# Review of the Merkle-Hellman cryptosystem and the LLL attack

Alitcha Alexandre Anzala-Yamajako and Antoine Rojat

Grenoble INP Ensimag
`http://www.ensimag.fr`
June 18, 2009

**Abstract.** In this article, we present a cryptosystem introduced in 1978 by Ralph Merkle and Martin Hellman [MH78] in section 2. This cryptosystem is based on the knapsack problem. The purpose of our article is to decide whether it is a valid alternative to the RSA cryptosystem or not. The RSA cryptosystem is described in section 3. In order to compare those protocols we implement both of them. We compare the performances of those two protocols in section 5. It appears that the knapsack cryptosystem is faster than RSA but regarding the security, it cannot sustain the comparison with RSA. We also describe an attack against the Merkle-Hellman cryptosystem based on lattice reduction in section 4. This attack does not allow to recover the private key of somebody but the attacker is able to decipher any ciphered message.

**Table of Contents**

Review of the Merkle-Hellman cryptosystem and the LLL attack . . . . . . . . . . . . . . . . . . 1

*Alitcha Alexandre Anzala-Yamajako and Antoine Rojat*

1 Introduction . . . . . . 3

2 Knapsack cryptosystems . . . . . . 4

2.1 The knapsack problem . . . . . . 4

2.2 The cryptosystem . . . . . . 5

3 The RSA cryptosystem . . . . . . 5

3.1 The RSA Cryptosystem [Riv78] . . . . . . 5

4 The $L^3$ algorithm [Len82] and attack on the knapsack cryposystem . . . . . . 6

4.1 The algorithm . . . . . . 6

4.2 Attack on the Merkle-Hellman cryptosystem [LO85] . . . . . . 8

5 Comparaison between RSA and knapsack . . . . . . 10

5.1 Encryption . . . . . . 10

5.2 Decryption . . . . . . 10

5.3 Key generation . . . . . . 11

5.4 Security . . . . . . 13

6 Conclusion . . . . . . 15

7 Thanks . . . . . . 15

8 References . . . . . . 16

9 Appendix . . . . . . 17

9.1 Knapsack: best known general algorithm . . . . . . 17

9.2 Merkle-Hellman . . . . . . 17

9.3 Masakatu Morii-Masao Kasahara . . . . . . 17

9.4 RSA-155 . . . . . . 17

## 1 Introduction

In [Dif76], Whitfield Diffie and Martin E. Hellman described the idea of public key cryptography and the concept of trap door one way function. This paper was a landmark as it offered an alternative to symmetric cryptography (when the same key is used for encryption as well as decryption) and the problem of key-exchange. In asymmetric cryptography the key needed to encrypt a plaintext is different from the key used to decrypt it. This allows someone, let's call her Alice, to publish her encryption key. If Bob wants to send a secret message to Alice, he will encrypt the message using Alice's public encryption key and send it to her. Alice can then decrypt and read the message. Assuming she didn't disclose her secret key, she's the only one who can do so. The process is summarized in figure 1. Another upside of asymmetric cryptography is that it

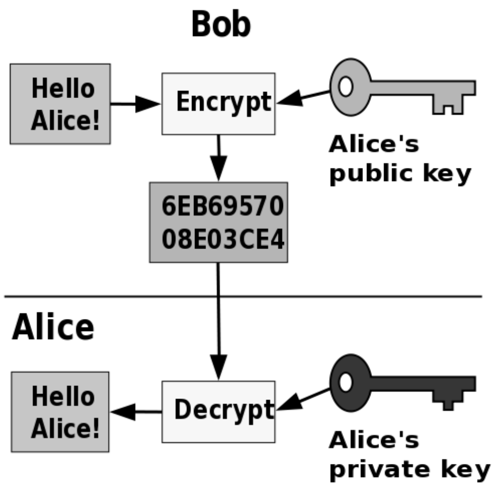


**Fig. 1.** Encryption-decryption mechanism

allows to build protocols for purposes others than only sending secrets such as digital signing (see [Riv78]), authentication or Zero-Knowledge (see [QQQ+90]).

The main underlying principle of public-key cryptography is the notion of trapdoor one-way function: A one-way function is a function that is "easy" to compute but "hard" to invert. More formally:

**Definition 1.** *$f$ is a one-way function if it is computationally feasible (in polynomial time) to get $f(x)$ from $x$, but for which it is computationally infeasible to get $x$ from $f(x)$*

One can see that if $x$ is the message Bob wants to send to Alice, computing $f(x)$ and sending it to Alice is quite useless: Alice (just like anyone else) does not have the computational power to get $x$ from $f(x)$. This is when the notion of trapdoor comes in handy:

**Definition 2.** *A trapdoor one-way function is a one-way function with the special property that with a particular piece of information (the trapdoor) it becomes computationaly feasible to get $x$ from $f(x)$*

Alice keeping this "trapdoor" secret,is ensured to be the only way one that can get $x$ from $f(x)$ in reasonable time.

Following the publication of [Dif76] researchers concentrated their efforts in finding candidate trapdoor one-way function. Several of these candidates relied on number theoritic problems, higher residuosity (see [Nac98]), discrete logarithm (see [Gam85]) and , integer factorisation (see [Riv78]) among others. Another way for building asymmetric cryptographic schemes is to choose a computationally hard problem, use an easy instance as the private key, a hard instance as the public key, the trapdoor one-way being the inversible transformation used to transform the easy instance into the hard instance (see [MH78]).

## 2 Knapsack cryptosystems

### 2.1 The knapsack problem

There are two kinds of knapsack problems:

– Additive: *Given a finite sequence of integers:* $(a_i)_{i=1..n}$ *and* $S \in \mathbb{N}$, *is it possible to find* $(\epsilon_i)_{i=1..n} \in \{0,1\}^n$ *such that* $S = \sum_{i=1}^{n} \epsilon_i a_i$ *?*

– Multiplicative: *Given a finite sequence of integers:* $(a_i)_{i=1..n}$ *and* $P \in \mathbb{N}$, *is it possible to find* $(\epsilon_i)_{i=1..n} \in \{0,1\}^n$ *such that* $P = \prod_{i=1}^{n} a_i^{\epsilon_i}$ *?*

**Theorem 1.** *The knapsack problem is* $\mathcal{NP}$*-hard*

Even tough the knapsack problem is said to be very hard, there are instances very easy to solve. Before describing such instances, we need to introduce the following definition:

**Definition 3.** $(a_j)_{j \in J}$ *is a super increasing sequence when*

$$\forall k \in J, \left( \sum_{j \in J,\ j<k} a_j \right) < a_k$$

**Proposition 1.** *Solving an additive knapsack problem when the coefficients form a super increasing sequence is feasible in linear complexity.*

*Proof.* Let $(a_i)_{i=1..n}$ be a super increasing sequence and $S = \sum_{i=1}^{n} \epsilon_i a_i$ be an instance of an additive knapsack. We need to determine the value of $(\epsilon_i)$.

Consider $\epsilon_n$: $a_n < S \Rightarrow \epsilon_n = 1$ indeed by hypothesis, $a_n > \sum_{i=1}^{n-1} a_i$.

Then, we can consider the following knapsack instance

$$(a_i)_{i=1..n-1} \text{ with } S' = S - a_n \text{ if } \epsilon_n = 1 \text{ or } S' = S \text{ if } \epsilon_n = 0.$$

This gives us a linear algorithm to solve this kind of problems.

**Proposition 2.** *Solving a multiplicative knapsack problem when the coefficients are all coprime is feasible in linear complexity.*

*Proof.* Let $(a_i)_{i=1..n}$ be a sequence such that $\forall (i,j),\ i \neq j,\ gcd(a_i, a_j) = 1$ and $P = \prod_{i=1}^{n} a_i^{\epsilon_i}$ be an instance of a multiplicative knapsack.
$\forall k \in \{1,..,n\}$, define $p_k = P\ mod\ [a_k]$.
If $\epsilon_k = 1, \exists K \in \mathbb{Z} P = a_k \times K$ this implies $p_k = 0$.
This gives an easy and fast way for recovering the $\epsilon_i$.
Reciprocally if $p_k = 0$ it implies that that $P\ mod\ [a_k] = 0$ and by definition $a_k | P$. So we can deduce that $\epsilon_k$

## 2.2 The cryptosystem

The exchanged messages will be composed by bits. Let $b = b_0 ... b_n$ be a message.

In order to encrypt a message, the sender needs to get the public key of the receiver: $(pub_i^{rec})_{i=1..n}$. Then he encrypts the message i.e. he computes

$$C = \sum_{i=1}^{n} b_i pub_i^{rec} \ (or \ \prod_{i=1}^{n} (pub_i^{rec})^{b_i}).$$

After that, he sends the message to the receiver.

The receiver gets $C$ and he computes $C^0 = f^{-1}(C)$ where $f$ is the one-to-one function such that $\forall i,\ pub_i^{rec} = f(priv_i^{rec})$. He is now able to decrypt $C^0$ and recover the message $b$.

We have implemented two different knapsack cryptosystems:

The first one has been proposed by Ralph Merkle and Martin Hellman [MH78] in 1978. The keys are generated as follows:
Compute a super increasing sequence $(priv_i)_{i=1..n}$ which will be the private key.
Let $M$ be such that $M > \sum_{i=1}^{n} priv_i$.
Let $W \in \mathbb{N}$ be such that $gcd(W, M) = 1$.
The public key will be $(pub_i)_{i=1..n}$ where $\forall i,\ pub_i = f_W(priv_i)$ and $f_W \colon \mathbb{Z}/M\mathbb{Z} \longrightarrow \mathbb{Z}/M\mathbb{Z}$ such that $\forall x \in \mathbb{Z}/M\mathbb{Z}\ f(x) = Wx \ mod\ [M]$. (see 9.2 for details)

The second one has been proposed by Masakatu Morii and Masao Kasahara in 1988. The private key is a multiplicative knapsack $(priv_i)_{i=1..n}$ where $\forall i,\ \gcd(priv_i, priv_{j \neq i}) = 1$.
Let $p \in \mathbb{N}$ such that $\forall i,\ \gcd(priv_i, p) = 1$ and $p > \prod_{i=1}^{n} priv_i$.
Let $e$ be such that $\gcd(e, p-1) = 1$ and compute $d = e^{-1} mod[p-1]$.
The public key is the multiplicative knapsack $(pub_i)_{i=1..n}$ where $\forall i,\ pub_i = priv_i^e[p]$ and $p$.
(See 9.3 for more details)

# 3 The RSA cryptosystem

## 3.1 The RSA Cryptosystem [Riv78]

1. We consider $n, p, q \in \mathbb{N}$ such that p and q are prime numbers and $n = pq$
2. We also consider $e, d \in \mathbb{N}$ such that $ed \equiv (\phi(n))$
3. The public key pk is (e,n) and the private key sk is (d,n)
4. $\forall x \in \mathbb{Z}/n\mathbb{Z}, \text{encrypt}_{pk}(x) = x^e \ mod[n]$
5. $\forall y \in \mathbb{Z}/n\mathbb{Z}, \text{decrypt}_{sk}(y) = x^d \ mod[n]$

**Fig. 2.** The RSA cryptosystem

To benchmark the knapsack cryptosystem we had to choose another cryptosystem to act as reference. The choice was fairly obvious: RSA (for cryptographers Rivest, Shamir and Adleman) is a well studied protocol and probably the most widely used one since the publication of [Riv78]. Rivest *et al.* paper was written following the publication of the forementioned[Dif76]. The idea behind the RSA scheme is that given an sufficiently well choosen integer $n$, it is computationally infeasible to factor it into the product of its prime factors and that finding $\phi(n)$ is as hard as factoring $n$. In figure 2 are shown the method for encryption and decryption for the RSA cryptosystem.

## 4 The $L^3$ algorithm [Len82] and attack on the knapsack cryposystem

### 4.1 The algorithm

Throughout this section, when we consider some vectors $(b_i)_{i=1..n}$, $(b_i^*)_{i=1..n}$ will always denote the result of the orthogonalization via the Gram-Schmidt process of $(b_i)_{i=1..n}$. And $\|.\|$ will always denotes the euclidian norme.

**Definition 4.** *A lattice $L$ is a discrete sub group of $\mathbb{R}^n$.*
*More precisely, $L$ is a lattice of dimension $m$ if $L \neq \varnothing$ and there exists $m$ vectors $b_1, \ \dots \ , b_n$ lineary independant over $\mathbb{R}$ such that $L = \oplus_{i=1}^{m} \mathbb{Z} b_i$*

**Definition 5.** *Let $(b_i)_{i=1..n}$ be the basis of a lattice. $(b_i)_{i=1..n}$ is said to be weakly-reduced when*

$$\forall i, j \ i \neq j \ , |\frac{(b_j|b_i^*)}{\|b_i^*\|^2}| < \frac{1}{2}$$

**Definition 6.** *We define $\llcorner x \urcorner$ as the closest integer from $x$:*

$$\llcorner x \urcorner = sgn(x) \lfloor |x| + 0.5 \rfloor.$$

*where $\lfloor . \rfloor$ is the floor function.*

*Remark 1.* This is the closest integer away from zero.

Here is an algorithm producing weakly-reduced basis:

```
Input: $(b_i)_{i=1..n}$ a basis of a lattice
Output: $(b_i)_{i=1..n}$ weakly-reduced
for $k = 2$ to $n$ do
  for $j = k-1$ to $1$ do
    $b_k = b_k - \llcorner \frac{(b_k|b_i^*)}{\|b_i^*\|^2} \urcorner b_j$
  end for
end for
```

**Fig. 3.** Weak reduction of a basis $(b_i)_{i=1..n}$

There are several other definitions of the closest integer such as:

1. $\llcorner n \urcorner = \lfloor n + 0.5 \rfloor$
2. $\llcorner n \urcorner$ is the closest even integer

In order to implement the weak reduction we had to choose a definition to handle the case of perfect halves. We choose the one we proposed because the algorithm on figure 4 is deterministic (for a given instance it will always output the same result). But with the following input $< 1, 20, 80 >, S = 21$, and the definition $\lfloor n + 0.5 \rfloor$ of the closest integer, the LLL algorithm was not able to find the solution and there is no reason in this context to round up to closest even integer. While choosing the previous we gave we found the solution. We verified with the implementation proposed by Damien Sthele [Ste05] and he found the same result as ours.

**Definition 7.** *A basis $(b_i)_{i=1..n}$ is said to be LLL-reduced with a factor $\delta$ when*

1. *it is weakly reduced*
2. *it verifies the Lovasz condition:*

$$\forall k,\ \|b_{k+1}^*\|^2 \geq (\delta - (\frac{(b_{k+1}|b_k^*)}{\|b_k^*\|^2})^2)\|b_k^*\|^2$$

In [Len82] A.K Lenstra, H.W Lenstra and L.Lovasz showed that if a basis is LLL-reduced with a factor $\frac{3}{4}$, their algorithm (see figure 4) will produce a "small" basis: a basis composed by small vectors. In order to implements the LLL algorithm we use a course teached by Nicolas Brisebarre ([Bri03]).

```
**Input**: $B = (b_i)_{i=1..n}$ a basis of a lattice
**Output**: $(\widehat{b_i})_{i=1..n}$ LLL-reduced with a factor $\delta$
{*At step k, we consider that* $B = (b_1, \dots, b_k)$ *is LLL-reduced.*}
$k \leftarrow 1$
while $k \leq n$ do
  $B \leftarrow B \bigcup b_{k+1}$
  WeakReduction(B)
  if $\|b_{k+1}^*\|^2 \geq (\delta - (\frac{(b_{k+1}|b_k^*)}{\|b_k^*\|^2})^2)\|b_k^*\|^2$ then
    $k \leftarrow k+1$
  else
    swap $b_k$ and $b_{k+1}$
    $k \leftarrow k-1$
  end if
end while
```

**Fig. 4.** The $L^3$ algorithm

The complexity of this algorithm was first given by Lenstra *et al.* in [Len82]. We have $O(n^6 \ln^3 B)$ where n is the size of the basis and B is such that $\forall i,\ \|b_i\| < B$. But faster implementations using floating-point calculus are know to work under $O(n^5(n + \ln B)\ln B)$ (see [Ste05]).

### 4.2 Attack on the Merkle-Hellman cryptosystem [LO85]

Altought the subset sum is $\mathcal{NP}$-hard the litterature shows that the instance used in the Merkle-Hellman cryptosystem can be broken in polynomial time by reduction to a lattice and the use of the LLL-base reduction algorithm. This method raises two questions:

1. How does one choose the lattice in question ?
2. Why is the output of LLL algorithm the solution to the knapsack problem ?

We denote Alice's public key $(a_1, a_2, \ldots, a_n)$, Bob plaintext m ($m_i, i = 1, \ldots, n$ being the i-th bit of m) and $M = \sum_{i=1}^{n} a_i m_i$. The lattice vector basis being referred to as $b_1, \ldots, b_{n+1}$ (see figure 5) the output of the LLL algorithm will be denoted $\widehat{b_1}, \ldots, \widehat{b_{n+1}}$. We also need to define the notion of density:

**Definition 8.** *The density $d$ of a subset sum problem is the quantity* $\frac{n}{\log_2(\max\limits_{i=1,\ldots,n} a_i)}$.

Let's consider $L$ the lattice spanned by the vectors in figure 5: $L = \{\sum_{i=1}^{n} z_i b_i + b_{n+1}, z_i \in \mathbb{Z}\}$. Notice that the vector solution expanded $\hat{m} = (m_1, \ldots, m_n, 0)$ is in $L$ because of the integer linear relation $\hat{m} = \sum_{i=1}^{n+1} m_i b_i$. We can also notice that $\hat{m}$ is a fairly short vector of $L$, this statement can be justified by the fact that in cryptographic settings we have $a_i >> n \geq \|\hat{m}\|^2$. Intuitively, if one wants to find a short vector in L, one better try to nullify the last coordinate because its contribution to the norm is likely to be far bigger than the rest of the vector.

$$\begin{aligned}
b_1 &= (1, 0, 0, \ldots, 0, -Kpub_1^{Alice}) \\
b_2 &= (0, 1, 0, \ldots, 0, -Kpub_2^{Alice}) \\
&\vdots \\
b_n &= (0, 0, 0, \ldots, 1, -Kpub_n^{Alice}) \\
b_{n+1} &= (0, 0, 0, \ldots, 0, KS)
\end{aligned}$$

**Fig. 5.** The subset sum lattice

In fact [LO85] Lagarias and Odlyzko stated that for all problems of density $d < 0.645$ the vector $\hat{m}$ solution has a very high probability of being the shortest non-zero vector of the lattice spanned by the vectors in 5 (see [CJL$^+$91] for a formal proof). This way we manage to reduce the subset sum problem to the one of finding the shortest vector of L. However the LLL algorithm doesn't actually find **the** shortest vector of a given lattice but we know that the length of the shortest (and first) vector of its output is bounded by the relation:

$$\|\hat{b_1}\|^2 < 2^{n-1}\lambda^2 \tag{1}$$

where $\lambda$ is the norm of the actual shortest vector of the lattice. (see [Len82], [JS94]). The constant $K$ is a trick to "fool" the LLL into giving us the shortest vector: we know that the shortest vector of the lattice is the solution to the knapsack problem. Being a binary vector, we have the following relation about the shortest vector $\lambda^2 < n$. Suppose $K > n2^{n-1}$ and $x \in L$ such that the last coordinate of $x$ isn't zero we can then derive that $\|x\|^2 > K > n2^{n-1}$ and $x$ cannot be the shortest vector of the output LLL without contradicting 1. We are now sure that the output of LLL is in fact the shortest vector of L which is $\hat{m}$, the vector solution to the subset sum problem. Note that the bound given by the equation 1 is a worst-case bound, in average , the vectors found by LLL are much better. This is why we allowed ourselves to set K to 1.

The last point we need to make to show the efficiency of attacks on the Merkle-Hellman protocol using LLL is about density. We know that by construction (see [MH78]) $a_i$'s are greater than $2^{2n}$ and therefore $d < \frac{n}{\ln_2 2^{2n}} = \frac{1}{2} < 0.645$

After breaking the Merkle Hellman cryptosystem, cryptographers went on to build knew knapsack-based cryptosystems of higher density, however Coster *et al.* showed in [CJL$^+$91] that the bound could be increased up to 0.9408 simply by using a different matrix.

## 5 Comparaison between RSA and knapsack

*Remark 2.* In order to measure the time spent by our programs, we use the **getrusage** POSIX tool.

### 5.1 Encryption

In our implementation of the RSA cryptosystem, the value of $e$ is the constant $2^{16}+1$. The encipher function computes *ciphered message* $=$ (*clear message*)$^e$ *mod* $[n]$. The complexity of this operation is $O(log_2(e))t(n)$ where $t(n)$ is the complexity of a modular multiplication of size $n$. That is why the encryption time for the RSA protocol is linear with respect to the size of the key.

For the Merkle-Hellman cryptosystem, the encryption is essentially the summation of size $2^n$ which is a linear operation. That is why we observe a linear dependency between the time and the size of the key used for encryption.

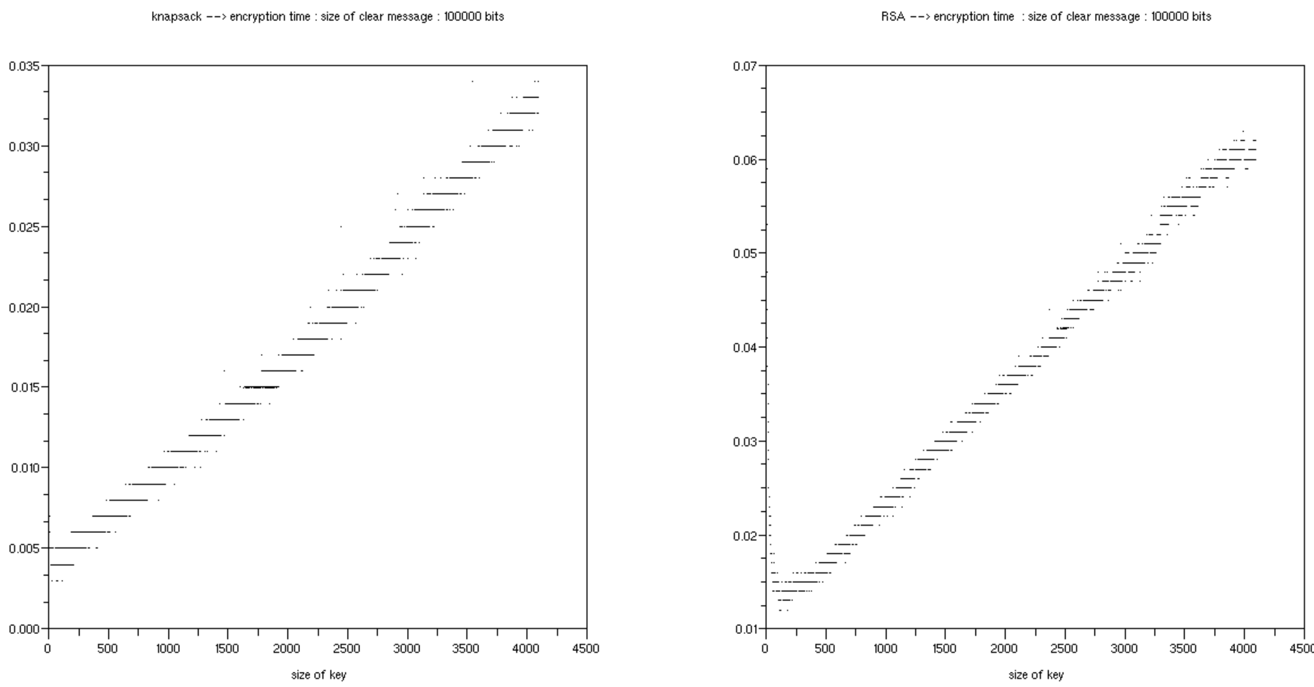


**Fig. 6.** Encryption time

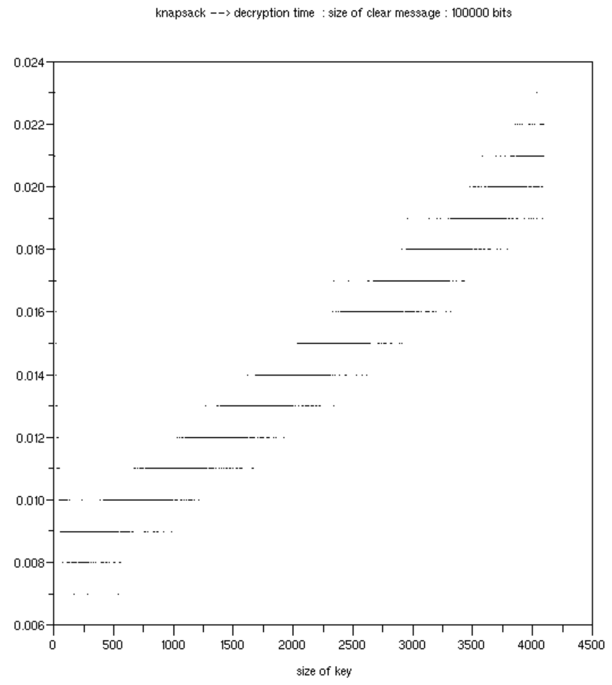
### 5.2 Decryption

The decryption in the Merkle-Hellman consists of

1. a modular multiplication
2. some substractions

The time needed for the modular multiplication is small regarding the number of substraction that is why we observe a straight line on figure 7.

For RSA, we compute a modular exponentiation. The order of the exponent's size is approximately $n$. Indeed as the parameter $e$ is constant, the size of the parameter $d$ will grow with $n$. That is why the time is quadratic at the beginning until the gmp library [gmp] change the algorithm used to compute the multiplication: at the begining it uses a naive algorithm (complexity $O(n^2)$) and then it uses Karatsuba algorithm (complexity $O(n^{\log_2(3)})$). This explains the changes on the curve of figure 7.



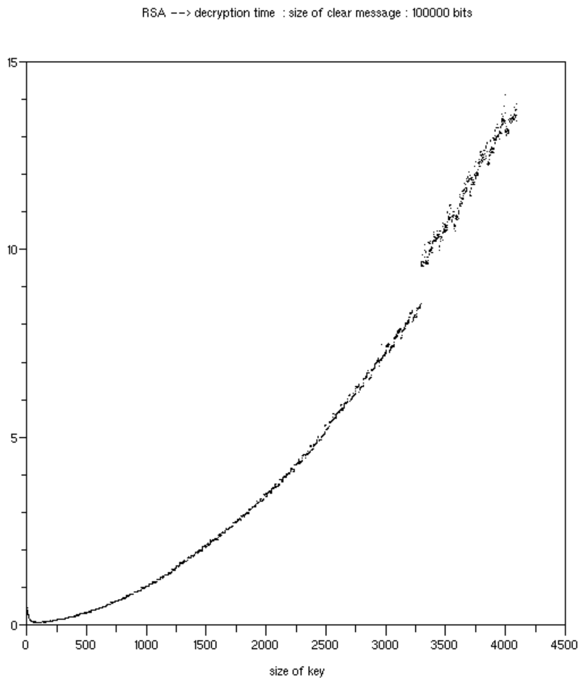


**Fig. 7.** Decryption time

## 5.3 Key generation

The generation of key in our implementation of the RSA protocol is done as follows:

1. compute two random numbers of size $\frac{n}{2}$
2. find to prime numbers $p_1$ and $p_2$ of size $\frac{n}{2}$ using the mpz_nextprime primitive on the two previous number.
3. compute $d = e^{-1}\ mod\ [(p_1 - 1)(p_2 - 1)]$ using the mpz_invert

The operation of finding the next prime is the longest. And as we use a probabilistic algorithm, the time needed for finding a prime number is also "random". That is why we observe scatter plot on figure 8.

For the generation of the Merkle-Hellman protocol keys we use the following algorithm:

```
SUM = 0
offset = 0
for $i = 1$ to $n$ do
    R = random number of size $n + offset$
    if R $\leq$ SUM then
        R=2R
        offset++
    end if
    privKey[i]=R
    SUM=SUM+R
    offset++
end for
```

That is why we observe the quadratic aspect of the curve on figure 8.

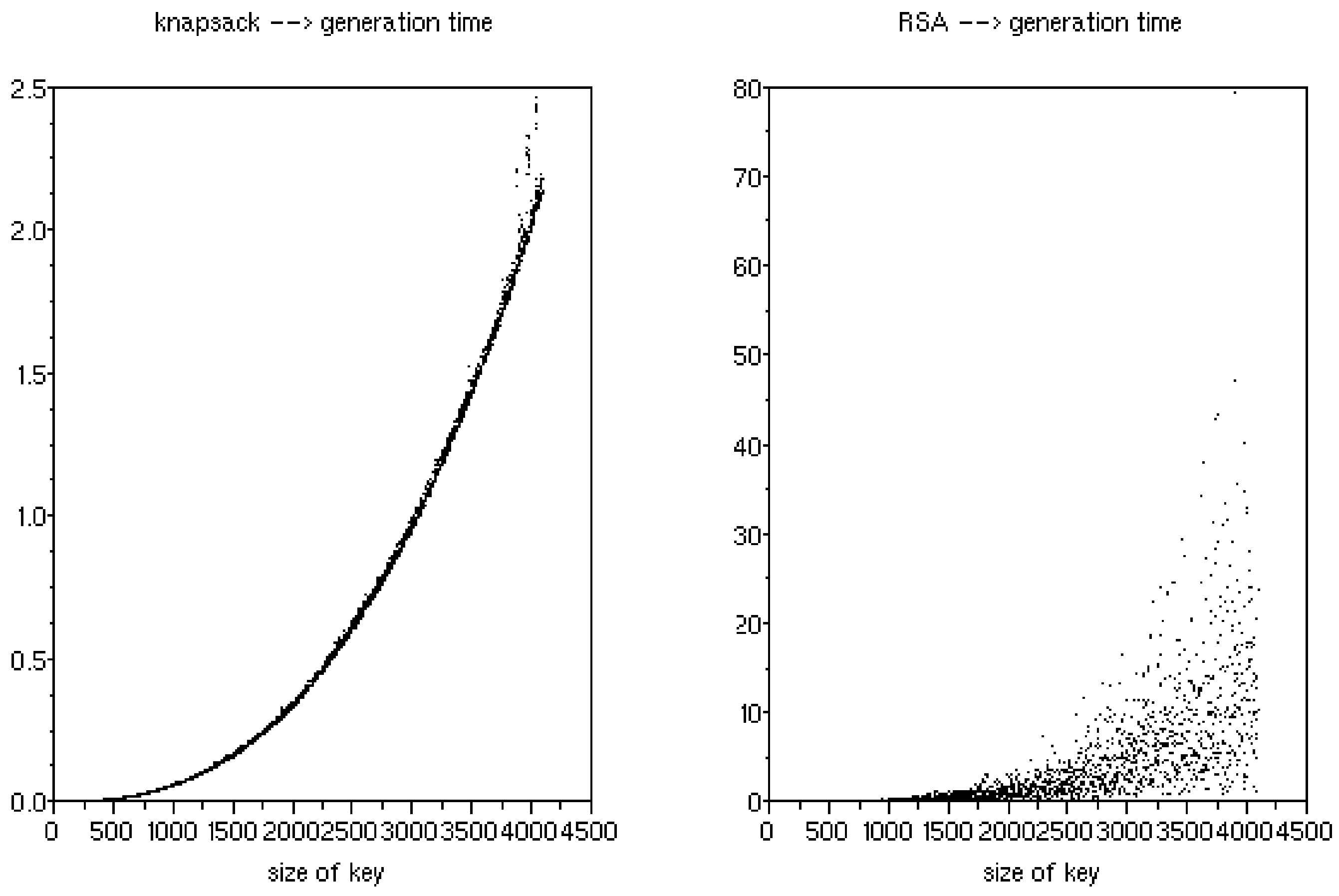


**Fig. 8.** Generation time

### 5.4 Security

The problem of breaking RSA is equivalent to factoring numbers. The best known algorithm for integer factorisation is *NFS* (see [Pom96]), its complexity is sub-exponential (see [Lan]). In 1999, a team of computer scientist factorized RSA-155 (see 9.4): a RSA number composed of 155 digits approximatively 512 bits (see [CLR+00]). The unit used to measure the number of operations needed to run the algorithm is *mips*: *a million operation per seconds during one year*. It took 8000 *mips* to factorize RSA-155.

The complexity of *NFS* is $O(e^{1.9223+o(1)(\ln(n))^{\frac{1}{3}}(\ln\ln(n))^{\frac{2}{3}}})$. As it is the best known algorithm for integer factorisation, In order to compare the security of RSA and the Merkle-Hellman cryptosystem, we used the result of [CLR+00] then extrapolated it. This was the only way to estimate the value of the constant of the theoretical complexity. We found a constant $\approx 0.0000059$. We also used the theoretical complexity of the $L^3$ algorithm (see 4.1) to plot the number of operations needed for breaking RSA or the Merkle-Hellman cryptosystem.

By construction (see 5.3), in the Merkle-Hellman protocol, the size of the public key parameters is $\approx 2^{3n}$. We can consider that the majorant is $2^{3n}+1$. The memory space used for storing a Merkle-Hellman key of size $n$ can be computed as follows: we have $n$ coefficients of size $2n$ bits in average for the private key and we have $n$ coefficients of size $3n$ bits. So the memory space is $O(n^2)$ bits. We can also simplify the complexity of $L^3$ as follows: $O(n^5(n + \ln B)\ln B) = O(n^7)$ because of the way we construct the coefficients. We used fplll (see [Ste05]) in order to determine more precisely the complexity: for reducing a basis it took 9 seconds on a single processor of 1.67 $GHz$ that is why we consider $O(n^7) \approx 0.0001503n^{\frac{1}{7}}$.

| size of RSA key | corresponding Knapsack key | |
|---|---|---|
| | number of coefficients | memory space |
| 128 | 34 | 5,6 Kb |
| 256 | 149 | 108,4 Kb |
| 512 | 1077 | 5,5 Mb |
| 1024 | 14756 | 1,01 Gb |
| 2048 | 463490 | 980 Gb |
| 4096 | 42906924 | 8.17 $10^6$ Gb |

**Fig. 9.** equivalence between RSA and Merkle-Hellman

The values we obtain are not precise enought and it is strange that RSA-256 or RSA-128 seems less secure that the knapsack cryptosystem. But after the values seem more realistic. Anyway the general idea of the previous array was to show that the Merkle-Hellman cryptosystem is not usable in practice.

Furthermore the knapsack cryptosystem has been totally broken by Adi Shamir in 1982 (see [Sha82]). Then several variants of the knapsack cryptosystems have been proposed over the years and almost all of them have been broken (for a global review see [Lai]).

## 6 Conclusion

The cryptosystems based on the knapsack problem are computationally easier than the RSA protocol when it comes to encryption and decryption which are nothing more than sums and at most one modular multiplication. Furthermore the key generation stage is simpler: it only requires the generation of a super-increasing sequence, one modular inversion and $n$ modular multiplications which is in no way comparable to the finding of two large strong primes. This is a substential advantage over RSA because some of the hardware cryptography is implemented on have very limited computing power and/or are very little powered. The knapsack cryptosystem, as it turned out, did not sustainted the comparison to the security offered by RSA. The equivalent to the widely used 1024 bit long RSA key is a 463490 bit long which would occupy almost a terabyte (1000 gigabytes) of storage.

For all the reason we mentionned above, when Merkle and Hellman proposed their protocol in [MH78], it appeared to be a good alternative to the RSA protocol. However their cryptosystem was broken by Shamir in 1982 (see [Sha82]). After that Merkle and Hellman published an amelioration of their cryptosystems which involved an iterative process for the public key generation. This was designed to resist Shamir's attack. However Brickell in [Bri84] proposed a method for breaking even this new cryptosystem. Actually various attacks, most of them based on the lattice reduction algorithm $L^3$ we described in section 4, broke almost all variants of cryptosystems based on the knapsack problem. That is why those kind of cryptosystems have never been used in practice and why researchers progressively lost interest.

However the Masakatu Morii-Masao Kasahara cryptosystem described in [Lai] which use a easy multiplicative knapsack as a private key and transform it into the private key using the RSA theorem. Laurent Evain in [Eva08] described a knapsack-based cryptosystem built on NP-hard instance.

Even thought knapsack-based cryptosystems were never as succesful as RSA the idea to use easy instance of "hard" problems as private key, hard instances as public key and a one-to-one secret transformation as the one trapdoor one way function lived on. The NTRU public key cryptosystem use the NP-hard in average problem of finding the shortest vector in a lattice [HPS98].

## 7 Thanks

We want to thanks M. Laurent Fousse for his useful remarks on our work.

## 8 References

## References

[Bri84] Brickell. Solving low density knapsacks. In *Advances in Cryptology, Proceedings of Crypto '83*, pages 25–37, 1984.

[Bri03] Nicolas Brisebarre. L'algorithme LLL et certaines de ses applications, May 2003.

[CJL$^+$91] Coster, Joux, Lamacchia, Odlyzko, Schnorr, and Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2:111–128, 1991.

[CLR$^+$00] Stefania Cavallar, Walter M. Lioen, H. J. J. Te Riele, B. Dodson, A. K. Lenstra, P. L. Montgomery, B. Murphy, B. Murphy Et Al, Mathematisch Centrum (smc, The Dutch Foundation, Stefania Cavallar, Walter Lioen, Herman Te Riele, Bruce Dodson, Arjen K. Lenstra, Peter L. Montgomery, Brian Murphy, Karen Aardal, and Jeff Gilchrist. Factorization of a 512-bit RSA modulus, 2000.

[Dif76] Hellman Diffie. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[Eva08] Laurent Evain. Knapsack cryptosystems built on np-hard instance. *CoRR*, abs/0803.0845, 2008.

[Gam85] El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[gmp] Gmp, a free library for arbitrary precision arithmetic. Technical report.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In *Lecture Notes in Computer Science*, pages 267–288. Springer-Verlag, 1998.

[JS94] Joux and Stern. Lattice reduction: a toolbox for the cryptanalyst. *Journal of Cryptology*, 11:161–185, 1994.

[Lai] Ming Kin Lai. Knapsack cryptosystems : The past and the future.

[Lan] Eric Landquist. The number field sieve factoring algorithm.

[Len82] Lovász Lenstra, Lenstra. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[LO85] Lagarias and Odlyzko. Solving low-density subset sum problems. *J. ACM*, 32(1):229–246, 1985.

[MH78] Merkle and Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions On Information Theory*, 24:525–530, 1978.

[Nac98] Stern Naccache. A new public key cryptosystem based on higher residues. In *ACM Conference on Computer and Communications Security*, pages 59–66, 1998.

[Pom96] Carl Pomerance. A tale of two sieves. *Notices Amer. Math. Soc*, 43:1473–1485, 1996.

[QQQ$^+$90] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis C. Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, Soazig Guillou, and Thomas A. Berson. How to explain zero-knowledge protocols to your children. In *CRYPTO '89: Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, pages 628–631, London, UK, 1990. Springer-Verlag.

[Riv78] Adleman Rivest, Shamir. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Sha82] Adi Shamir. A polynomial time algorithm for breaking the basic merkle-hellman cryptosystem. pages 145–152, 1982.

[Ste05] Damien Stehlé. *Algorithmique de la réduction de réseaux et application à la recherche de pires cas pour l'arrondi de fonctions mathématiques.* PhD thesis, Université Nancy I, 12 2005.

## 9 Appendix

### 9.1 Knapsack: best known general algorithm

The best known algorithm for solving the knapsack without any condition on its input problem runs with $O_c(n2^{\frac{n}{2}})$ and also $O_m(2^{\frac{n}{2}})$ where

- $O_c$ will denote the complexity regarding the number of operations performed for solving a problem
- $O_m$ will denote the storage space needed for solving a problem

### 9.2 Merkle-Hellman

**Proposition 3.** *If $gcd(W, M) = 1$ then $f_W : \mathbb{Z}/M\mathbb{Z} \longrightarrow \mathbb{Z}/M\mathbb{Z}$ such that $\forall x \in \mathbb{Z}/M\mathbb{Z}$ $f(x) = Wx \ mod \ [M]$ is a one-to-one function and $f_W^{-1} = f_{W^{-1} \ mod \ [M]}$.*

Let's assume Bob wants to send a message $b = b_1...b_n$ to Alice: Bob computes $C = \sum_{i=1}^{n} b_i pub_i^{Alice}$ and send $C$ to Alice.
When Alice receives the message, she computes

$$f_{W^{-1} \ mod \ [M]}(C) = f_{W^{-1} \ mod \ [M]}(\sum_{i=1}^{n} b_i pub_i^{Alice}) = \sum_{i=1}^{n} b_i f_{W^{-1} \ mod \ [M]}(priv_i^{Alice}) = \sum_{i=1}^{n} b_i priv_i^{Alice}$$

She is now able to decrypt the message $f_{W^{-1} \ mod \ [M]}(C)$ and recover $b$.

### 9.3 Masakatu Morii-Masao Kasahara

**Theorem 2.** *Consider $n \in \mathbb{P}$.
Let $e, d$ be such that $de = 1 \ mod \ [n-1]$ and $e, d < n$.
$\forall x \in \mathbb{Z}/n\mathbb{Z}$, $x^{ed} = x \ mod \ [n]$*

*Proof.* $ed = 1 \ mod \ [p-1]$ so $\exists K \in \mathbb{Z}$ such that $ed = K(p-1)+1$
$x^{ed} = x^{K(p-1)+1} = x \ (x^{(p-1)})^K$ using Fermat's theorem, we know that
$x^{(p-1)} = 1 \ mod \ [p]$. So we obtain $x^{ed} = 1 \ mod \ [p]$ .

Again let's assume that Bob wants to send a message $b = b_0...b_n$ to Alice. He computes $C = \prod_{i=1}^{n}(pub_i^{Alice})^{b_i} \ mod \ [p]$ and sends $C$ to Alice.
When she receives the message, Alice will compute:

$$C^d \ mod \ [p] = \prod_{i=1}^{n}(pub_i^{Alice})^{db_i} \ mod \ [p] = \prod_{i=1}^{n}(priv_i^{Alice})^{b_i} \ mod \ [p]$$

She is now able to decrypt the message $\prod_{i=1}^{n}(priv_i^{Alice})^{b_i} \ mod \ [p]$ and recover $b$.

### 9.4 RSA-155

RSA-155 $=$
10941738641570527421809707322040357612003732945449205990913842131476349984288934784717
997257891267332497625752899781833797076537244027146743531593354333897
$=$ 102639592829741105772054196573991675900716567808038066803341933521790711307779
$\times$106603488380168454820927220360012878679207958575989291522270608237193062808643