Lattice Reduction: a Toolbox for the Cryptanalyst

Antoine Joux
DGA/CELAR
Bruz, France

Jacques Stern
Laboratoire d'Informatique
Ecole Normale Supérieure, Paris

May 18, 1994

Abstract

In recent years, methods based on lattice reduction have been used repeatedly for the cryptanalytic attack of various systems. Even if they do not rest on highly sophisticated theories, these methods may look a bit intricate to the practically oriented cryptographers, both from the mathematical and the algorithmic point of view. The aim of the present paper is to explain what can be achieved by lattice reduction algorithms, even without understanding of the actual mechanisms involved. Two examples are given, one of them being the attack devised by the second named author against Knuth's truncated linear congruential generator, which has been announced a few years ago and appears here for the first time in journal version.

1 Introduction

1.1 Historical background

A lattice is a discrete subgroup of R^n or equivalently the set L

$$\lambda_1 b_1 + \cdots + \lambda_p b_p$$

of all integral linear combination of a given set of n-dimensional vectors $b_1, \cdots, b_p$. If these vectors are independent then $(b_1, \cdots, b_p)$ is said to be a basis of L and p is its dimension.

From the mathematical point of view, the history of lattice reduction goes back to the theory of quadratic forms developed by Lagrange, Gauss, Hermite, Korkine-Zolotareff and others (see [Lag73, Gau01, Her50, KZ73]) and to Minkovski's geometry of numbers ([Min10]). With the advent of algorithmic number theory, the subject had a revival around 1980, when Lovász found a polynomial-time algorithm that computes a so-called *reduced basis* of a lattice. Actually, a reduction algorithm of the same flavor had already been included in Lenstra's work on integer programming (cf. [Len83], circulated around 1979) and the lattice reduction algorithm reached a final form in the paper [LLL82] of Lenstra, Lenstra and Lovász, from which the name *LLL algorithm* comes. Further refinements of the LLL algorithm were proposed by Schnorr ([Sch87, Sch88]).

The relevance of those algorithms to cryptography was immediately understood: in April 1982, Shamir ([Sha82]) found a polynomial time algorithm breaking the Merkle-Hellman public key cryptosystem ([MH78]) based on the knapsack problem, that had been basically the unique alternative to RSA. Shamir used Lenstra's integer programming algorithm but, the same year, Adleman ([Adl83]) extended Shamir's work by treating the cryptographic problem as a lattice problem rather than a linear programming problem. Further improvements of these methods were obtained by Lagarias and Odlyzko ([LO83]), by Brickell ([Bri85]) and, more recently by Coster, La Macchia, Odlyzko, Schnorr and the authors ([CJL+92]).

Lattice reduction has also been applied successfully in various other cryptographic contexts: against a version of Blum's protocol for exchanging secrets ([FHK+88]), against truncated linear congruential generators ([FHK+88, Ste87]), against cryptosystems based on rational numbers ([ST90]) or modular knapsacks ([JS91, CJS91]). Despite the available literature, papers are still submitted (and sometimes published) that describe cryptographic protocols that can be broken, via lattice reduction techniques, almost by inspection. This fact, which may be due to the apparent technicality of the subject, drove us to try to write a paper that explains the power of lattice reduction in cryptography, without requiring any understanding of the actual mechanisms involved in the algorithms. Of course, this was also an opportunity to publish, in final form, results that had been announced in [Ste87] and [GJ94].

1.2 Functional description of lattice reduction algorithms

As was already mentioned, a lattice L consists of all integral linear combination

$$\lambda_1 b_1 + \cdots + \lambda_p b_p$$

of a given set of n-dimensional vectors $b_1, \cdots, b_p$. From the algorithmic point of view, we are interested into the case where all b_i's have integer coordinates. In this case, the lattice L can be represented by a very simple data structure by considering the matrix B_L whose columns are the coordinates of the vectors $b_1, \cdots, b_p$. Lattice reduction algorithms perform the following very simple operations:

i) exchanging two columns of B_L

ii) adding up to a given column an integer multiple of another one
iii) deleting zero columns
What is not simple is the precise way the sequence of above transformations is chosen. We will simply mention that the algorithm tries:
i) to have the shortest columns ahead
ii) to make the columns mutually "as orthogonal as possible"

Ideally, we would like to come out with the first column of the matrix consisting of the coordinates of a shortest non zero vector of L and with "almost" orthogonal columns. Unfortunately, this is not the case and we note that no efficient algorithm is known for finding the shortest non-zero vector of L. This is actually a fundamental problem which lies at the heart of the solution of many problems in number theory. Still, from the output of the algorithm, it is possible to build a vector whose length does not exceed the length of a shortest vector by more than a given multiplicative constant, depending on the dimension of L as well as on the variant of the algorithm used. It turns out that this is enough for many applications.

1.3 Proved Performances

Let L be a lattice generated by of a set of n-dimensional vectors Let B_L be the associated matrix. Denote by B the value of the matrix obtained as an output of the LLL algorithm and by $b_1, \cdots, b_q$ its column vectors. Finally let λ_1 be the length of a shortest non zero vector of L (in the usual euclidean sense). The following essentially comes from [LLL82]:

Fact 1 *i)* $b_1, \cdots, b_q$ *is a basis of* L
ii) $|b_1| \leq 2^{(q-1)/2} \times \lambda$
iii)$|b_1| \leq 2^{q(q-1)/2} \times (\Delta(L))^{1/q}$

In the above, $\Delta(L)$ denotes the determinant of L, that is the (euclidean) volume of the q-dimensional parallelepiped enclosed by $b_1, \cdots, b_q$. In case the lattice is full dimensional (which means $n = q$), this volume is the absolute value of the determinant of B or of any other basis generating L. In the general case, $\Delta(L)$ can also be computed by a simple formula which we omit. Condition iii) means that the length of b_1 is not too far from what it is in the "ideal" case, corresponding to a basis consisting of mutually orthogonal vectors of equal length.

For the cryptanalyst, the heuristic meaning of fact 1 is that, if he only needs a short enough vector of a lattice, LLL will do the job. Similarly, if he knows that the (unknown) shortest vector is much smaller than the other elements of the lattice or much smaller that the value $(\Delta(L))^{1/q}$, then LLL will presumably disclose it. There is a generalization of fact 1 which is sometimes useful: it is related with the so-called *successive minima* of the lattice: the ith minimum is the smallest positive value λ_i such that there exist i linearly independent elements of the lattice in the ball of radius λ_i centered at the origin.

Fact 2 $|b_i| \leq 2^{(q-1)/2} \times \lambda_i$

For the cryptanalyst, this fact amounts to saying that if i (unknown) linearly dependent vectors of the lattice are very small, the sublattice they span will be disclosed by LLL.

Actually, the LLL algorithm consists of a family of different algorithms depending on a constant γ, $\frac{1}{4} < \gamma < 1$. The case that is described above corresponds to the value $\gamma = 3/4$ and if another value of γ is chosen, then the powers of two appearing in the above facts must be replaced by the same powers of $4/(4\gamma - 1)$.

In [Sch87], Schnorr proposes a whole hierarchy of lattice reduction algorithms, which are extensions of the LLL algorithms and which he calls blockwise Korkine-Zolotareff reductions (BKZ). What changes here is the strategy to perform the operations on the matrix B_L. The extended strategy involves a search on sublattices generated by blocks of columns of the original matrix. When the size of the blocks grows, the performances of the algorithm get better and better, achieving the situation obtained from fact 1 by replacing powers of two by powers of any constant $\sigma > 1$.

1.4 Actual performances

In all the applications, experiments show that LLL behaves much more nicely than should be expected in view of the theory. Especially the worst-case constant K_q, which appears in fact 1 as $2^{\frac{q-1}{2}}$ seems to be much smaller in practical terms.

1.5 Implementations

The running time of the LLL algorithm is polynomial in the dimension n of the space, the dimension q of the lattice and the size of the matrix B_L. More precisely, if m is the maximal number of bits in the coefficients of the original matrix B_L, then the running time of the standard LLL algorithm is $O(nq^5m^3)$. Albeit polynomial, this is not negligible and does not allow any efficient implementation. Following a suggestion made by Odlyzko and independently by Schnorr, actual implementations of LLL reductions, including the one used by the authors, substitute floating arithmetic to the rational arithmetic required by the original specification of the algorithms. Nevertheless, this cannot be done in a naive way as the strategy may occasionally be misled by floating point errors and enter a loop. Fortunately, these occasional errors can be spotted and corrected, at a minor cost in terms of computing time, by performing the "exact computation".

Of course, the running time of the LLL algorithm also depends on the value of the constant γ adopted and several heuristics can be helpful such as computing first with a moderate value of γ and ultimately with a value close to 1. Similarly, BKZ reductions have a worse computing time than LLL. Finally, the computing time also depends on the kind of problems one addresses.

2 Generic problems that fall under the scope of lattice reduction

2.1 Direct use of lattice reduction

By "direct use", we mean practical applications where the lattice comes from the data in a natural way. This was the situation for the original attack against the Merkle-hellman cryptosystems. We simply mention the method, firstly because it does not involve any specific analysis besides the results stated in section 1 and also because it did not prove useful in more recent work.

2.2 Dependence relations with coefficients of moderate size

The search of linear dependence relations with small coefficients in a family of numbers or vectors is probably the source of most frequent uses of lattice reduction. This general class of applications can be further divided into two cases, ordinary relations and modular relations. We deal with the ordinary case here, and with the modular case in the next subsection. It should be noted that we do not cover here the problems of solving knapsacks and finding minimal polynomials. We will consider them as specific problems and they will receive a detailed treatment in subsequent subsections. Before we turn to practical matters, we briefly discuss the question from a mathematical point of view.

2.2.1 Some combinatorial landmarks

For the cryptographer, the search for linear dependence relations with coefficients of moderate size can occur either because he is looking for specific objects, whose existence is known (trapdoors etc.) or for generic objects he might use for a further analysis. The following combinatorial lemma ensures the existence of such generic relations.

Lemma 1 *Assume $V_1, \ldots, V_n$ is a family of vectors with integer coefficients in the t-dimensional space, with $t < n$. Let M denote an upper bound for the absolute values of all coefficients of the various V_is. There exists an integer relation*

$$\sum_{i=1}^{n} \lambda_i V_i = 0$$

such that $\max|\lambda_i| \leq B$*, where* B *is given by*

$$\log B = t\frac{\log M + \log n + 1}{n-t}$$

Remark: In the whole paper log will denote the base 2 logarithm.

Proof: Consider all possible linear combinations

$$\sum_{i=1}^{n} \mu_i V_i$$

with $0 \leq \mu_i < B$. An easy counting argument shows that the numbers of such relations is exactly B^n and that the resulting vectors have all coordinates (strictly) bounded by nBM. Since there are less than $(2nBM)^t$ such vectors, two distinct relations have to compute the same value, as soon as

$$(2BM)^t \leq B^n$$

which amounts to the given relation

$$\log B = t\frac{\log M + \log n + 1}{n-t}$$

This gives

$$\sum_{i=1}^{n} \mu_i V_i = \sum_{i=1}^{n} \mu_i' V_i$$

with $0 \leq \mu_i < B$ and $0 \leq \mu_i' < B$. The result follows by difference.

Remark: It is obvious that the shortest dependence relation (say w.r.t. the euclidean length) can be much shorter than what is claimed in the above lemma. We give a heuristic argument to show that our estimate is probably pessimistic. If we consider that vectors computed by the formula

$$\sum_{i=1}^{n} \mu_i V_i$$

behave like random elements in the t-dimensional cube of size $(2BM)$ then, by the birthday paradox, we can see that a collision happens with constant probability as soon as

$$\log B = \frac{t}{2}\frac{\log M + \log n + 1}{n - t/2}$$

2.2.2 Practical point of view

Given a family of integer vectors (or numbers) $V_1, \ldots, V_n$, let us describe the principle of dependence relations search. We construct the lattice given by the columns of the following matrix:

$$\begin{pmatrix} KV_1 & KV_2 & \cdots & KV_n \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

where K is a well chosen constant.

We distinguish two cases: either we are looking for exact relations or else for approximate relations. In the first case, K should be large enough to ensure that the first vector of the reduced basis has zero components in its upper part corresponding to the first t coordinates, where t is the dimension of the V_is. More accurately, in view of fact 1, K should be larger than the size of the expected linear relation multiplied by a a safety coefficient $2^{n/2}$. Thus, LLL will discover short vectors whose upper part is guaranteed to be zero, and these vectors clearly correspond to linear dependencies with small coefficients. The coefficients appear as coordinates of rank $t+1, \cdots, t+n$ of the output vector.

In the case of approximate relations, we can choose $K = 1$. Output vectors will be short but there is no reason why the upper part should be zero. This clearly corresponds to approximate dependencies with small coefficients.

2.3 Modular relations

In the previous section, we explained how to disclose linear relations with moderate coefficients between integer vectors. We now discuss the case of mod$m$ numbers. The basic problem is how one can force lattice reduction to deal with modular relations. The answer is very simple and consists in adding to the lattice basis a few columns that ensure modular reduction as shown in the following matrix:

$$\begin{pmatrix} KV_1 & KV_2 & \cdots & KV_n & KmI \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

where I is a t-dimensional identity matrix, with t the dimension of the V_is. It is clear that the added columns force reduction of numbers modulo m.

From a practical point of view, we need to foretell whether or not the resulting lattice will disclose the expected dependence relation. To discuss this question and provide heuristics, we remark that the lattice includes short vectors that are not related whatsoever with the existence of any linear relation. These vectors can be obtained by multiplying any of the first n vectors in the initial basis by a factor m and then by reducing upper part mod$m$, with the help of the extra columns. We obtain a vector whose components are all zero except one whose value is m. Applying the above construction to all V_is, we get a family of n vectors of size m that are mutually orthogonal. Experiments show that, if m is too small, this family appears in sequence as the first output vectors of a reduced basis, and thus masks any useful information about linear relations. However, if the (euclidean) size of the expected relation is smaller than m, we can reasonably hope that the reduction algorithm will find it. Using fact 1 above, it is possible to give conditions that will ascertain the above heuristic observations. Still, this is not very useful in practice and we will not pursue the matter. We close the section, by observing that, in the special case where $m = 2$, we cannot expect to disclose relations with more than 3 ones. Moreover, such relations can usually be found faster by exhaustive search. This explains why lattice reduction algorithms are not successful for attacking binary problems, such as finding the shortest codeword in a linear code, or the solution of a SAT problem.

2.4 Knapsack problems

Solving knapsack problems is a subcase of searching linear relations between given numbers. However, we treat it specifically, not only because of its historical importance in cryptography but also because it is is more involved than the general case, due to the fact the the expected relations have coefficients in $\{0, 1\}$. In cryptographic scenarios, we know that such a relation exists between the given elements of the knapsack $a_1, \ldots, a_n$ and the target sum $s = \sum_{i=1}^{n} \epsilon_i a_i$. Moreover we know that the euclidean size of this relation is $\sqrt{\alpha n}$, where α is the proportion of ones in the relations. α may or may not be known to the cryptanalyst but, in most practical examples it is a part of the cryptographic system itself. Furthermore α is an important parameter when trying to analyze the performances of lattice-based attacks against knapsack problems. However, discussing the influence of α is somewhat technical and is not within the scope of this article. We refer the interested reader to [CJL+92] or [Jou93]. In the sequel, we will consider the most natural case and set $\alpha = 1/2$.

Another parameter that is quite important in knapsack problems is the density of the knapsack:

$$d = \frac{n}{\log_2(\max_i a_i)}.$$

This parameter is the ratio between the number of elements in the knapsack and the number of bits in each element. This parameter determines the size of short vectors in the lattice other than the $\{0, 1\}$ solution vector. It was shown in [LO83], that, when the density is low, then the shortest vector gives the solution to the knapsack problem. If we use the lattice that was described above, and if we assume that shortest lattice-vectors can be efficiently computed (even if this is not totally accurate), then low density means $d < 0.6463$. In recent work ([CJL+92]), this condition was improved to $d < 0.9408$. In order to reach that bound, either one of the following lattices can be used

$$\begin{pmatrix} Ka_1 & Ka_2 & \cdots & Ka_n & -Ks \\ n+1 & -1 & \cdots & -1 & -1 \\ -1 & n+1 & \cdots & -1 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -1 & -1 & \cdots & n+1 & -1 \\ -1 & -1 & \cdots & -1 & n+1 \end{pmatrix} \quad \begin{pmatrix} Ka_1 & Ka_2 & \cdots & Ka_n & Ks \\ 1 & 0 & \cdots & 0 & 1/2 \\ 0 & 1 & \cdots & 0 & 1/2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1/2 \end{pmatrix}.$$

Before we close this section, let us warn the reader on the meaning of the low-density attacks. The inequality $d < 0.9408$, provides a *provable* guarantee that, from a shortest vector for a lattice computed from the problem one can, with high probability, solve the original knapsack problem. This kind of result is sometimes described in the setting of "oracles": it states that, if one is granted access to a lattice reduction oracle, i.e. to a function that returns the shortest vector of a lattice (at no computation cost), then one can solve the low-density knapsack problem. It does not mean at all that one cannot successfully attack knapsack problems with a higher density: it only means that such attacks will not follow from a theorem but only from various heuristics. From a practical point of view, it does not make much difference.

2.5 Minimal polynomials

Finding the minimal polynomial of a real algebraic number x of degree d corresponds to searching a linear dependency between $1, x, x^2, \ldots, x^d$. Since we are working with integer lattices, we choose a large integer K and we try to find an approximate relation between the closest integers to $K, Kx, Kx^2, \ldots, Kx^d$. More precisely, we reduce the following lattice:

$$\begin{pmatrix} K & \lfloor Kx \rceil & \lfloor Kx^2 \rceil & \cdots & \lfloor Kx^d \rceil \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

The first vector of the reduced lattice can be written as:

$$\begin{pmatrix} \epsilon \\ a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix}.$$

Since we wish to interpret $a_0, \ldots, a_d$ as the coefficients of the minimal polynomial of x, i.e. we want to conclude that $a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d = 0$. The most important parameters here are K and d. If d is smaller than the degree of the minimal polynomial of x then this technique cannot succeed. Likewise, if K is too small, then it cannot succeed either. To see this, assume for example that x is between 0 and 1 and apply lemma 1: this yields a linear combination of the elements on the first row of the above matrix with coefficients bounded above by B, where B satisfies:

$$\log B = \frac{\log K + \log d + 1}{n-1}$$

If K is small, this relation is much more likely to appear as an output to lattice reduction algorithms than the one corresponding to the minimal polynomial. Taking into account the heuristic remarks following lemma 1, it is safe to have $K \geq (\max |a_i|)^{2d}$. Hence, K should be much larger than the expected size of the coefficients of the minimal polynomial. If d is not exactly known, for example if we only know an upper bound on the degree of the minimal polynomial of x, then the following trick can be applied: take the first two or three vectors appearing in the output reduced lattice, transform them into polynomials and compute their gcd. If K was large enough the minimal polynomial of x is usually obtained.

It is very important to know that the procedure we just described can give positive results, i.e. it can find a minimal polynomial, but cannot give negative one.

3 Two examples

We now turn to two specific examples. As mentioned in the introduction, these were chosen among the authors' contributions in the area and appear here for the first time in journal version.

3.1 Cryptanalysis of Knuth's truncated linear congruential generators

In this section, we discuss the predictability of the sequence given by outputting a constant proportion of the leading bits of the numbers produced by a linear congruential generator. As is known, linear congruential generators (LCG) are a quite popular tool to produce pseudo-random sequences. The LCG works as follows: a modulus m is chosen as well as a multiplier a, relatively prime to m, and an increment b. Then, from a given seed x_0, one can generate the sequence (x_i), defined by

$$x_{i+1} = (a x_i + b) \bmod m$$

Knuth's book ([Knu69]) contains a thorough discussion of these generators.

In case all the bits of the successive x_is are announced, the sequence becomes exactly predictable even if the modulus, the multiplier and the increment are not known. This is a result of J. Boyar (see [Plu82]). The journal version [Boy89], which appeared after [Ste87], extends the initial method to the case where a small portion of the lower bits are discarded.

The idea of outputting the leading bits of each of the x_is in order to increase the resistance of the LCG goes back to Knuth ([Knu80]). Thus, one can output, for example, half of the bits or a smaller proportion. The predictability of the resulting sequence has been investigated by Frieze, Hastad, Kannan, Lagarias and Shamir ([FHK+88]). They showed that, provided both the modulus m and the multiplier a are known, the sequence becomes completely predictable once the leading bits corresponding to the first few x_is have been announced. Actually, their algorithm may fail on a set of exceptional multipliers but the proportion of integers $\bmod m$ in this set is shown to be as small as $O(m^{-\epsilon})$, for some given positive constant $\epsilon < 1$. Of course, the parameter ϵ is connected with the number of observed outputs: the more observations are available, the more the algorithm is reliable. We refer to [FHK+88] for exact statements. We note that the technique applies equally to the case where any fixed proportion α of the bits is announced. The mathematical analysis becomes more intricate and the proof is only carried through for specific values of m: square-free or "almost square-free" numbers. Again, we refer to [FHK+88] and we observe that, for practical purposes, the mild theoretical restrictions in the proofs are not too relevant.

Our results cover the case where m and a are unknown parameters. In view of the above, our sole task is to disclose these values. We will describe a polynomial-time algorithm that performs this task. This algorithm includes two steps: in the first step, our algorithm produces a polynomial $P(x)$ of degree $O(\sqrt{\log m})$, with integral coefficients, such that

$$P(a) = 0 \bmod m$$

This part of the algorithm is proved, using results from [FHK+88] and similar assumptions on m and a. In the second step, we start with a sequence of such polynomials and we propose an algorithm that provably outputs a multiple $\tilde{m}$ of m. Based on a heuristic analysis, we then make highly plausible that $\tilde{m}$ quickly decreases to m when the number of polynomials in the sequence increases. This is confirmed by experiments. Finally, we show how to compute a once the correct value of m has been recovered.

3.1.1 First step of the algorithm

In order to describe both the algorithm and the underlying analysis, we need some notations. We let ν be the number of bits of the modulus m. If we output a proportion α of bits, we can write

$$x_i = 2^{\beta\nu} y_i + z_i$$

where $\beta = 1 - \alpha$, y_i consists of the leading bits of x_i and z_i of the trailing bits. Our algorithm is more accurately described as a sequence of different algorithms depending on a parameter t. We let V_i be the element of Z^t defined by

$$V_i = \begin{pmatrix} y_{i+1} - y_i \\ y_{i+2} - y_{i+1} \\ \vdots \\ y_{i+t} - y_{i+t-1} \end{pmatrix}$$

Applying the techniques of section 2.2, we can find an linear relation

$$\sum_{i=1}^{n} \lambda_i V_i = 0$$

whose coefficients are moderate integers. More precisely, it follows from lemma 1 that such a relation exists with $|\lambda_i| \leq B$ with

$$\log B = t\frac{\log(2^{\alpha\nu}) + \log n + 1}{n-t} = t\frac{\alpha\nu + \log n + 1}{n-t}$$

Considering the multiplicative loss $2^{(n-1)/2}$ coming from fact 1, it follows that, if we use LLL algorithm, the euclidean length $|\lambda|$ of the output relation will satisfy

$$|\lambda| \leq \sqrt{n}2^{(n-1)/2}B$$

We now consider the (unknown) vectors W_i defined by

$$W_i = \begin{pmatrix} x_{i+1} - x_i \\ x_{i+2} - x_{i+1} \\ \vdots \\ x_{i+t} - x_{i+t-1} \end{pmatrix}$$

and we let

$$U = \sum_{i=1}^{n} \lambda_i W_i$$

We note that each coordinate of $W_i - U_i$ is the difference $z_{i+1} - z_i$ of two integers between 0 and $2^{\beta\nu}$ hence is $\leq 2^{\beta\nu}$. From this, using the Schwarz inequality, we get that $|U| = |\sum_{i=1}^{n} V_i - W_i|$ is bounded above by $M = \sqrt{tn}2^{\beta\nu}2^{(n-1)/2}B$. Taking logarithms, we have

$$\log M = \frac{\log t + \log n}{2} + \beta\nu + \frac{n-1}{2} + t\frac{\alpha\nu + \log n + 1}{n-t}$$

We balance the two terms with largest contribution besides $\beta\nu$ by letting $n \simeq \sqrt{2\alpha t\nu}$. We get (for fixed t):

$$\log M = \beta\nu + \sqrt{2\alpha t\nu} + o(\sqrt{\nu})$$

Since ν is basically $\log m$, we finally obtain: $|U| = O(m^{\beta+\delta})$, for any $\delta > 0$.

We will now proceed into showing that U is zero. We give a heuristic argument and refer the interested reader to appendix A of this paper where we provide a mathematical proof, based on results from [FHK+88]. We note that

$$x_{i+j+1} - x_{i+j} = a^j(x_{i+1} - x_i) \bmod m$$

From this it follows that all vectors W_i belong to the lattice $L(a)$ generated by the columns of the following matrix

$$\begin{pmatrix} 1 & 0 & & \cdots & 0 \\ a & m & & \cdots & 0 \\ a^2 & 0 & m & \cdots & 0 \\ \vdots & & & \vdots & \\ a^{t-1} & 0 & 0 & \cdots & m \end{pmatrix}$$

It is easily seen that the determinant of this lattice is m^{t-1}, hence the expected size of short vectors is around $m^{\frac{t-1}{t}}$. Since U belongs to $L(a)$ and is of size $O(m^{\beta+\delta})$, for any $\delta > 0$, U is unusually short as soon as $\beta < \frac{t-1}{t}$, which means $\alpha > \frac{1}{t}$. Such a vector has to be zero.

Now that we know that $U = 0$, we notice that

$$\sum_{i=1}^{n} \lambda_i W_i = (x_1 - x_0) \sum_{i=1}^{n} \lambda_i a^i \bmod m$$

As soon as $x_1 - x_0$ is prime to m, we get that the polynomial $P(x) = \sum \lambda_i x^i$ vanishes at a modulo m. This is precisely what we wanted from the first step. Again we ignore bad luck: in appendix B, we prove that exceptional values for $x_1 - x_0$ appear with negligible probability.

3.1.2 Second step of the algorithm

If we apply part 1 of our algorithm several times, we come out with a sequence

$$P_1, \cdots, P_r$$

of polynomials of degree n, each of these vanishing at a modulo m. Now, if we identify polynomials of degree n with elements of Z^{n+1}, we see that the polynomials that vanish at a modulo m form a lattice L generated by the sequence

$$Q_i(x) = x^i - a^i \qquad 1 \le i \le n$$

and by the constant polynomial m. This lattice is generated by the columns of the following matrix

$$\begin{pmatrix} m & -a & -a^2 & \cdots & -a^n \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

The determinant of the lattice is m. Now, if the P_is generate the lattice, then one can apply lattice reduction, output a basis of the lattice and compute the determinant. Based on experiments, we claim that such an algorithm actually discloses m. Unfortunately, we cannot prove this fact mathematically, but, as already observed all along the paper, the lack of proof is only a minor nuisance for the cryptanalyst. In place, we offer heuristic arguments that should convince the reader that

- the dimension on the subspace spanned by the P_is very quickly increases to $n+1$
- once a full dimensional lattice has been reached, the determinant of the lattice generated by the P_is, which is, *a priori*, a multiple $\tilde{m}$ of m, very quickly decreases to m.

We first justify the first statement: recall that the output of the first part of the algorithm provides an *actual* relation

$$\sum \lambda_i(x_{i+1} - x_i)$$

(as opposed to a relation modulo m). If the successive relations found did not span the entire space, the vectors W_is would live in a proper subspace and hence, there would exist at least one non-trivial linear relation between their components which holds truly (and not only modulo m). Such a relation would, in turn, provide a fixed linear recurrence relation satisfied by the sequence (x_i). But the behavior of the sequences defined by linear recurrence relations is well known: except under exceptional circumstances they quickly tend over to zero or infinity and therefore cannot consist of integers mod$m$.

We now turn to the second statement. Remember that the coefficients of each output polynomial P_i are bounded above by B, with

$$\log B = \sqrt{\frac{\alpha t \log m}{2}} + o(\sqrt{\log mu})$$

By Hadamard inequality, it follows that the determinant $\tilde{m}$ of the lattice spanned by $n+1$ linearly independent P_is is at most $(\sqrt{n+1}B)^{n+1}$. As n is equivalent to $\sqrt{2\alpha t \log m}$, this bound is $2^{\alpha t \log m + o(\log m)}$, i.e. $m^{\alpha t + o(1)}$. Now, if we assume that each extra polynomial P_i is somehow uncorrelated to the previous ones, then, with probability $\tilde{m}/m$, it is not a member of the lattice generated by these previous polynomials, hence, when added, it decreases the determinant of the resulting lattice by some multiplicative factor. Thus, it should not take long to reach m. In appendix C, using the heuristic hypothesis that the output polynomials are random element of L with coefficients bounded by B, we bound the number of polynomials needed by $O(\alpha t \log m)$.

We finally say a word on recovering a from m. We twist the lattice L by multiplying all coefficients of degree ≥ 2 by a large constant K and we apply lattice reduction. If K is large enough (say $K \geq m2^{n/2}$), then, by fact 2, it follows that the sublattice of L generated by polynomials m and $x - a$ will be disclosed: this is because the second minimum of L is of size m any any polynomial of the lattice with degree ≥ 2 exceeds this size by a factor $\geq 2^{n/2}$. By linear algebra, one can find an element $A(x)$ of the sublattice with leading coefficient one; then a is exactly $A(0) \bmod m$.

We close the section by various remarks.

Remarks.

1) As was noticed repeatedly in the paper, reduction algorithms behave much nicely than what is expected from the worst-case proved bounds. This has practical consequences that may speed up our attack: for example, one can undertake this attack with less observed data than what is suggested from the formula $n \simeq \sqrt{2\alpha t \log m}$. Also, one can try to keep more than one relation from the output of the lattice reduction used in step one.

2) The case where the trailing bits of the successive x_is are announced in place of the leading bits can be attacked by a similar technique, at least if m is odd.

3) Paper [FHK+88] includes an idea which can be used to adapt our techniques to the case where m is prime and a window of successive bits of the x_is is announced. We find the details too technical to be included in the present paper.

3.2 Cryptanalysis of Damgård's hash function

In [Dam89], Damgård proposed to base a hash function on a knapsack compression function using 256 (non modular) numbers a_i of size 120 bits. His idea was to divide the message to be hashed into blocks of 128 bits, and to apply to following process:

- Start with a fixed initial value on 128 bits. Appending the first 128-bit block of the message, one gets a block B of 256 bits.

- (Compression phase) Compute the knapsack transform of these 256 bits, i.e., starting from zero, add up all a_is whose index corresponds to the position of a one bit of B. The resulting number can be encoded using 128 bits.

- Append the next block, to get 256 bits and iterate the compression phase.

In order to find a collision for this hash function, it is clearly enough to find two different 128-bit blocks that, when appended to the initial value, yield the same hash value. This clearly corresponds to finding a collision in a knapsack transform based on 128 numbers of 120 bits. In the sequel, we study how collisions in such a knapsack transform can be found using lattice reduction, and we show that it is feasible to build collisions for Damgård's hash function. A completely different kind of attack against this hash function has already appeared in the work of P. Camion and J. Patarin ([CP91]). Still, it has never been implemented, and besides, it could only find collisions for the compression function rather than for the hash function itself. In contrast with this approach, our attack runs on a computer and actually outputs collision for the size of the parameters suggested by Damgård.

Unfortunately, our attack cannot be proved, even in the lattice oracle setting described in section 2.4. Nevertheless, for a slightly weaker notion of a collision, which we call pseudo-collision, a correct mathematical analysis can be carried through. A *pseudo-collision* for Damgård's hash function consists of two messages whose hash value coincide except for the 8 leading bits. The practical significance of pseudo-collisions is obvious since pseudo-collisions have a non negligible chance of being actual collisions.

3.2.1 The basic strategy

In this section, we associate a lattice to any given knapsack-based compression-function in such a way that collisions correspond to short vectors. Before describing the reduction, we make our definitions and notations a bit more precise: we fix a sequence of integers, $a = a_1, \ldots, a_n$. The knapsack-compression function S_a, which we simply denote by S, takes as input any vector x in $\{0,1\}^n$ and computes

$$S(x) = \sum_{i=1}^{n} a_i x_i$$

A *collision* for this function consists of two values x and x' such that $S(x) = S(x')$.

In order to search collisions, we reduce the following lattice:

$$B = \begin{pmatrix} Ka_1 & Ka_2 & \cdots & Ka_n \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

Note that this lattice is exactly the lattice used in the original Lagarias-Odlyzko attack for solving knapsack problems (see [LO83]). Let us consider the possible output of lattice reduction. Since K is large, it is clear that the first coordinate of a short vector is 0. As for the other coordinates, we expect them to be all 0, 1 or -1. Indeed, if this happens we clearly get a collision: from an element of the lattice

$$e = \begin{pmatrix} 0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

with all coordinates 0, 1 or -1, we get that

$$\sum_{i=1}^{n} \epsilon_i a_i = 0$$

and thus:

$$\sum_{\epsilon_i=1} a_i = \sum_{\epsilon_i=-1} a_i.$$

3.2.2 Practical results in small size

As was stated above, finding a collision for Damgård's hash function amounts to compute collisions for a knapsack compression function based on 128 numbers with 120 bits each. We develop this approach in appendix E where we show that actual lattice reduction programs disclose such collisions. Thus, Damgård hash-function can be broken using lattice reduction.

3.2.3 The size of pseudo-collisions

Pseudo-collisions are collisions for the function obtained by replacing the original knapsack-compression function by reducing it modulo some power of two. This means that we will be working with modular knapsacks instead of usual knapsacks. In this subsection and the next one, we use the same approach as in the Lagarias-Odlyzko attack. More precisely, we fix a value $\tau < 1$, we let $m = \lfloor \tau n \rceil$ and assume that the a_is are random integers between 0 and $2^m - 1$. Our aim is to compute collisions for the resulting knapsack function, where reduction modulo 2^m is performed

$$S(x) = \sum_{i=1}^{n} x_i a_i \bmod 2^m$$

Before applying lattice reduction techniques, we need to estimate the minimum size of a collision, i.e. the minimum size of $x - x'$, where $S(x) = S(x')$. The following lemma provides a bound.

Lemma 2 *Let ρ be a fixed constant such that*

$$\rho + H_2(\rho) > \tau > \rho$$

With probability tending exponentially to 1 when n tends to infinity, there exists a relation

$$\sum_{i=1}^{n} \epsilon_i a_i = 0$$

where all ϵ_is are 0, 1 *or* -1 *and where*

$$\sum |\epsilon_i| \leq \rho n$$

In the above $H_2(\alpha)$ denotes, as usual, $-\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$.
Proof: Consider the family of all possible vectors with n coordinates, all of them 0, 1 or -1, with size ρn. This family has N members where $N = 2^{\rho n}\begin{pmatrix} n \\ \rho n \end{pmatrix}$ which, by classical estimates, is roughly $2^{\rho n}2^{H_2(\rho)n}$. For any such vector ϵ, let $S(\epsilon) = \sum_{i=1}^{n} \epsilon_i a_i$. $S(\epsilon)$ is a random variable depending on the random numbers a_i. When ϵ varies, we get (roughly) $2^{(\rho + H_2(\rho))n}$ random variables and it is easily seen that, provided one discards one vector out of each pair $\{\epsilon, -\epsilon\}$, these variables are pairwise independent and uniformly distributed over the integers $\{0, \cdots 2^m - 1\}$. Consider the characteristic function χ_ϵ of the event $S(\epsilon) = 0$. We get $N/2$ pairwise independent random variables with mean value $\delta = \frac{1}{2^m}$ and standard deviation $\sqrt{\delta(1-\delta)}$. By Chebychev's inequality the probability that

$$\sum_{\epsilon} \chi_\epsilon < \frac{N\delta}{4}$$

is bounded by $\frac{8}{N\delta}$. Now if

$$\sum_{\epsilon} \chi_\epsilon \geq \frac{N\delta}{4}$$

then, as soon as $\frac{N\delta}{4}$ exceeds one, a suitable collision is obtained. To see this, observe that some value $S(\epsilon)$ is zero and therefore

$$\sum_{\epsilon_i = 1} a_i = \sum_{\epsilon_i = -1} a_i$$

The condition on N can be written $N \geq 4 2^{\tau n}$ and is satisfied, for n large enough, under the hypothesis of the lemma. We get that a collision exists with probability $\geq 1 - \frac{8}{N\delta}$. As $N\delta$ is exponentially small, this concludes the proof of the lemma.
Remarks.
1) The careful reader will have noticed that the above proof is (consciously) flawed. Due to the fact that 2^m is even, $S(\epsilon)$ and $S(\epsilon')$ are not independent when ϵ and ϵ' have the same domain. This involves a correction term for the standard deviation σ of

$$\sum_{\epsilon} \chi_\epsilon$$

The correction for σ^2 is bounded by

$$\delta^2 2^{H_2(\rho)n} 2^{2\rho n} \leq N\delta 2^{(\rho-\tau)n}$$

This gives an opportunity to use the hypothesis $\tau > \rho$ in order to conclude that the correction is negligible.
2) Let $L(\tau)$ be the real number $< 1/2$ defined by

$$L(\tau) + H_2(L(\tau)) = \tau$$

Then, for ρ slightly greater than $L(\tau)$, the hypotheses of the lemma hold. Hence we may sum up the lemma by stating that collisions of size $L(\tau) + o(1)$ almost always exist.

3.2.4 A provable mildly exponential attack for pseudo-collisions

For the rest of the paper, we assume that we are granted access to a lattice reduction oracle. As was already pointed out in section 2.4, this approach, provides a way to focus on the reduction of a given problem (here collision search) to a lattice problem, without needing to worry about the state of the art in lattice reduction algorithms. Recall that a lattice oracle outputs at no computing cost a shortest vector of a given lattice. In practice, any call to the oracle will be replaced either by the LLL algorithm [LLL82] or by a blockwise Korkine-Zolotarev algorithm.

In order to search collisions in the modular case, we could try to reduce the following lattice, which is a variant of the one used above:

$$B = \begin{pmatrix} Ka_1 & Ka_2 & \cdots & Ka_n & K2^m \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

Unfortunately, a lattice oracle is not guaranteed to output a vector with all coordinates 0, 1 or -1, even given the known bound for such a vector that comes from the previous lemma. In order to take advantage of this known bound, we fix a subset Y of $\{1, \cdots, n\}$ with αn elements together with a function σ from Y into $\{-1, +1\}$. We then coalesce those a_is with index in Y by setting

$$b_0 = \sum_{i \in Y} \sigma(i) a_i \bmod 2^m$$

Reindexing the other a_is as $b_1, \ldots, b_{(1-\alpha)n}$, we thus obtain a modular knapsack containing $(1-\alpha)n + 1$ random modular numbers $b_0, b_1, \ldots, b_{(1-\alpha)n}$. We can now associate the following lattice to the knapsack problem:

$$B' = \begin{pmatrix} Kb_0 & Kb_1 & \cdots & Kb_{(1-\alpha)n} & K2^m \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

We know that the lattice generated by B contains a short vector of size $(L(\tau)+o(1))n$ with coordinates 0, 1 or -1, where $L(\tau)$ has been defined in the previous section. If the restriction of this vector to Y matches up with σ, then, grouping the indices in Y, we get a short vector of B' of size $(L(\tau)+\epsilon-\alpha)n$ with coordinates 0, 1 or -1. Using an argument similar to those in the Lagarias-Odlyzko paper [LO83], we can prove that, with probability tending exponentially to 1, a lattice oracle will output such a vector, provided α exceeds some value depending on τ. Details of the proof appear in appendix D. It turns out that, even though we can't give a close form for the minimum value of α, we obtain that α stays bounded by 1/3000.

We can now derive a provable mildly exponential algorithm, where we pick at random a subset of size αn and try all functions σ from such a subset into 1s and -1s. This algorithm requests $O(2^{\alpha n})$ calls to the lattice reduction oracle. Its probability of success can be estimated as $2^{-\mu n}$ with

$$\mu = -(1-\alpha)\log(1-\alpha) - \rho\log\rho + (\rho-\alpha)\log(\rho-\alpha)$$

Computations show that the expected number of oracle calls $2^{(\alpha+\mu)n}$ remains bounded by $2^{n/1000}$. Albeit exponential, this is much more efficient than exhaustive search.

References

[Adl83] L. M. Adleman. On breaking generalized knapsack public key cryptosystems. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 402–412, 1983.

[Boy89] J. Boyar. Inferring sequences produced by a linear congruential generator missing low-order bits. *J. Cryptology*, 1(3):177–184, 1989.

[Bri85] E. F. Brickell. Breaking iterated knapsacks. In G. R. Blakley and D. C. Chaum, editors, *Proceedings CRYPTO 84*, pages 342–358. Springer, 1985. Lecture Notes in Computer Science No. 196.

[CJL+92] M. J. Coster, A. Joux, B. A. LaMacchia, A. Odlyzko, C.-P. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2:11–28, 1992.

[CJS91] Y. M. Chee, A. Joux, and J. Stern. The cryptanalysis of a new public-key cryptosystem based on modular knapsacks. In J. Feigenbaum, editor, *Advances in Cryptology: Proceedings of Crypto'91*, volume 576 of *LNCS*, pages 204–212. Springer-Verlag, 1991.

[CP91] P. Camion and J. Patarin. The knapsack hash-function proposed at crypto'89 can be broken. In D. W. Davies, editor, *Advances in Cryptology, Proceedings of Eurocrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 39–53, New York, 1991. Springer-Verlag.

[Dam89] I. Damgård. A design principle for hash functions. In *Advances in Cryptology, Proceedings of Crypto'89*, volume 435 of *Lecture Notes in Computer Science*, pages 25–37, New York, 1989. Springer-Verlag.

[FHK+88] A. M. Frieze, J. Hastad, R. Kannan, J. C. Lagarias, and A. Shamir. Reconstructing truncated integer variables satisfying linear congruences. *SIAM J. Computing*, 17(2):262–280, April 1988.

[Fri86] A. M. Frieze. On the Lagarias-Odlyzko algorithm for the subset sum problems. *SIAM J. Comput.*, 15(2):536–539, 1986.

[Gau01] C.F. Gauss. Disquisitiones arithmeticae. Leipzig, 1801.

[GJ94] L. Granboulan and A. Joux. A practical attack against knapsack based hash functions. *Proceedings of EUROCRYPT 94*, 1994. to appear.

[Her50] C. Hermite. Extraits de lettres de M. Hermite à M. Jacobi sur différents objets de la théorie des nombres, deuxième lettre. *J. Reine Angew. Math*, 40:279–290, 1850.

[Jou93] A. Joux. *La Réduction de Réseaux en Cryptographie.* PhD thesis, Ecole Polytechnique, Palaiseau, France, 1993.

[JS91] A. Joux and J. Stern. Cryptanalysis of another knapsack cryptosystem. In *Advances in Cryptology: Proceedings of AsiaCrypt'91*, volume 739 of *Lecture Notes in Computer Science*, pages 470–476. Springer-Verlag, 1991.

[Knu69] D.E. Knuth. *The Art of Computer Programming: Vol. 2, Seminumerical Algorithms.* Addison-Wesley, 1969.

[Knu80] D. E. Knuth. Deciphering a linear congruential encryption. Technical Report 024800, Stanford University, 1980.

[KZ73] A. Korkine and G. Zolotarev. Sur les formes quadratiques. *Math. Ann.*, 6:336–389, 1873.

[Lag73] L. Lagrange. *Recherches d'arithmétique*, pages 265–312. Nouv. Mém. Acad., Berlin, 1773.

[Len83] H. W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Methematische Ann.*, 261:513–534, 1982.

[LO83] J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 1–10. IEEE, 1983.

[MH78] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Inform. Theory*, IT-24:525–530, September 1978.

[Min10] H. Minkowski. *Geometrie der Zahlen.* Teubner, Leipzig, 1910.

[Plu82] J. Plumstead. Inferring a sequence generated by a linear congruence. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 153–159, Chicago, 1982. IEEE.

[Sch87] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.

[Sch88] C.-P. Schnorr. A more efficient algorithm for lattice basis reduction. *J. Algorithms*, 9:47–62, 1988.

[SE91] C.-P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In L. Budach, editor, *Proceedings of Fundamentals of Computation Theory 91*, volume 529 of *Lecture Notes in Computer Science*, pages 68–85, New York, 1991. Springer-Verlag.

[Sha82] A. Shamir. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 145–152. IEEE, 1982.

[ST90] J. Stern and P. Toffin. Cryptanalysis of a public-key cryptosystem based on approximations by rational numbers. In *Advances in Cryptology: Proceedings of Eurocrypt'90*, volume 473 of *Lecture Notes in Comp Sci*, pages 313–317. Springer-Verlag, 1990.

[Ste87] J. Stern. Secret linear congruential generators are not cryptographically secure. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 421–426. IEEE, 1987.

Appendix A

In section 3.1.1, we had to consider a lattice $L(a)$ generated by the columns of the following matrix

$$\begin{pmatrix} 1 & 0 & & \cdots & 0 \\ a & m & & \cdots & 0 \\ a^2 & 0 & m & \cdots & 0 \\ \vdots & & & \vdots & \\ a^{t-1} & 0 & 0 & \cdots & m \end{pmatrix}$$

We then claimed that a vector U belonging to $L(a)$ and of size $O(m^{\beta+o(1)})$ was presumably equal to zero. When m is a squarefree integer, this is justified by the following:

Theorem 1 *Let t and ϵ be given. There exist constants $K(t)$, $c(\epsilon,t)$ and an exceptional set $E(m,\epsilon,t)$ of values of the multiplier a, such that*
i) $\mid E(m,\epsilon,t) \mid \leq m^{1-\epsilon}$
ii) for any $m > c(\epsilon,t)$ taken outside the exceptional set $E(m,\epsilon,t)$, $\lambda(L)$ is bounded from below by $K(t)m^{\frac{t-1}{t}-\epsilon}$

The proof if this theorem is implicit in [FHK+88]: since it is doubtful that the present appendix will be of any use to readers not familiar with this paper, we simply indicate how to extract the above result from the proof of lemma 3.2 of [FHK+88]. Our lattice $L(a)$ is exactly obtained from a lattice denoted L_a^* in this proof by multiplying all vectors by m and our variable t stands for k. With these minor notational changes in mind, one can see that it is proved in [FHK+88] that, for m exceeding some constant $c(\epsilon,t)$, the following holds:

$$\lambda(L_a^*) \geq \frac{1}{2t}3^{-(t-1)}m^{-\frac{1}{t}-\epsilon}$$

This is precisely what we need.

Following [FHK+88], the theorem can be extended to the case when m is δ-squarefree i.e. when

$$m = \prod_{i=1}^{f} p_i^{e_i} \quad \text{and} \quad \prod_{i=1}^{f} p_i^{e_i-1} \leq m^{\delta}$$

The resulting bound becomes

$$K(t)m^{\frac{t-1}{t}-\delta-\epsilon}$$

Following [FHK+88], again, the condition on m can be dropped when $t = 3$. We simply state the result.

Theorem 2 *Let ϵ be given. There exist constant $K(t)$, $c(\epsilon)$ and an exceptional set $E(m, \epsilon)$ of values of the multiplier a, such that*
i) $|E(m, \epsilon, t)| \leq c(\epsilon) m^{1-\epsilon/2}$
ii) for any m taken outside the exceptional set $E(m, \epsilon)$, $\lambda(L)$ is bounded from below by $Km^{\frac{2}{3}-\epsilon}$

Appendix B

In this appendix, we investigate the probability that $x_1 - x_0$ is not prime to m. This appears in the analysis of our algorithm against Knuth's truncated linear congruential generator. If this happens, a part of our argument fails: the one by which we showed that the polynomial $P(x)$, found at the end of the first step, vanishes at a mod$m$. Fortunately, this is not too likely as shown by the following.

Theorem 3 *Assume m is δ-squarefree, then, if a and x_0 are chosen independently and uniformly from the integers* mod$m$, *then the probability that m and $x_1 - x_0$ are not relatively prime is bounded by $2m^{\delta-1}$.*

Proof: We have

$$x_1 - x_0 = (a-1)x_0 \bmod m$$

Now, it is easily seen that $x_1 - x_0$ is not prime to m if either x_0 or $a - 1$ has non-trivial gcd with m. Each event happens with probability $\frac{\prod_{i=1}^{f} p_i^{e_i - 1}}{m} \leq m^{\delta-1}$

Appendix C

In this appendix, we show that if L and $\tilde{L}$ are full dimensional lattices, L a sublattice of $\tilde{L}$ and if random elements of L, say $P_1, \cdots, P_q$ are taken from a ball of fixed (large) radius, then, as soon as q is large enough, L is, with high probability, the lattice spanned by a basis of $\tilde{L}$ together with $P_1, \cdots, P_q$. This is related to the second part of our algorithm against Knuth's truncated linear congruential generator, where we use a bunch of polynomials obtained from step one in order to disclose the values of m and a. Although, this offers no proof of the success of our attack, this gives strong evidence that the algorithm will disclose the correct secret values. Repeated experiments confirm this heuristic analysis.

We let L_i be the lattice generated by L and $P_1, \cdots, P_i$. By volume estimates, we can see that the probability that P_i is taken within L_{i-1} is essentially equal to

$\frac{\Delta(L)}{\Delta(L_{i-1})}$ (the word essentially coming from the fact that the ball from which the P_is are drawn does not consist of an exact number of parallelepiped generated by a basis of L). In in any case, since both determinants are integers, this is bounded by $\tau = 1/2 + \epsilon$, unless $L = L_{i-1}$. When P_i lies in L_{i-1} or, equivalently, $L_i = L_{i-1}$, we say that i is a stationary index. We let $k = \log(\frac{\Delta(\tilde{L}}{\Delta(L)}))$. If, at least k indices are non-stationary, then, since for every such index the determinant of L_i decreases by a multiplicative integer ratio, we get $\Delta(L_q) \leq \frac{\Delta(\tilde{L})}{2^k}$ and therefore $\Delta(L_q) < 2\Delta(L)$. This, in turn implies $\Delta(L_q) = \Delta(L)$ since $\Delta(L_q)$ is a multiple of $\Delta(L)$ and finally $L_q = L$. The following lemma bounds the probability that this equality does not hold.

Lemma 3 *There exists a constant $\Lambda(\epsilon)$ such that whenever q is at least $\Lambda(\epsilon)k$, the probability that $L_q \neq L$ is exponentially small w.r.t. k*

Proof: If $L_q \neq L$, then, for each $i \leq q$, L_i is different from L and the number of nonstationary indices is less than k. For a fixed subset S of $\{1, \cdots, q\}$, the probability that this situation happens with S being the set of nonstationary indices is at most $\tau^{q-|S|}$. This gives for the requested probability π the upper bound

$$\sum_{i=0}^{k} \binom{q}{i} \tau^{q-i}$$

Using the classical upper estimate

$$\sum_{i=0}^{\alpha q} \binom{q}{i} \leq 2^{qH_2(\alpha)}$$

where $H_2(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$. we get that

$$\pi \leq 2^{k[\lambda H_2(\frac{1}{\lambda}) + (\lambda - 1)\log \tau]}$$

Since the coefficient of k in the exponent is equivalent to $\lambda \log \tau$ when λ tends to infinity, the exponent is $\leq -k$ when λ exceeds some constant $\Lambda(\epsilon)$.

Appendix D

In this appendix, we clarify the link between the Lagarias-Odlyzko attack against low-density knapsacks and our cryptanalysis of knapsack hash-functions. The Lagarias-Odlyzko scenario deals with elements of a given knapsack together with a target sum, which is obtained from a proportion p of the elements. Thus, we know that in the Lagarias-Odlyzko lattice, there exists a short vector of size roughly $\sqrt{pn}$. The question amounts to the following: Is this vector the shortest one? If the answer is yes, then an oracle for lattice reduction successfully decrypts the cipher, otherwise the proof fails. We now follow the analysis of the Lagarias-Odlyzko attack that appears in [Fri86] and refer to this paper. The basic method of [Fri86] is to count the number of integer points lying in the sphere of radius $\sqrt{pn}$ centered at the origin and to claim that, for each such integer point, the probability that it provides a short vector of the Lagarias-Odlyzko lattice is $2^{-n/d}$ where d is the density of the knapsack. From this analysis, one can derive a sufficient condition ensuring that a call to a lattice oracle inverts the cipher with very high probability. The condition is written:

$$c(p) \leq \frac{1}{d}$$

where $c(p)$ is defined by $c(p) = \log_2(h(z_0)/z_0^p)$, with $h(z) = 1 + 2\sum_{k=1}^{\infty} z^{k^2}$ and z_0 the unique solution in $]0,1[$ of

$$z\frac{h'}{h}(z) = p,$$

In our attack against hash-functions, the analysis is similar but the parameters differ. More precisely, once we have guessed αn correct bits of a collision, as explained in section 3.2.4, we obtain a short vector of size $(L(\tau)+\epsilon-\alpha)n$ which belongs to the lattice built from a knapsack consisting of $(1-\alpha)n$ numbers with τn bits. Thus the values of the parameters are:

$$p = \frac{L(\tau)+\epsilon-\alpha}{1-\alpha}$$

and

$$d = \frac{1-\alpha}{\tau}.$$

The inequality $c(p) \leq \frac{1}{d}$ can accordingly be written as

$$\alpha \geq G_\epsilon(\tau).$$

Since ϵ is as small as we want, we set $\epsilon = 0$, and compute G_0.

Clearly

$$\alpha < L(\tau) < 1,$$

thus p as well as $c(p)$ are decreasing functions of α. Also, $1/d$ is an increasing function of α so that, for fixed τ, the curves $y = c(p)$ and $y = 1/d$ with α ranging in the interval $[0, L(\tau)]$ cross at a single point. The corresponding value of α at the cross point is the required real number $G_0(\tau)$. There is no closed form for $G_0(\tau)$. Still, we have made numerical computations, and computed the curve $\alpha = G_0(\tau)$ as shown in figure 3.2.4.

In order to bound the expected number of oracle calls $2^{(\alpha+\mu)n}$, we also computed μ as a function of τ for the limit case $\epsilon = 0$. The resulting curve is shown in figure 3.2.4.

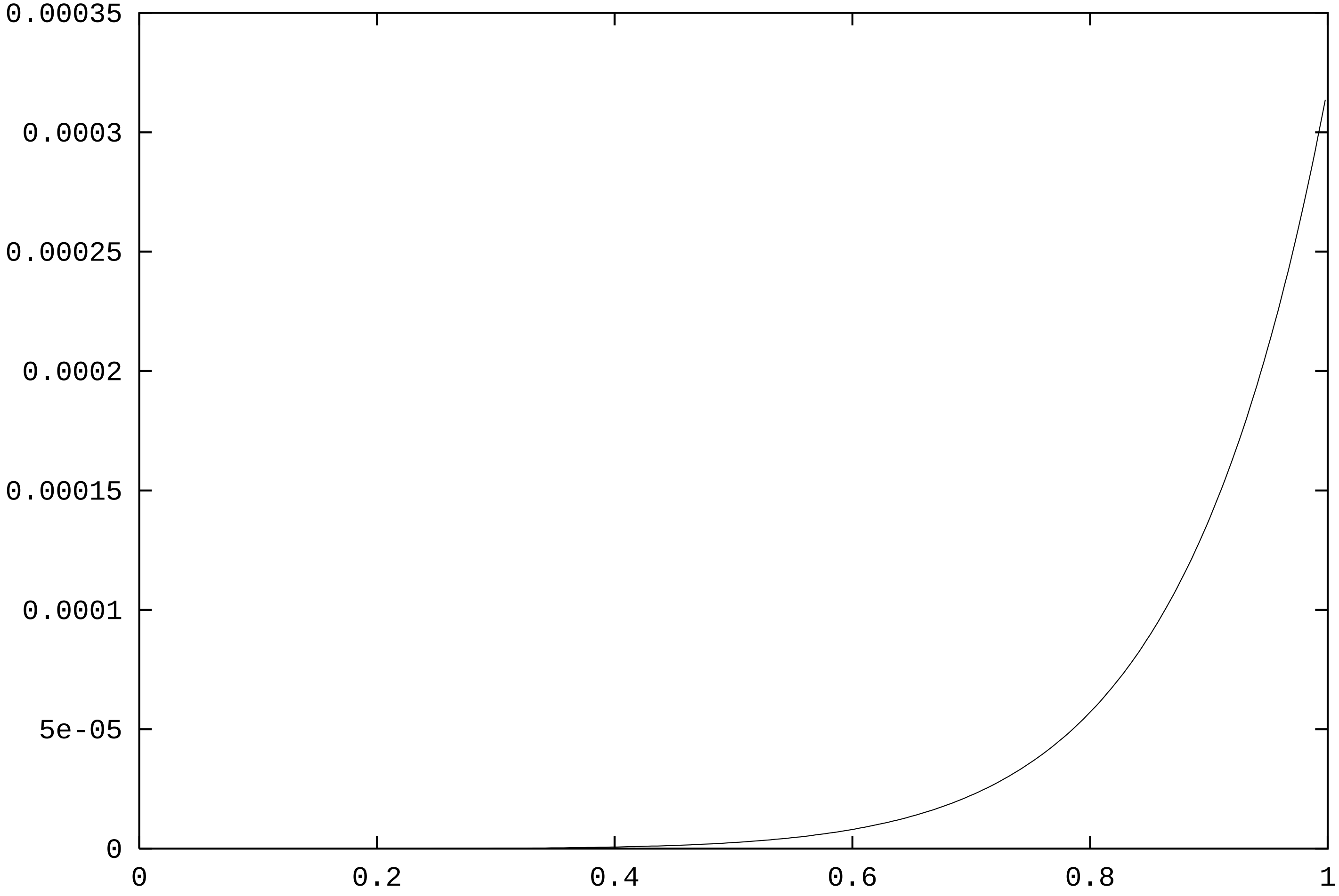

Figure 1: α as a function of τ

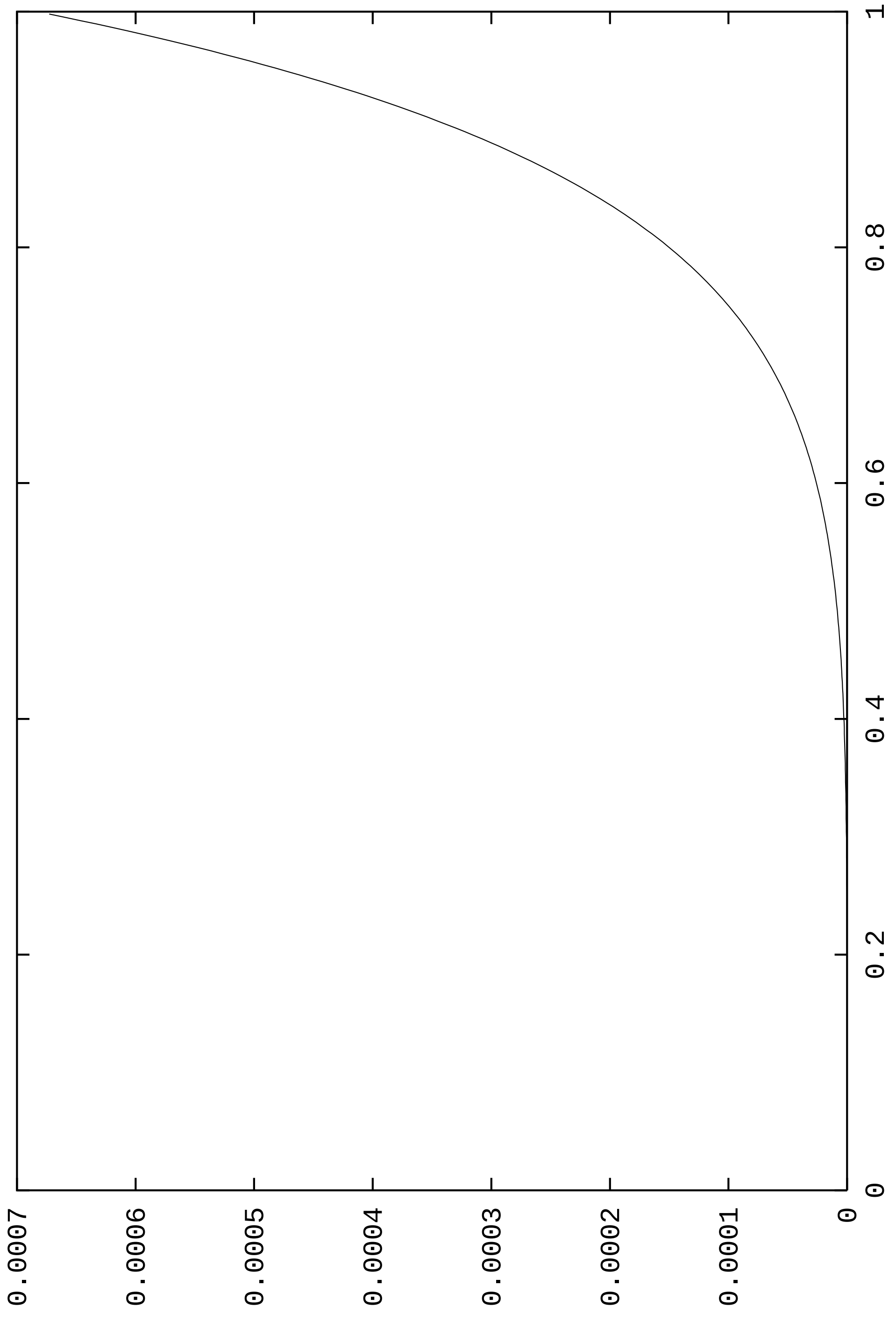

Figure 2: μ as a function of τ

Appendix E

In this section, we report on practical results showing that collisions in knapsack compression functions with compression rate $\tau = 1$ can be actually found, at least up to dimension 120. In order to obtain these results, we have devised a specific lattice reduction algorithm that is very efficient for the kind of lattices involved. This algorithm is a variation of Schnorr and Euchner (see [SE91]) pruned blockwise Korkine-Zolotarev reduction. There are two slight changes in the algorithm, which we briefly discuss for the cognoscendi. Firstly, whenever the first vector in the lattice consists only of 0, 1 and -1, the program stops since it has found a collision. The second change is in the so-called enumeration step: during this enumeration, the program searches through the same space as in the original algorithm but, instead of searching the whole space and choosing the new vector at the current position as the vector with shortest possible projection, the enumeration stops as soon as an improvement for the current position has been found. The following table gives the success rate and average running time for blocksize 50 and dimension varying from 50 to 95. The experiments were made on an IBM RS6000 model 590.

Dimension	Success rate	Av. Time
50	10/10	3.7s
55	10/10	8.9s
60	10/10	11.5s
65	10/10	19.4s
70	10/10	26.9s
75	10/10	44.6s
80	10/10	76.5s
85	10/10	225.2s
90	8/10	234.2s
95	8/10	461.4s

Dealing with dimension 120 is more difficult since unsuccessful trials can appear that waste a lot of time. We have bypassed this problem by limiting the running time for each trial. Using this technique, we have obtained the following results:

Time limit	# trials	# success	rate
1h	100	3	0.03
4h	20	3	0.15
$\approx$ 12h	30	8	0.27

<u>Remark</u>: The experiments with time limit $\approx$ 12h, were not made on the IBM, but on a Sun Sparc 10 model 51, which is roughly twice slower. The actual time limit was 24h.