficients of \( L U N \) so one can suppose \( \sigma = \tau = \text{Identity} \). An entry in \( U \) is in \( \{0, \ldots, x\} \). The left multiplication with \( L \) replaces a line \( L_i, i > 1 \)
with \(L_i + L_1\). The right multiplication with \(N\) replaces a column \(C_i, i < s\) with \(C_i + C_s\). Thus an entry of \(LUN\) is in \(\{0, \ldots, 4^r\}\).

### 1.3 Suggested choice for the parameters

In this section, suggestions for our list of parameters \(M, s \in \mathbb{N}, \epsilon \in M_{s \times s}(\mathbb{N}), p_1, \ldots, p_s, q_1, \ldots, q_s \in \mathbb{N}, x_0 \in M_{1 \times s}(\mathbb{N})\) are given. We fix two integers \(s, p\) as based parameters. The other parameters are constant or functions of \(s\) and \(p\).

The level of security depends on the size of \(s\) and \(p\). To give an idea of the size of the numbers involved, \(s > 300\) and \(p > 10^6\) are sensible choices.

**Suggested choice for the parameters as constants or functions of \(s, p\):**

- \(M = 2\)
- \(\epsilon = \epsilon(s, \lfloor p/4s \rfloor)\) is the random matrix considered in proposition 8
- \(p_i = 1, q_i\) chosen randomly in \([p+1, 2p]\) (uniform law)
- \(x_0\) has entries chosen randomly in \([0, 2^s]\) (uniform law)

**Comments on the choices.**

The choice \(M = 2\) is to make the system as simple as possible. Moreover, Shamir has shown that compact knapsack cryptosystems (ie. those with messages in \(\{0, \ldots, M-1\}^s\) and small \(M\)) tend to be more secure [10].

The reason for the choice of the matrix \(\epsilon\) has been given before proposition 8 (compromise between randomness and inversibility). Note that the required condition \((M - 1) || |\epsilon_i| \lambda_i < 1\) is satisfied by proposition 8.

As to the choice of \(\lambda_i = \frac{2^s}{q_i}\), we have explained that \(q_i\) is large to make the most of the one way function. Looking at the recursive definition of \(x_i\), it appears that the \(x_i\)'s are large when \(p_i\) is large. Thus we take \(p_i = 1\) to limit the size of the key.

The entries of the initial vector \(x_0\) are chosen randomly in \([0, 2^s]\) so that the density of the knapsack cryptosystem associated to \(x_0\) is expected close to one. If the density is lower, there could be a low density attack on \(x_0\), and maybe an attack on \(x_s\) as \(x_s\) is a modification of \(x_0\). On the other hand, it is not clear that a higher density is dangerous. It could even be a better choice. Experiments are needed to decide. Thus we propose a variant of higher density:

**Variant for the choice of parameters**

- \(x_0\) has entries chosen randomly in \(\{0, \ldots, 2^s\}\).
- All other parameters are chosen as before.

### 1.4 Complexity results

The complexity of the cryptosystem is described in the following theorem, using the first variant for the choice of parameters (ie. \(x_0\) has entries in \(\{0, \ldots, 2^s\}\)).

We denote by \(\text{size}(A)\) the number of bits needed to store an element \(A\) and by \(\text{time}(A)\) the number of elementary operations needed to compute \(A\). Recall that, for all \(\epsilon > 0\), computing a multiplication of two integers \(p\) and \(q\) takes \(\text{time}(pq) = O(\text{size}(p) + \text{size}(q))^{1+\epsilon}\) elementary operations [5]. Moreover, the complexity of a division is the same as the complexity of a multiplication.
Theorem 9. Suppose that $s = o(p)$. Then:
Size of the public key $x_s$: $O(s^2 \log_2(p))$
Size of the private key $e, q, \sigma, \tau$: $O(s^2 \log_2(p))$
Encryption time: $O(s^2 \log_2(p))$
Decryption time: $O(s^2 \log_2(p))^{\epsilon^+}$
Creation time of the public key: $O(s^3 \log_2(p)^{\epsilon^+})$
Density of the knapsack associated with $x_s$: $1/\log_2(p)$.

Proof.
\begin{itemize}
  \item $||\epsilon_i||_{\infty} \leq p$
  \item $\text{size}(||\epsilon_i||_{\infty}) = O(\log_2(p))$
  \item $\text{size}(\epsilon_i) \leq s \text{ size}(||\epsilon_i||_{\infty}) = O(s \log_2(p))$
  \item $\text{size}(e) = \sum_i \text{size}(\epsilon_i) = O(s^2 \log_2(p))$
  \item $\text{size}(q_1, \ldots, q_s) = O(s \log_2(p))$
  \item $\text{size}(\sigma) = \text{size}(\tau) = \text{time}(\sigma) = \text{time}(\tau) = O(s \log_2(s))$
  \item $\text{size}(\text{private key}) = \text{size}(e, q_1, \ldots, q_s, \sigma, \tau) = O(s^2 \log_2(p))$
  \item $||x_i = q_i x_{i-1} + \epsilon_i||_{\infty} \leq ||q_i||_{\infty} ||x_{i-1}||_{\infty} + ||\epsilon_i||_{\infty} \leq 2p ||x_{i-1}||_{\infty} + p$ thus $||x_i||_{\infty} \leq 3p^i ||x_0||_{\infty}$.
  \item $\text{size}(||x_i||_{\infty}) = O(s \log_2(p) + \text{size}(||x_0||_{\infty})) = O(s \log_2(p) + s)$
  \item $\text{size}(x_i) \leq \text{size}(||x_i||_{\infty}) = O(s \log_2(p) + s^2)$
  \item $\text{size}(\text{public key}) = \text{size}(x_s) = O(s^2 \log_2(p))$
  \item $\text{encryption time} = \text{size}(\text{public key}) = O(s^2 \log_2(p))$
  \item $\text{time}(x_i) = O(\text{size}(q_i)^{1+\epsilon} + \text{size}(x_{i-1})^{1+\epsilon} + \text{size}(\epsilon_i)) = O(\text{size}(x_{i-1})^{1+\epsilon} + O((s \log_2(p) + s^2)^{1+\epsilon})) \leq O((s^2 \log_2(p)^{1+\epsilon}))$
  \item $\text{time}(N_i) = [N_{i+1}/q_i] = O(\text{size}(q_i)^{1+\epsilon} + \text{size}(N_{i+1})^{1+\epsilon}) = O(\log_2(p)^{1+\epsilon} + \text{size}(x_{i+1}m)^{1+\epsilon}) \leq O(\log_2(p)^{1+\epsilon} + \text{size}(s \log_2(p))^{1+\epsilon} \leq O((s \log_2(p))^{1+\epsilon})$
  \item $\text{time}(N_0, \ldots, N_s) = O(\log_2(p)s^2)^{1+\epsilon}$.
  \item $\text{time}(O_i = (N_i - q_i N_{i-1})) = O(\text{time}(N_i))$
  \item $\text{time}(N_0, \ldots, N_s, O_1, \ldots, O_s) = \text{time}(N_0, \ldots, N_s) = O(\log_2(p)s^2)^{1+\epsilon}$
\end{itemize}

To solve the linear $cm = O$ with $\epsilon = M_s LUNM_{\tau}$. we first suppose that $\epsilon = U$ (ie. $M_s = L = N = M_{\tau} = Id$). The entries $e$ in $e$ and $O$ satisfy $\text{size}(e) = O(\log_2(p))$. Since $\epsilon = U$ is triangular, solving the system takes a time $\tau = O(s^2 \log_2(p))^{1+\epsilon}$. We have $\text{time(\text{decryption})} = \text{time}(N_1, \ldots, N_s, O_1, \ldots, O_s, \text{solving}(\epsilon.m = O))$, thus the decryption takes $O(s^2 \log_2(p))^{1+\epsilon}$ operations.

Remark 10. \textit{These theoretical results are consistent with the experimental results of the introduction.}

2 Second system

2.1 Description of the system

Since the size of the key is a bit large, we propose a second system to reduce the size of the key. The implicit one way function is the same as before. We only change the private key and take a superincreasing sequence instead of an invertible matrix.

\begin{itemize}
  \item List of parameters: $M, s \in N, \epsilon \in N^p, p_1, q_1 \in N, x_0 \in M_{1 \times s}(N)$, a permutation $\sigma$ of $\{1, \ldots, s\}$
\end{itemize}
• **Message to be transmitted**: a column vector \(m \in \{0, 1, \ldots, M-1\}^s\).

• **Private key**:
  - A permutation \(\sigma\) of \(\{1, \ldots, s\}\)
  - A row matrix \(\epsilon \in M_{1 \times s}(\mathbb{N})\) such that the sequence \(\epsilon_{\sigma(1)}, \ldots, \epsilon_{\sigma(s)}\) is a superincreasing sequence.
  - A positive rational number \(\lambda_1 = \frac{\ell_1}{q_1}\), such that \((M-1)\lambda_1 ||\epsilon||_1 < 1\).

• **Construction**: Choose a random row vector \(x_0 \in \mathbb{N}^s\). Define the row vector \(x_1\) by \(x_1 = q_1 x_0 + p_1 \epsilon\).

• **Public key**: \(x_1\)

• **Cyphered message**: \(x_1 m \in \mathbb{N}\).

**Notation 11.** We denote by \(C\) the cyphering function \(\{0, 1, \ldots, M-1\}^s \rightarrow \mathbb{N}\), \(m \mapsto C(m) = x_1 m\)

**Proposition 12.** The function \(C\) is injective.

It suffices to explain how to decypher to prove the proposition. We define \(N_1, N_0\), and \(O\) as follows

- \(N_1 = C(m) = x_1 m\)
- \(N_0 = \lfloor \frac{N_1}{q_1} \rfloor\)
- \(O = (N_1 - q_1 N_0)/p_1\).
- Let \(N\) be the column vector with entries \(N_0, N_1\).
- Let \(X\) be the matrix with rows \(x_0, x_1\).

The same proof as for proposition 6 shows:

**Proposition 13.** The initial message \(m\) verifies \(Xm = N, \epsilon m = O\).

Now, since \(\epsilon_{\sigma(i)}\) is a superincreasing sequence, the map \(m \mapsto \epsilon m\) is injective and the formula to decypher \(m\) expresses \(m_{\sigma(i)}\) by decreasing induction on \(i \leq s\).

**Proposition 14.**
- \(m_{\sigma(s)} = 1\) if \(O \geq \epsilon_{\sigma(s)}\) and \(m_{\sigma(s)} = 0\) otherwise
- \(m_{\sigma(i)} = 1\) if \(O - \sum_{j>i} \epsilon_{\sigma(j)} m_{\sigma(j)} \geq \epsilon_{\sigma(i)}\) and \(0\) otherwise.

### 2.2 Suggestion for the choice of the parameters

The parameters \(s\) and \(p\) depend on the required level of security and the other parameters are constant or functions of \(s\) and \(p\).

**Variant 1.** Choose:
- \(\epsilon_{\sigma(1)} \in [0, p], \epsilon_{\sigma(2)} \in [p, 2p], \ldots, \epsilon_{\sigma(s)} \in [(2^{s-1} - 1)p, 2^{s-1}p]\) (uniform law)
- \(x_0 \in [0, p]\) (uniform law)
- \(p_1 = 1, M = 2\)
- \(q_1 \in [2^s p, 2^{s+1} p]\) (uniform law)

**Variant 2.** Choose
- \(x_0 \in [0, 2^s]\) (uniform law)
- the other parameters as above.

### 2.3 Complexity results

As before, we suppose that the parameters \(s\) and \(p\) satisfy \(s = o(p)\). For the parameters chosen as in variant 1, we have:

**Theorem 15.** Size of the public key \(x_1\): \(O(s^2 + s \log_2(p))\)

Size of the private key: \(O(s^2 + s \log_2(p))\)
Encryption time: $O(s^2 + s \log_2(p))$
Decryption time: $O(s^2 + \log_2(p)^{1+\epsilon})$
Time to create the public key: $O(s^2 + \log^2(p)^{1+\epsilon})$
Density of the knapsack associated with $x_s$: $\frac{1}{1 + \frac{2}{s} \log_2(p)}$.

For the parameters chosen as in variant 2, we have:

**Theorem 16.** Size of the public key $x_1$: $O(s^2 \log_2(p))$
Size of the private key: $O(s^2 + s \log_2(p))$
Encryption time: $O(s^2 + s \log_2(p))$
Decryption time: $O(s^2 + \log_2(p)^{1+\epsilon})$
Time needed to create the public key: $O(s^2 + \log_2(p))$
Density of the knapsack associated with $x_s$: $\frac{1}{2 + \frac{1}{s} \log_2(p)}$.

For brevity, we include the proof only for variant 1. Proof. (for variant 1).

- $|x_1| = q_1 x_0 + \epsilon \leq 2^{s+1} p |x_0| \leq 2^{s+1} p^2 + 2^{s+1} p < 2^{s+2} p^2$
- $\text{size(public key)} = \text{size}(x_1) \leq s \text{size}(|x_1|) = O(s^2 + s \log_2(p))$.
- $\text{size}(\epsilon) \leq s \log_2(p) + 1 + 2 + \cdots + (s - 1) = O(s^2 + s \log_2(p))$.
- $\text{size}(q_1) = O(s + \log_2(p))$
- $\text{size}(\sigma) = O(s \log_2(s))$
- $\text{size(private key)} = \text{size}(x_0, q_1, \epsilon, \sigma) = O(s^2 + s \log_2(p))$.
- Encryption time = size(public key) = $O(s^2 + s \log_2(p))$
- $\text{size}(N_1) \leq \log_2(s||x_1||) = O(s + \log_2(p))$.
- $\text{time}(N_0) \leq O(\text{size}(N_1)^{1+\epsilon} + \text{size}(q_1)^{1+\epsilon}) = O(s^{1+\epsilon} + \log_2(p)^{1+\epsilon})$
- $N_0 \leq \frac{N_1}{4q_1} \leq \frac{2^{s+2}sp^2}{2p} = 4sp$
- $\text{size}(N_0) = O(\log_2(s) + \log_2(p))$.
- $\text{time}(O) = O(\text{size}(N_1) + \text{size}(q_1)^{1+\epsilon} + \text{size}(N_0)^{1+\epsilon}) = O(s^{1+\epsilon} + \log_2(p)^{1+\epsilon})$ since $s \leq p$.
- $O - \sum_{j > 1} \epsilon_{\sigma(j)} m_{\sigma(j)} \leq \sum_{j < 1} \epsilon_{\sigma(j)} \leq p + 2p + \cdots + 2^{(s+1)} p < 2^sp$.
- $\text{time}(m_{\sigma(i)})$ in proposition 14 = $\text{size}(O - \sum_{j > 1} \epsilon_{\sigma(j)} m_{\sigma(j)}) = O(i + \log_2(p))$.
- $\text{time}(m) = \sum_{i=1}^{s} \text{time}(m_{\sigma(i)}) = O(s \log_2(p) + 1 + 2 + \cdots + s) = O(s \log_2(p) + s^2)$.
- Decryption time = $\text{time}(N_0, O, m) = O(s^2 + \log_2(p)^{1+\epsilon})$.
- $\text{time}(\text{public key}) = \text{time}(q_1, x_0 + \epsilon) = O(\text{time}(\epsilon) + \text{time}(q_1) + \text{time}(x_0) + \text{size}(q_1)^{1+\epsilon} + \text{size}(x_0)^{1+\epsilon} + \text{size}(\epsilon)) = O(\text{size}(q_1)^{1+\epsilon} + \text{size}(x_0)^{1+\epsilon} + \text{size}(\epsilon))$ since $\text{time}(\epsilon) = O(\text{size}(\epsilon))$ and similarly for $q_1$ and $x_0$. Thus $\text{time}(\text{public key}) = O(s^2 + \log_2(p)^{1+\epsilon})$.
- Density of the knapsack = $\frac{1}{\log_2(||x_1||)} > \frac{1}{s + 2 + 2 \log_2(p)} = \frac{1}{1 + \frac{2}{s} \log_2(p)}$.

3 Third system

Two cryptosystems have been constructed so far. In the second system, the key is shorter than in the first one, but the system could be less secure because of the superincreasing sequence.

This section presents a hybrid system, a compromise between the two previous systems. We still use a superincreasing sequence to shorten the key as in the second system, but the matrix $\epsilon$ has several lines as in the first system to hide more carefully the superincreasing sequence. Hopefully, this is a good compromise between security and length of the key.
• List of parameters: $M, s \in \mathbb{N}$, $\epsilon \in M_{2 \times s}(\mathbb{N})$, $p_1, q_1, p_2, q_2 \in \mathbb{N}$, $x_0 \in M_{1 \times s}(\mathbb{N})$, $\sigma$ a permutation of $\{1, \ldots, s\}$.

• **Message to be transmitted**: a column vector $m \in \{0, 1, \ldots, M - 1\}^s$.

• **private key**:
  - A permutation $\sigma$ of $\{1, \ldots, s\}$
  - An invertible $2 \times s$ matrix $\epsilon$ with entries in $\mathbb{N}$ such that the row $\mu = \epsilon_2 - \epsilon_1$ is a superincreasing sequence with respect to the permutation $\sigma$, i.e. $\mu\sigma(1), \ldots, \mu\sigma(s)$ is a superincreasing sequence.
  - Two positive rational numbers $\lambda_i = \frac{p_i}{q_i}$, such that $(M - 1)\lambda_i||\epsilon_i|| < 1$.

• **Construction**: Choose a random row vector $x_0 \in \mathbb{N}^s$. Define the row vectors $x_1, x_2$ by $x_1 = q_1x_0 + p_1\epsilon_1$, $x_2 = q_2x_1 + p_2\epsilon_2$

• **Public key**: $x_2$

• **Cyphered message**: $N_2 = x_2m \in \mathbb{N}$.

To decrypt, we define $N_1, N_0$ and $O_2, O_1$ as before, and $\omega = O_2 - O_1$:

• Compute $N_1$ and $N_0$ with the formula $N_{i-1} = \frac{N_i}{q_i}$.

• Compute $O_i = (N_i - q_iN_{i-1})/p_i$.

• Compute $\omega = O_2 - O_1$.

Let $N = \begin{pmatrix} N_0 \\ N_1 \\ N_2 \end{pmatrix} \in M_{3 \times 1}(\mathbb{N})$ and $X = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \in M_{3 \times s}(\mathbb{N})$

The same proof as for proposition 14 shows:

**Proposition 17.** The initial message $m$ verifies $Xm = N$, $\epsilon m = O$, $\mu m = \omega$.

Now, since $\mu$ is a superincreasing sequence, the map $m \mapsto \mu m$ is injective and the formula to decipher is as in proposition 14.

### 4 Security results

In this section, we analyse the security of the second cryptographic system (section 2). We concentrate our attention on this system because it is the easiest system to attack: the key is short and no special effort has been done to hide the superincreasing sequence.

We recall the notations. The private key is $q, \epsilon_1, \ldots, \epsilon_n, x_0, \sigma$ where $x_0 = (v_1, \ldots, v_s)$, $\epsilon_\sigma(i)$ is a superincreasing sequence and $\sum_{i=1}^n \epsilon_i < q$. The public key is $x_1 = (w_1, \ldots, w_s)$ where $w_i = qv_i + \epsilon_i$.

Obviously, $\epsilon_i = w_i - \frac{[w_i]}{q}$, and $\sigma$ is determined by $\epsilon$. In other words, the whole private key is determined by $q$. We thus call $q$ the private key.

#### 4.1 Unicity of the pseudo-key

It is not necessary to find the private key $q$ to cryptanalyse. Any number $q'$ with the same properties as $q$ would do the job. We call such a number a pseudo-key. Explicitly, in our context, a pseudo-key is an integer $q'$ such that the numbers $v'_i, r_i$ defined by the euclidean divisions $w_i = q'v'_i + r_i$ verify $\sum_{i=1}^s r_i < q'$ and $(r_i)$ is a superincreasing sequence up to permutation.

If there are many pseudo-keys, it is easier to attack the system. For instance, in the Merkell-Hellman modular knapsack cryptanalysed by Shamir-Adleman, there were many pseudo-keys. The strategy of Shamir was to find a pseudo-key.

The experiments made on our cryptosystem show that usually the pseudo-key is unique. We chose random instances of the parameters and we count the percentage of cases where the pseudo-key is unique. Those results suggest that when $s > 200$, which are the cases considered in practice, the pseudo-key should be unique and equal to the private key with high probability.
Proposition 18. Consider the second cryptosystem, variant 2. The results of the experiments are as follows.

- $s = 5, 20 < p < 35$, the pseudo-key is unique in 2% of the cases.
- $s = 6, 30 < p < 45$, the pseudo-key is unique in 46% of the cases.
- $s = 7, 30 < p < 45$, the pseudo-key is unique in 79% of the cases.
- $s = 8, 40 < p < 55$, the pseudo-key is unique in 96% of the cases.

Besides this computation, we want to explain why we expect a unique pseudo-key when $s$ is large enough.

For a fixed $q'$, the rests $r_i = w_i \mod q'$ are numbers between $0 \ldots q' - 1$. In the absence of relation between $w_i$ and $q'$, these rests are expected to follow a uniform law of repartition in $\{0, \ldots, q' - 1\}$. Of course the exact law of $r_i = w_i \mod q'$ depend on the law of $w_i$ (hence of the law of $q, v_i, \epsilon_i$ as $w_i = qv_i + \epsilon_i$) and of the choice of $q'$, but a uniform law is an approximation for the law of $r_i$.

If one accepts this approximation, the next proposition is an estimation of the probability to find a $q$ such that the sum of the rests is bounded by $q$, as required for a pseudo-key.

Proposition 19. Let $q \geq 2$. Consider the rests $r_1(q), \ldots, r_s(q)$ where $r_i(q) = w_i \mod q$. Suppose that $r_1(q), \ldots, r_s(q)$ follow independent uniform laws with values in $\{0, \ldots, q - 1\}$. The probability $P$ that $\sum_{i=1}^{s} r_i(q) \leq q - 1$ satisfies $P \leq \left(\frac{q}{2}\right)^{s-1}$

Lemma 20. Let $a_1 \geq a_2 \geq \cdots \geq a_n$ and $p_1 \leq p_2 \leq \cdots \leq p_n$. Then $n \sum_{i=1}^{n} a_i p_i \leq \left(\sum_{i=1}^{n} a_i\right)\left(\sum_{i=1}^{n} p_i\right)$.

Proof. of the lemma

$\left(\sum_{i=1}^{n} a_i\right)\left(\sum_{i=1}^{n} p_i\right) - n \sum_{i=1}^{n} a_i p_i = \sum_{i=1}^{n} a_i p_i + \sum_{i=1}^{n} a_i \sum_{k=1, k \neq i}^{n} p_k - \sum_{i=1}^{n} a_i p_i - (n-1) \sum_{i=1}^{n} a_i p_i = \sum_{1 \leq i < k \leq n} (a_i - a_k)(p_k - p_i) \geq 0$.

Proof. of proposition

We have $P(r_i(q) = k) = \frac{1}{q}$ for every $k \in \{0, \ldots, q-1\}$. For $0 \leq r \leq q-1$, denote by $P_{q,s,r}$ the probability that $\sum_{i=1}^{s} r_i(q) = r$. We show by induction on $s \geq 1$ that $P_{q,s,0} \leq P_{q,s,1} \cdots \leq P_{q,s,q-1}$ and that $P_{q,s,r} \leq \left(\frac{r}{q}\right)^{s-1}$. This is obvious for $s = 1$. Note that $P_{q,s,r} = \frac{\sum_{i=0}^{r} P_{q,s-1,i}}{q}$. In particular, $\sum_{r=0}^{q-1} P_{q,s,r} = \frac{q P_{q,s-1,0} + (q-1) P_{q,s-1,1} + \cdots + P_{q,s-1,q-1}}{q} \leq \frac{q+1}{2} P_{q,s-1,0} + \cdots + P_{q,s-1,q-1}$ by the lemma. Now the induction implies that the right hand side of the inequality is bounded by $\frac{q+1}{2q} \left(\frac{3}{q}\right)^{s-2} \leq \left(\frac{4}{q}\right)^{s-1}$ for $q \geq 2$.

Proposition 21. Let $s \in \mathbb{N}$ be a fixed number and $t >> s$. Let $S_{st}$ the number of superincreasing sequences $r_1, \ldots, r_s$ with sum $t$ and $C_{st}$ the number of sequences with sum $t$. Then $\frac{C_{st}}{S_{st}}$ is asymptotically equal to $\frac{1}{2^{t-1}}$ when $t$ tends to infinity.

Proof. The number of sequences $r_1, \ldots, r_s$ with sum $t$ is $\binom{t+s-1}{s-1}$ and is equivalent to $\frac{t^{s-1}}{(s-1)!2^{t-1}}$. Remark that $S_{st} = \sum_{i=1}^{\lfloor t/p \rfloor} S_{s-1,i}$. By induction on $s$, $S_{st}$ is equivalent to $\frac{t^{s-1}}{(s-1)!2^{t-1}}$.

Summing up the situation, a number $q$ is a pseudo-key if the sum of the rests $r_i(q)$ is less than $q$ and if these rests form a superincreasing sequence. By proposition 19 the probability for the first condition is less than $\left(\frac{4}{q}\right)^{s-1}$. And by proposition 21 the probability that the second condition is satisfied is around $\frac{1}{2^{t-1}}$.

In particular we expect a unique pseudo-key $q$ when the number of possible values for $q$ is asymptotically dominated by $\left(\frac{4}{q}\right)^{s-1} \frac{t^{s-1}}{(s-1)!2^{t-1}}$. This is the case for the second system we have constructed with the suggested choices of parameters and this gives an explanation to the results of proposition 18.
Theorem 22. If it is possible to solve two primes, the input data), then why the pseudo-key is unique for many cryptosystems. Thus the security of the system relies on the factorisation of $n$. In particular, if one can find a factorisation of $n$, then it is more difficult to find all the keys than to find one key, and the problem is easier when more information is given as input, as long as the definition of “more difficult” is sensible (polynomial time reduction, probabilistic polynomial time reduction ...). In particular, if $\eta > 0$, then problem 1 is more difficult than problem 2, and problem 1 is more difficult than problem 3. The previous section explained why the pseudo-key is unique for many cryptosystems. Thus the security of the system relies on the difficulty of solving Problem 4. We show that solving Problem 4 is as difficult as factorising a product of two primes.

4.2 Finding a pseudo-key is as difficult as factorising an integer

In this section, we show that the problem of finding the exact value of the private key $q$ is as difficult as factorising an integer $n$, product of two primes. More precisely, we show that an easier problem (finding a pseudo-key with the help of some extra-information) is as difficult as the factorisation of $n$, in the sense of a probabilistic reduction.

There are several problems, depending on whether one wants to compute one key or all keys, and depending on the information given as input.

- **Input of problem 1**: the public key $w_1, \ldots, w_s$. Problem 1: compute all the pseudo-keys $q$
- **Input of problem 2**: the public key $w_1, \ldots, w_s$. Problem 2: compute one pseudo-key $q$
- **Input of problem 3**: the public key $w_1, \ldots, w_s$ and integers $r_1 < \cdots < r_{s-1}$, a range $[a, b]$. Problem 3: compute all pseudo-keys $q$ such that the rests of the divisions $w_i = qv_i + \epsilon_i$, satisfy $\epsilon_i = r_i$ for $0 < i < s$ and $\epsilon_i \in [a, b]$.
- **Input of problem 4**: the public key $w_1, \ldots, w_s$ and integers $r_1 < \cdots < r_{s-1}$, a range $[a, b]$. Problem 4: compute one pseudo-key $q$ such that the rests of the divisions $w_i = qv_i + \epsilon_i$, satisfy $\epsilon_i = r_i$ for $0 < i < s$ and $\epsilon_i \in [a, b]$.

Obviously, it is more difficult to find all the keys than to find one key, and the problem is easier when more information is given as input, as long as the definition of “more difficult” is sensible (polynomial time reduction, probabilistic polynomial time reduction ...). In particular, if $\eta > 0$, then problem 1 is more difficult than problem 2, and problem 1 is more difficult than problem 3. However, when the pseudo-key is unique, then problem 1 is problem 2 and the easiest problem in the list is Problem 4. The previous section explained why the pseudo-key is unique for many cryptosystems. Thus the security of the system relies on the difficulty of solving Problem 4. We show that solving Problem 4 is as difficult as factorising a product of two primes.

- **Input of problem 5**: an integer $n$ which is a product of two primes. Problem 5: Find the factors $p, q$ of $n$.

**Theorem 22.** If it is possible to solve Problem 4 in polynomial time (with respect to the length of the input data), then $\forall \eta > 0$, it is possible to solve Problem 5 in polynomial time with a probability of success at least $1 - \eta$.

**Proof.** Let $n$ be an integer. We make a polynomial time probabilistic reduction to Problem 4 to get the factorisation of $n = pq$.

Choose any superincreasing sequence $0 < r_1 < \cdots < r_{s-1}$. First, try to divide $n$ by all elements $q$ with $1 < q \leq 3 \sum_{i=1}^{s-1} r_i$. If this doesn’t succeed, then all the divisors $q$ of $n$ satisfy $q > 3 \sum_{i=1}^{s-1} r_i$.

Let $w_i = n + r_i$ for $1 \leq i \leq s - 1$. Let $r$ be an integer such that $\left(\frac{a}{b}\right)^r < \eta$. Let $w_{s1}, \ldots, w_{sr}$ be integers chosen randomly in the range $[\frac{a}{b}, n]$. With these $r$ numbers, we consider $r$ problems $P_1, \ldots, P_r$.

The problem $P_k$ is Problem 4 with input $w_1, \ldots, w_{s-1}, w_{sk}, r_1, \ldots, r_{s-1}, a = 0, b = [\frac{a}{b}]$.

Let $q$ be a proper divisor of $n = pq$. It satisfies $q > 3 \sum_{i=1}^{s-1} r_i$. Thus, for each $k$, there is a probability $x > \frac{1}{q}$ that $w_{sk}$ mod $q$ satisfies $\sum_{i=1}^{s-1} r_i < w_{sk}$ mod $q < q$. Remark that $(1 - x)^r < (\frac{a}{b})^r < \eta$. Then, with probability at least $(1 - \eta)$, among the $r$ random choices $w_{sk}, \ldots, w_{sr}$ for $w_s$, one of them $w_{sk}$ satisfies $\sum_{i=1}^{s-1} r_i < w_{sk}$ mod $q < q$. We denote by $(*)$ this condition. To conclude, it suffices to show that one can find a factorisation of $n$ in polynomial time when $(*)$ is satisfied.
We thus suppose that one problem $P_k$ in the list $P_1, \ldots, P_r$ satisfies the condition $(\ast)$. Since $r_i < q$, the equality $w_i = q \cdot p + r_i$ is the euclidean division of $w_i$ by $q$ when $0 < i < s$. Since the rest $\epsilon_{sk}$ of the division $w_{sk} = q \cdot w_{sk} / q + \epsilon_{sk}$ satisfies $\epsilon_{sk} > \sum_{i=1}^{s-1} r_i$ and $\epsilon_{sk} < q \leq \frac{5}{2}$, it follows that a proper divisor $q$ of $n$ is a solution to problem $P_k$.

Reciprocally, a solution $q$ of $P_k$ is a divisor of $n$ different from 1 since $w_1 \mod q = r_1$. This divisor of $n$ is not $n$ since the condition $\epsilon_{sk} \in [a, b]$ is not satisfied for $q = n$. Thus a polynomial time algorithm that solves Problem 4 returns a strict divisor $q$ of $n$ when applied to $P_k$. Hence the factorisation of $n$ in polynomial time.

A priori, we don’t know which problem $P_k$ satisfies $(\ast)$ in the list $P_1, \ldots, P_r$. We thus run a multi-threaded algorithm which tries to solve in parallel the problems $P_1, \ldots, P_r$ and which stops as soon as it finds a solution for one problem.

### 4.3 Comparing LLL attacks on $x_0$ and $x_1$

The previous sections have explored the security of the key. It remains to analyse the security of the system with respect to heuristic attacks. As most heuristic attacks of knapsack cryptosystems rely on variants of the LLL algorithm, we analyse the security of the system for LLL-based heuristic attacks.

The knapsack problem is NP-complete and experiments show that the heuristic attacks fail when the encryption is done with a well chosen general key $x_0$. In our system, the encryption is realised with a key $x_1 = qx_0 + \epsilon$ which is a modification of $x_0$, and it could happen that the key $x_1$ is less secure than $x_0$. Thus we look for a security result asserting that the key $x_1$ is as secure as $x_0$ for LLL-attacks.

The key $x_1$ could be weaker than $x_0$ for two reasons:

- the heuristic algorithm used to break the system could perform faster for a message encrypted with $x_1$ than with a message encrypted with $x_0$
- the heuristic could fail for a message encrypted with $x_0$ but could succeed for the same message encrypted using $x_1$.

We fix an algorithm to attack the ciphertexts. To measure the speed of the algorithm, we denote by $n(N)$ the number of steps of the algorithm when the attack is run on the ciphertext $N$. To measure the probability of success of the algorithm, we introduce the symbol $R(N)$ which is the result of the attack ($R(N) = m$ if the attack succeeds and recovers the plain text message $m$, $R(N) = FAILURE$ otherwise). As the algorithm depends on a matrix $M$ chosen randomly in the unit ball $B(1)$, the precise notations are $n_M(N)$ and $R_M(N)$.

The two keys $x_0$ and $x_1$ yield two ciphertexts $N_0$ and $N_1$. The following theorem says that the key $x_1 = qx_0 + \epsilon$ is as secure as $x_0$ both from speed consideration and probability of success of the attack.

Both the numbers of steps $n$ and the returned message $R$ are unchanged when replacing $x_0$ with $x_1$ provided that two conditions are satisfied: the matrix $M$ must live in a dense open subset and $\|\cdot\|$ must be small enough. These two conditions are compatible with the practice: $M$ is chosen randomly and falls with high probability in a dense open subset and $\|\cdot\|$ is small by the very construction of our cryptosystem.

**Theorem 23.** \forall m\forall x_0, there exists a dense open subset $V \subset B(1)$, there exists $\eta > 0$ such that $\forall M \in V, \forall x_1 = qx_0 + \epsilon$ with $|\epsilon| < \eta$:

- $n_M(N_0) = n_M(N_1)$
- $R_M(N_0) = R_M(N_1)$.

The key arguments of our proof are as follows:

- The elements $x_1$ and $x_0$ are close as points of the projective space
- The LLL algorithm can be factorized to give an action on the projective level
The number of steps in the algorithm and the result of the algorithm are functions of the input which are locally constant on a dense open subset. In particular, replacing \( x_0 \) with \( x_1 \) does not change the number of steps and the result when \( x_0 \) and \( x_1 \) are sufficiently close.

Though the algorithm required for the attack is fixed, its precise form is not important. The key point is that it relies on the LLL algorithm and that the additional data \( M \) required to run the algorithm is chosen randomly. Similar theorems can be obtained with other heuristics relying on the LLL algorithm. Thus, besides the precise attack considered, our theorem suggests that replacing the public key \( x_0 \) with \( x_1 \) does not expose our system to LLL-based attacks.

### 4.3.1 The LLL-algorithm

This section shows that the output of the LLL-algorithm depends continuously of the input when the input takes value in a dense open subset.

This is not clear a priori, since the operations performed during the LLL algorithm include non continuous functions (integer parts). We introduce a class of algorithms that we call analytic. The LLL algorithm is an analytic algorithm. Analytic algorithms can include non continuous functions in continuous functions (integer parts). We introduce a class of algorithms that we call analytic. The input takes value in a dense open subset.

Theorem 26. Let \( A : U \rightarrow V \) be the output function associated to an analytic algorithm i.e. for \( D \in U \), the value of \( A(D) \) is the output of an analytic algorithm with input \( D \). Then there exists a dense open subset \( V \subset U \) such that

- \( A : V \rightarrow U \) is analytic.
Proof. We keep the notations of definition 24. In particular, the algorithm starts in state 1 and ends in state 0. A sign function $\epsilon$ of length $\text{length}(\epsilon) = k$ is by definition a function $\epsilon : \{1, \ldots, k\} \mapsto \{+, -\}$. We associate to any sign function of length $k$ a finite sequence $n_0(\epsilon), \ldots, n_k(\epsilon)$ constructed with the integers $i^+$ and $i^-$ of the analytic algorithm. Explicitly $n_0(\epsilon) = 1$, $n_1(\epsilon) = n_0(\epsilon) c^{(1)}$, $\ldots$, $n_k(\epsilon) = n_{k-1}(\epsilon) c^{(k)}$. We use below the notation $n_i$ instead of $n_i(\epsilon)$ to shorten the notation. Let $A_\epsilon : U \rightarrow U$, $A_\epsilon = T_{n_{k-1}} \circ \cdots \circ T_{n_1} \circ T_{n_0}$. Let $g_\epsilon : U \rightarrow \mathbb{R}$, $g_\epsilon = f_{n_k} \circ A_\epsilon$. We define by induction on $k = \text{length}(\epsilon)$ a set $W_\epsilon$ such that

- $W_\epsilon \subset U$ is an open inclusion
- $A_\epsilon : W_\epsilon \rightarrow U$ is analytic.
- $D \in W_\epsilon \Rightarrow$ the successive states $s_0, \ldots, s_k$ of the algorithm $A$ applied with input $D$ are $s_0 = n_0(\epsilon) = 1$, $s_1 = n_1(\epsilon), \ldots, s_k = n_k(\epsilon)$. Moreover, the value of the datum after the algorithm arrives in state $n_k(\epsilon)$ is $A_\epsilon(D)$.
- $\bigcup_{\text{length}(\epsilon) = k} W_\epsilon$ is dense in $U$.

We start the induction with $k = 0$, using the convention that there is a unique function $\epsilon$ defined on a set with $k = 0$ element and that $A_\epsilon = \text{Id}$. Then $W_\epsilon = U$ obviously satisfies the list of required conditions.

Let now $k > 0$. Let $\tau : \{1, \ldots, k-1\} \mapsto \{+, -\}$ be the restriction of $\epsilon$ to $\{1, \ldots, k-1\}$. Let $W_{\tau^+} = W_\epsilon \cap \{D \in U, g_\epsilon(D) > 0\} \cap (A_\epsilon)^{-1}(U_{n_{k-1}})$ where $U_{n_{k-1}}$ is the open subset of $U$ where $T_{n_{k-1}}$ is analytic. Similarly, let $W_{\tau^-} = W_\epsilon \cap \{D \in U, g_\epsilon(D) < 0\} \cap (A_\epsilon)^{-1}(U_{n_{k-1}})$. The disjoint union $W_{\tau^+} \bigcup W_{\tau^-}$ is dense in $W_\epsilon$ since the difference is included in the closed analytic subset $(g_\tau = 0) \cup A^\tau_\tau - (U - U_{n_{k-1}})$.

Let $W_\epsilon = W_{\tau^+}$ if $\epsilon(k) = +$ and $W_\epsilon = W_{\tau^-}$ if $\epsilon(k) = -$. Since $W_{\tau^+} \cup W_{\tau^-}$ is dense in $W_\tau$ and since $\bigcup_{\text{length}(\tau) = k-1} W_\tau$ is dense in $U$ by induction, we obtain the density of $\bigcup_{\text{length}(\epsilon) = k} W_\epsilon$ in $U$.

The other claims of the list are satisfied by construction.

Let $W_k = \bigcup_\epsilon$ of length $k W_\epsilon$. The intersection $V = \cap_{k \geq 0} W_k$ is equal to the disjoint union

$$\bigcup_{k, \epsilon, \text{length}(\epsilon) = k, n_k = 0, n_{k-1} \neq 0} W_\epsilon.$$

The set $V$ is open as a union of open sets, and it is dense in $U$ by Baire’s theorem. On each open subset $W_\epsilon$ appearing in the disjoint union, the algorithm applied to $D$ returns $A_\epsilon(D)$ which is analytic and the number of steps of the algorithm is $\text{length}(\epsilon)$, thus it is constant on each open set of the disjoint union.

Proposition 27. Let $b_1, \ldots, b_n$ be a basis of a lattice $L \subset \mathbb{R}^m$, $m \geq n$. Let $(c_1, \ldots, c_n) = \text{LLL}(b_1, \ldots, b_n)$ be the reduced basis computed by the LLL algorithm. There exists a dense open subset $U \subset (\mathbb{R}^m)^n$ such that

- $U \mapsto (\mathbb{R}^m)^n$, $(b_i) \mapsto (c_i)$ is continuous.
- $U \mapsto \mathbb{N}$, $(b_i) \mapsto$ number of steps of the LLL-algorithm is locally constant.

Proof. Follows from proposition 25 and theorem 26.

Corollary 28. Let $\psi : U \rightarrow \text{SL}_n(\mathbb{Z})$, $(b_1, \ldots, b_n) \mapsto M$ such that $\begin{pmatrix} c_1 & \cdots \\ \cdots & \cdots \\ c_n \end{pmatrix} = M \begin{pmatrix} b_1 \\ \cdots \\ b_n \end{pmatrix}$ is locally constant.

Proof. The map is continuous with values a discrete set.
4.3.2 The heuristic attack

Let \( w_1, \ldots, w_s \in \mathbb{N} \) be a public key. Let \( m \in \{0,1\}^s \) be a plaintext message and \( N = \sum_{i=1}^{s} m_i w_i \) be the associated ciphertext. The following attack is well known.

**Heuristic Attack 1.**
- Choose \( \lambda = 2^{-2s} \min(w_i) \)
- Apply the LLL algorithm to the lattice generated by the rows \( b_i \) of the matrix \( D = \begin{pmatrix} \lambda & 0 & \ldots & 0 & w_1 \\ 0 & \lambda & \ldots & 0 & w_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \lambda & w_s \\ 0 & 0 & 0 & 0 & N \end{pmatrix} \). Any vector \( c_i \) of the reduced basis is a linear combination:
  \[
  c_i = \sum_{j=1}^{s+1} r_{ij} b_j
  \]
- For each vector \( c_i \) of the reduced basis, check if the set \( r_{ij}, j \leq s \) (or \( -r_{ij} \)) is equal to \( m \) (i.e., check if \( r_{ij} = 0 \) or 1, and if \( \sum_{j=1}^{s} r_{ij} w_j = N \))

In the above attack, the precise value of the coefficients of the matrix \( D \) is not important. The precise shape of \( D \) has been chosen to speed-up the computations and simplify the presentation, but is not required by theoretical considerations. The attack could start with any invertible matrix whose first columns contain small numbers and whose last column is close to the last column of \( D \). Thus the following attack is more general and natural.

**Heuristic attack 2.**
- Choose \( \lambda = 2^{-2s} \min(w_i) \)
- Choose coefficients \( m_{ij}, i, j \leq s + 1 \) with \( |m_{ij}| \leq 1 \). Let \( M = (m_{ij}) \) be the corresponding matrix.
- Let \( X = \begin{pmatrix} 0 & 0 & \ldots & 0 & w_1 \\ 0 & 0 & \ldots & 0 & w_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 & w_s \\ 0 & 0 & 0 & 0 & N \end{pmatrix} \). Apply the LLL algorithm to the lattice generated by the rows \( b_i \) of the matrix
  \[
  D = X + \lambda M = \begin{pmatrix}
  \lambda m_{11} & \ldots & \lambda m_{1s} & w_1 + \lambda m_{1,s+1} \\
  \lambda m_{21} & \ldots & \lambda m_{2s} & w_2 + \lambda m_{2,s+1} \\
  \ldots & \ldots & \ldots & \ldots \\
  \lambda m_{s1} & \ldots & \lambda m_{ss} & w_s + \lambda m_{s,s+1} \\
  \lambda m_{s+1,1} & \ldots & \lambda m_{s+1,s} & N + \lambda m_{s+1,s+1}
  \end{pmatrix}
  \]
- Any vector \( c_i \) of the reduced basis is a linear combination: \( c_i = \sum_{j=1}^{s+1} r_{ij} b_j \) and the coefficients \( r_{ij} \) can be computed during the LLL algorithm.
- For each vector \( c_i \) of the reduced basis, check if the set \( r_{ij}, j \leq s \) or \( -r_{ij}, j \leq s \) is equal to \( m \).

4.3.3 Proof of the theorem

Consider a plaintext message \( m \). It can be encrypted with the generic key \( x_0 = (v_1, \ldots, v_s) \) or with the key \( x_1 = qx_0 + \epsilon = (w_1, \ldots, w_s) \). The two ciphertexts associated with the keys \( x_0 \) and \( x_1 \) are denoted by \( N_0 \) and \( N_1 \).

We compare below how these two encryptions resist to “Heuristic attack 2” presented above. For this algorithm, we need a random matrix \( M \) in the unit ball \( B(1) \). Recall that we called \( n_M(N) \) the number of steps of the algorithm when the attack is done on the ciphertext \( N \). Similarly, we defined \( R_M(N) \) to be the result of the attack \( (R_M(N) = m \) if the attack recovers the plain text message \( m \) and \( R_M(N) = FAILURE \) otherwise).
Theorem 29. \( \forall m, \forall x_0, \) there exists a dense open subset \( V \subset B(1), \) there exists \( \eta > 0 \) such that \( \forall M \in V, \forall x_1 = qx_0 + \epsilon \) with \( \frac{||\epsilon||}{|q|} < \eta: \)

- \( n_M(N_0) = n_M(N_1) \)
- \( R_M(N_0) = R_M(N_1). \)

Proof. We keep the notations \( X, \lambda, D = X + \lambda M \) introduced in the description of the attack. These data depend on the public key \( x = (u_1). \) We denote by \( X_0, \lambda_0, D_0 \) and \( X_1, \lambda_1, D_1 \) these data for the keys \( x_0 \) and \( x_1. \)

If \( C(\epsilon, q) \) is the matrix defined by \( X_1 = q(X_0 + C(\epsilon, q)), \) then \( C(\epsilon, q) \rightarrow 0 \) when \( \frac{||\epsilon||}{|q|} \rightarrow 0. \)

If \( M \) is a matrix with lines \( b_1, \ldots, b_s, \) and if \( (c_1, \ldots, c_s) = LLL(b_1, \ldots, b_s) \) is the reduced basis computed by the LLL-algorithm, we adopt a matrix notation and we denote by \( LLL(M) \) the matrix with lines \( c_1, \ldots, c_s. \) We denote by \( \psi(M) \) the matrix that gives the base change ie. \( LLL(M) = \psi(M).M. \) Finally, we denote by \( n(M) \) the number of steps to perform the LLL-algorithm on the lines of \( M. \)

According to proposition 27 and corollary 28, there exists a dense open subset \( U \) where LLL is continuous and where \( n \) and \( \psi \) are locally constant.

Let \( V = \frac{U - X_0}{\lambda_0} \cap B(1). \) Thus \( V \) is a dense open subset in \( B(1) \) where the map \( \psi_0 : M \mapsto \psi(D_0(M)) \) is continuous. Moreover, the number of steps of the algorithm which computes \( \psi_0 \) is locally constant on \( V. \)

The analysis of the LLL algorithm given in [6] shows that it is a “projective algorithm” ie, in symbols: if \( \rho \in \mathbb{R}, \) we have \( LLL(\rho M) = \rho LLL(M), \psi(\rho M) = \psi(M) \) and \( n(\rho M) = n(M). \)

By definition of the attack considered, the result \( R_M(N_1) \) of the attack is a function of the coefficients \( r_{ij} \) which appear in the matrix \( \psi(D_1(M)). \) In particular, if \( \psi(D_0(M)) = \psi(D_1(M)), \) then \( R_M(N_0) = R_M(N_1). \)

\( \psi(D_1(M)) = \psi(q(X_0 + C(\epsilon, q)) + \lambda_1 M) = \psi(X_0 + C(\epsilon, q) + \frac{\lambda M}{q}) = \psi(X_0 + \lambda_0(\frac{\lambda M}{q}) + \frac{C(\epsilon, q)}{\lambda_0})) = \psi_0(\frac{\lambda M}{q} + \frac{C(\epsilon, q)}{\lambda_0}). \) When \( \frac{||\epsilon||}{|q|} \rightarrow 0, \) the argument of \( \psi_0 \) tends to \( M. \) Since \( M \) is in the open set of continuity of \( \psi_0, \) and since \( \psi_0 \) is locally constant, \( \psi_0(\frac{\lambda M}{q} + \frac{C(\epsilon, q)}{\lambda_0}) = \psi_0(M) = \psi(D_0(M)) \) if \( \frac{||\epsilon||}{|q|} \) is small enough.

Since \( n \) is locally constant too, one can do a similar reasoning with \( n \) instead of \( \psi \) to show that \( n_M(N_0) = n(D_0(M)) = n(D_1(M)) = n_M(N_1). \)

References

[1] L. Babai. On Lovász’ lattice reduction and the nearest lattice point problem. Combinatorica, 6(1):1–13, 1986.

[2] E. F. Brickell and A. M. Odlyzko. Cryptanalysis: a survey of recent results. In Contemporary cryptology, pages 501–540. IEEE, New York, 1992.

[3] Ernest F. Brickell. Breaking iterated knapsacks. In Advances in cryptology (Santa Barbara, Calif., 1984), volume 196 of Lecture Notes in Comput. Sci., pages 342–358. Springer, Berlin, 1985.

[4] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Advances in cryptology—CRYPTO ’97 (Santa Barbara, CA, 1997), volume 1294 of Lecture Notes in Comput. Sci., pages 112–131. Springer, Berlin, 1997.

[5] Donald E. Knuth. The art of computer programming. Vol. 2: Seminumerical algorithms. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont, 1969.
[6] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, Boca Raton, FL, 1997. With a foreword by Ronald L. Rivest.

[7] Ralph C. Merkle and Martin E. Hellman. Hiding information and signatures in trapdoor knapsacks. In *Secure communications and asymmetric cryptosystems*, volume 69 of AAAS Sel. Sympos. Ser., pages 197–215. Westview, Boulder, CO, 1982.

[8] Phong Q. Nguyen and Jacques Stern. The two faces of lattices in cryptology. In *Cryptography and lattices (Providence, RI, 2001)*, volume 2146 of Lecture Notes in Comput. Sci., pages 146–180. Springer, Berlin, 2001.

[9] A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Cryptography and computational number theory (Boulder, CO, 1989)*, volume 42 of Proc. Sympos. Appl. Math., pages 75–88. Amer. Math. Soc., Providence, RI, 1990.

[10] Adi Shamir. On the cryptocomplexity of knapsack systems. In *Conference Record of the Eleventh Annual ACM Symposium on Theory of Computing (Atlanta, Ga., 1979)*, pages 118–129. ACM, New York, 1979.

[11] Adi Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *23rd annual symposium on foundations of computer science (Chicago, Ill., 1982)*, pages 145–152. IEEE, New York, 1982.

[12] Serge Vaudenay. Cryptanalysis of the Chor-Rivest cryptosystem. In *Advances in cryptology—CRYPTO ’98 (Santa Barbara, CA, 1998)*, volume 1462 of Lecture Notes in Comput. Sci., pages 243–256. Springer, Berlin, 1998.