MASTER'S THESIS

Lattice Reduction and Polynomial Solving

Raphaël Marinier

Advisors: Jean-Charles Faugère et Guénaël Renault
Equipe Projet SALSA INRIA/LIP6
104 avenue du Président Kennedy
75016 Paris
February 23 – July 24 2009

1 Introduction

Since 1996, Coppersmith's methods for finding integer or modular roots of univariate or multivariate polynomials are well-known tools to cryptographers.

The univariate case was first studied in [6] by Coppersmith. The problem (in its most general formulation) is as follows: Given a monic polynomial $f \in \mathbb{Z}[x]$ of degree δ, and an integer N of *unknown* factorization that has a divisor $p \geq N^\beta$, $0 < \beta \leq 1$ [1], find efficiently all integer solutions x_0 for the equation $f(x) = 0 \mod p$. The most recent results state that we can find all such solutions verifying $|x_0| \leq cN^{\frac{\beta^2}{\delta}}$ in time $\mathcal{O}(c\delta^5 \log^9 N)$. They led to numerous applications, including attacks related to RSA and factorization with partial information (see [19] for a survey).

In the multivariate case, we are given a polynomial $f \in \mathbb{Z}[x_1, \ldots, x_n]$, and we want to find efficiently either its integer or modular (modulo an unknown divisor of an integer N) solutions $(x_{1,0}, \ldots, x_{n,0})$ satisfying $|x_{i,0}| < X_i$, where the X_i's are bounds that we want to maximize. The problem has first been studied by Coppersmith in [8]. Results concerning this problem depend upon the set of monomials of the input polynomials and are thus a bit more difficult to state. They are presented in section 3 for the case in which we are looking for integer solutions. One famous application of this problem is the cryptanalysis of RSA with small private key of Wiener [26] and Boneh-Durfee [4]. One major problem with this kind of methods is that they become heuristic (with non-understood failures) as soon as $n \geq 3$, except in some special cases with $n = 3$ that were made rigorous by Bauer and Joux in [2].

One goal of the internship was to generalize Coppersmith's multivariate method to a system of polynomial equations. Namely, we are given k algebraically independent polynomials $f_1, \ldots, f_k \in \mathbb{Z}[x_1, \ldots, x_n]$ and our goal is to find efficiently all integer solutions $(x_{1,0}, \ldots, x_{n,0})$ of the following system of equations:

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \qquad \vdots \\ f_k(x_1, \ldots, x_n) = 0 \end{cases}$$

satisfying $|x_{1,0}| < X_1, \ldots, |x_{n,0}| < X_n$ where the X_i's are bounds we want to maximize. Note that given such a system, one can apply Coppersmith's original method on one of the f_i's, or on a polynomial coming from the elimination of variables in the system. Our bounds should therefore be better than what we can achieve using the two aforementioned ideas.

Our generalization is given in section 4. Due to obstructions, we were not able to prove any bounds. Our method is thus fully heuristic (as is Coppersmith's multivariate method for $n \geq 4$). We present nevertheless encouraging experimental results in section 4.3.

The second problem that we studied intensively is the implicit hint factoring problem, as this is one application (appeared recently in [25]) of Coppersmith's multivariate method.

The implicit hint factoring problem first appeared in PKC 2009 in the paper of May and Ritzenhofen [20]. The problem is as follows: Let $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$ be two RSA moduli of unknown factorization. The attacker is given the additional information that p_1 and p_2 share some of their bits (it can be

[1] $\beta = 1$ is an important case.

most significant bits, or least significant bits, or both, or bits in the middle). The problem is to find under which conditions on the number and layout of the shared bits N_1 and N_2 can be factored in polynomial time in $\log_2(N_1)$ and $\log_2(N_2)$. Note that unlike the well-known Coppersmith's result [8] which states that a balanced RSA modulus $N = pq$ can be factored in polynomial time as soon as we know half of the most significant bits of p, here the additional information is only *implicit*. That is, the shared bits between p_1 and p_2 are *unknown* to the attacker. Finally, the problem can be extended in the case the attacker has access to k distinct RSA moduli $N_i = p_i q_i$, with all the p_i sharing bits.

May and Ritzenhofen's result [20] applies in the case where p_1 and p_2 share *least* significant bits (LSBs). Namely, if q_1 and q_2 are α-bit primes, and if p_1 and p_2 share t LSBs, their lattice-based method allows them to provably factorize both N_1 and N_2 in quadratic time as soon as $t \geq 2\alpha + 3$. Of course, this bound implies that the method applies to unbalanced RSA moduli, which are not used in practice. Let's take a numerical example to make things clearer. If N_1 and N_2 are 1000-bit RSA moduli, q_1, q_2 are 250-bit primes, and p_1, p_2 are 750-bit primes. May and Ritzenhofen's result state that we can factorize N_1 and N_2 in polynomial time when p_1 and p_2 share at least 503 bits (that is, we can "discover" as much as 247 bits with these parameters). Besides, when they are provided with k RSA moduli with all the p_i's sharing t LSBs, their method generalizes and they can factorize all the N_i's in polynomial time as soon as $t \geq \frac{k}{k-1}\alpha$. Note that this last result is heuristic, as it relies on an assumption about a shortest vector of a k-dimensional lattice.

Very recently, in [25], Sarkar and Maitra applied Coppersmith's multivariate method to the problem of implicit factoring, and improved heuristically the previous bounds in some cases. Indeed, they reduced the problem of implicit factoring (with shared MSBs and/or LSBs, or shared bits in the middle) to the one of finding integer roots of trivariate integer polynomials (which makes their method heuristic). Their method does not generalize to several RSA moduli. We give an insight into their work in section 3.2.

One of the most fruitful part of the internship was to try to generalize Sarkar et al.'s results to various RSA moduli, using our generalized Coppersmith's method. We were first able to reduce the implicit factoring problem with $k \geq 3$ RSA moduli to finding integer roots of a system of polynomial equations in section 4.3. In our experiments, our generalized method allowed us to solve the implicit factoring problem with 3 RSA moduli with better bounds than Sarkar et al.'s, though in an unexpected way. That gave us crucial hints to devise a direct lattice-based method that solves the implicit factoring problem with shared MSBs, and with an arbitrary number k of moduli. This method is described in section 5 and led to the writing of an article. They form the main results obtained during the internship. Our contribution consists of a novel and rigorous lattice-based method that address the implicit factoring problem when p_1 and p_2 share *most* significant bits. That is, we obtained an analog of May and Ritzenhofen's results for shared MSBs, and our method is rigorous contrary to the work of Sarkar and Maitra in [25]. Namely, let $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$ be two RSA moduli of same bit-size n. If q_1, q_2 are α-bit primes and p_1, p_2 share t most significant bits, our method provably factorizes N_1 and N_2 as soon as $t \geq 2\alpha + 3$ (which is the same as the bound on t for least significant bits in [20]). This is the first rigorous bound on t when p_1 and p_2 share most significant bits. Moreover, contrary to [25], the method heuristically generalizes to an arbitrary number k of RSA moduli, allowing us to factorize k RSA moduli as soon as $t \geq \frac{k}{k-1}\alpha + 6$ MSBs are shared between the p_i's (a more precise bound is stated later in this report). A summary of the comparison of our method with the methods in [20] and [25] can be found in table 1.

Table 1: Comparison of our results for the problem of implicit factoring against the results of [20] and [25]

k (number of RSA moduli)	Results of May and Ritzenhofen in [20]	Results of Sarkar and Maitra in [25]	Our results
$k = 2$	When p_1, p_2 share t LSBs: rigorous bound of $t \geq 2\alpha+3$ using 2-dimensional lattices of $\mathbb{Z}^2$.	When p_1, p_2 share either t LSBs or MSBs: heuristic bound better than $t \geq 2\alpha + 3$ when $\alpha \geq 0.266n$, and experimentally better when $\alpha \geq 0.21n$, using 46-dimensional lattices of $\mathbb{Z}^{46}$.	When p_1, p_2 share t MSBs: rigorous bound of $t \geq 2\alpha+3$ using 2-dimensional lattices of $\mathbb{Z}^3$.
$k \geq 3$	When the p_i's all share t LSBs: heuristic bound of $t \geq \frac{k}{k-1}\alpha$ using k-dimensional lattices of $\mathbb{Z}^k$.	Cannot be applied.	When the p_i's all share t MSBs: heuristic bound of $t \geq \frac{k}{k-1}\alpha + \delta_k$, with $\delta_k \leq 6$ and using k-dimensional lattices of $\mathbb{Z}^{\frac{k(k+1)}{2}}$.

2 Preliminaries

We quickly set the notations and state some of the results used in this report. A reader interested in getting a more complete introduction to Gröbner basis terminology can refer to [11].

Let k be an algebraically closed field and $f \in k[x_1, \ldots, x_n]$ be a multivariate polynomial. We call the set of its monomials its *shape*. Let M be a set of monomials. We will say that f is *defined over* M when the sets of its monomials is included in M. We will often associate to a monomial $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ the point $(\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, and to M the set of all the points associated with all its elements. This allows us to speak about sets of monomials as geometric shapes. For instance, $\{x_1^{\alpha_1} \cdots x_n^{\alpha_n} \mid \sum_{i=1}^n \alpha_i \leq D\}$ will be called a *lower-triangle*. Besides, we denote the euclidean norm on polynomials by $\|.\|$.

An ideal I generated by polynomials $f_1, \ldots, f_p$ is denoted by $I = \langle f_1, \ldots, f_p \rangle$. I is by definition $\{a_1 f_1 + \cdots + a_p f_p \mid \forall i, a_i \in k[x_1, \ldots, x_n]\}$ and $f_1, \ldots, f_p$ is one of its *basis*. The *variety* defined by I (or the set of monomials $f_1, \ldots, f_p$) is the set of points of k^n on which all the f_i's simultaneously vanish. It is invariant by the choice of the basis of I. Its *dimension* is (very) informally the maximum number of free variables when we describe the surface. An important particular case is the one of 0-dimensional varieties, which are finite sets of points. Note that given a 0-dimensional variety defined by generators of an ideal, there are efficient methods (that use Gröbner basis for instance) that can enumerate its points.

Polynomials $f_1, \ldots, f_p$ are called *algebraically independent* when for any $P \in k[x_1, \ldots, x_p]$, $P(f_1, \ldots, f_p) = 0$ implies $P = 0$. p algebraically independent polynomials of $k[x_1, \ldots, x_n]$ define a variety of dimension $n - p$. In particular, we get a 0-dimensional variety from n algebraically independent polynomials.

A *monomial ordering* is any total and well-ordered relation $\prec$ on the monomials of $k[x_1, \ldots, x_n]$ that is compatible with summation of monomials. The *leading monomial* of a polynomial is its greatest monomial according to $\prec$. A monomial ordering is *graded* when for any monomials α, β, we have $\alpha \prec \beta$ whenever the total degree of β is greater than the total degree of α. Let M be a set of monomials. M is said to be *compatible* with $\prec$ if for any monomials α, β, α is in M whenever $\beta \in M$ and $\alpha \prec \beta$. One useful example

is when $\prec$ is a graded order and M is a lower-triangle.

A Gröbner basis of an ideal I with respect to a specific monomial ordering is a particular basis that informally allows us to do many computations on I, including deciding whether a polynomial is in I and recovering the points of a 0-dimensional variety.

Finally, we also use throughout this report very common results and algorithms on integer lattices. We do not feel the need to detail them here. A reader interested in getting the statements of the results we use may refer to appendix A.

3 Finding integer roots of multivariate polynomials

The problem is as follows. Given an irreducible multivariate polynomial $f \in \mathbb{Z}[x_1, \ldots, x_n]$, find efficiently all the integer roots $(x_{1,0}, x_{2,0}, \ldots, x_{n,0})$ of f satisfying $|x_{i,0}| < X_i$ where the X_i are bounds that we want to maximize. The bounds depend upon n, the degree of f (more precisely, the set of its monomials) and the size of its coefficients. We furthermore assume that the GCD of all the coefficients of f is 1.

In 1996, Coppersmith gave the first method [8] to solve this problem in the bivariate case, using again lattice-based techniques. The method was subsequently simplified by Coron in [9], where the method was however less efficient, and in [10] where he obtained the same efficiency as Coppersmith's original method. Namely, Coppersmith's result is the following:

Theorem 1 (Coppersmith, [8]). *Let $f(x, y) = \sum_{i,j} f_{i,j} x^i y^j$ be an irreducible bivariate polynomial of degree at most δ in each variable and such that the GCD of its coefficients is 1. Let X and Y be bounds on the absolute values of the roots (x_0, y_0) of f that we want to find. Define $W = \max_{i,j} |f_{i,j}| X^i Y^j$. Then, for fixed δ, all the roots verifying $|x_0| < X$ and $|y_0| < Y$ can be recovered in polynomial-time in $\log W$ as soon as:*

$$XY < W^{\frac{2}{3\delta}} \tag{1}$$

Since the bound on the coefficients of the polynomial is defined implicitly, let's illustrate this theorem with two examples: one simple toy example, and a useful application. When f is a bivariate polynomial of degree δ in *each variable* with all coefficients roughly of the size of an integer A, the bound on X and Y becomes: $XY < A^{\frac{2}{\delta}}$.

For the second example, which is the first application of the method given in [8], suppose that $N = pq$ is a balanced RSA moduli. Suppose that we know the high-order $\frac{\log_2(N)}{4}$ bits of p, that is, we know an approximation $\tilde{p}$ of p such that $|\tilde{p} - p| \leq N^{\frac{1}{4}}$. Define $\tilde{q} = \frac{N}{\tilde{p}}$, we have that $|\tilde{q} - q| \lessapprox N^{\frac{1}{4}}$. Let $f(x, y) = (\tilde{p} + x)(\tilde{q} + y) - N = (\tilde{p}\tilde{q} - N) + \tilde{p}y + \tilde{q}x + xy$ and observe that this polynomial has the small root $(x_0, y_0) = (p - \tilde{p}, q - \tilde{q})$ over the integer. Set $X = Y = N^{\frac{1}{4}}$, then $W \geq \tilde{p}Y \gtrapprox N^{\frac{3}{4}}$. Since $W^{\frac{2}{3}} \approx N^{\frac{1}{2}}$, equation 1 of Theorem 1 is satisfied, and the roots can be recovered in polynomial time, which immediately yields the factorization of N.

Surprisingly, the analysis of Coppersmith's method significantly depends upon the shape of the polynomial, and not only upon its degree. It has to be redone for every different shape. Theorem 1 provides bounds for polynomials of maximum degree δ in *each variables*, that is for rectangular-shaped polynomials. In [8],

Coppersmith analyzes also the case of lower triangular-shaped polynomials (polynomials of total degree δ), and obtains a different bound that guarantees the success of the method:

$$XY < W^{\frac{1}{\delta}} \tag{2}$$

Compared to equation 1, this bound is better for triangular-shaped polynomials (and of course much worse for rectangular-shaped polynomials). Much effort has been made to improve the bounds for different polynomial shapes, for instance in [3]. In [17], Jochemsz and May gave a more systematic and simpler method to obtain bounds for arbitrary polynomial shapes. This is the method we present in the next section.

3.1 Overview of the method

We use the notations previously introduced. The method is summarized by these three steps.

1. Using f, construct a collection C of polynomials $g_1, \ldots, g_c \in \mathbb{Z}[x_1, \ldots, x_n]$ such that:

   $$f(x_{1,0}, \ldots, x_{n,0}) = 0 \text{ and } |x_{1,0}| < X_1, \ldots, |x_{n,0}| < X_n \quad \Rightarrow \quad \forall i, g_i(x_{1,0}, \ldots, x_{n,0}) = 0 \mod R$$

   where R is a modulus which is carefully chosen.

2. Find $n-1$ new polynomials $(h_1, \ldots, h_{n-1})$ which are linear combinations of the g_i's and satisfying the following two conditions:

   - $f, h_1, \ldots, h_{n-1}$ are algebraically independent.
   - For every i, $|h_i(x_{1,0}, \ldots, x_{n,0})|$ is strictly smaller than R. That is, the h_i's vanish on the roots of f over the integers.

   This step is carried out by LLL-reducing a lattice spanned by the coefficient vectors of the polynomials $g_i(x_1X_1, \ldots, x_nX_n)$.

3. Finally compute all the integer roots of f within the bounds by enumerating the points of the 0-dimensional variety defined by $(f, h_1, \ldots, h_{n-1})$ and keeping only the integer points.

We can already explain why this method is rigorous for two variables, and heuristic for more than two variables. Indeed, with three or more variables, there is no guarantee that the method in step 2 will output polynomials that are algebraically independent, and we heuristically assume that it will. Besides, this heuristic is not very well understood, as it appears to be always valid in practice in some cases, and to have a high rate of failure in other cases. In [2], Bauer et al. succeeded in making the method fully rigorous in three variables, unfortunately only in a special case.

We now explain the method in more details. We fix an error term $\epsilon > 0$ and an integer m depending on $\frac{1}{\epsilon}$. We call d_j the maximal degree of x_j in f and define $W = \|f(x_1X_1, \ldots, x_nX_n)\|_\infty$. Moreover, we define $R = W \prod_{j=1}^{n} X_j^{d_j(m-1)}$. The method works when f has constant term $a_0 = 1$. If this is not the case, we can increase a little bit the X_j's such that a_0 is invertible modulo R, and use $a_0^{-1} f \mod R$ instead of f. In the case where $a_0 = 0$, we can translate f so that its constant term is not null anymore. Note that this latter trick can modify the analysis, as it changes the set of the monomials of f.

We consider two sets of monomials, S and M. The set S is the set of monomials by which we shift f, and M is such that all the monomials of the shifts of f are included in M. We denote by l_j the largest exponent of x_j that appears in the monomials of S, that is $l_j = d_j(m-1)$. Define the collection C as the following shift polynomials:

$$g : x_1^{i_1} \cdots x_n^{i_n} f(x_1, \ldots, x_n) \prod_{j=1}^{n} X_j^{l_j - i_j} \qquad \text{for } x_1^{i_1} \cdots x_n^{i_n} \in S$$

$$g' : x_1^{i_1} \cdots x_n^{i_n} R \qquad \text{for } x_1^{i_1} \cdots x_n^{i_n} \in M \backslash S$$

Note that all g and g' vanish on $(x_{1,0}, \ldots, x_{n,0})$ modulo R. We construct the lattice L using as a basis the coefficient vectors of the polynomials $g(x_1X_1, \ldots, x_nX_n)$ and $g'(x_1X_1, \ldots, x_nX_n)$. The next step is to reduce L to find $n-1$ small vectors of the lattice. Let's call h_i the polynomials associated with these small vectors. We have to ensure two conditions on the h_i's so that the method is successful. First, $|h_i(x_{1,n}, \ldots, x_{n,0})|$ has to be small enough. This can be guaranteed by imposing a condition on the determinant of the lattice (and thus on the shape and the size of the coefficients of f), using Theorem 9 about the norm of the vectors of an LLL-reduced basis and the following lemma that links the size of $|h_i(x_{1,n}, \ldots, x_{n,0})|$ with the norm of $h_i(x_1X_1, \ldots, x_nX_n)$:

Lemma 1 (Howgrave-Graham). Let $h(x) \in \mathbb{Z}[x_1, \ldots, x_n]$ be a multivariate polynomial with ω monomials, and let R be a positive integer. Suppose that:

1. $h(x_{1,0}, \ldots, x_{n,0}) = 0 \mod R \quad \text{where} \quad |x_{i,0}| < X_i$
2. $\|h(x_1X_1, \ldots, x_nX_n)\| < \frac{R}{\sqrt{\omega}}$

Then $h(x_{1,0}, \ldots, x_{n,0}) = 0$ holds over the integer.

The second condition is to ensure that $(f, h_1, \ldots, h_{n-1})$ are algebraically independent. We can indeed only prove that f is pairwise algebraically independent with each of the h_i's. However, when $n \geq 3$, this is not sufficient to deduce that $(f, h_1, \ldots, h_{n-1})$ are algebraically independent, and the method becomes heuristic. We use Hinek Stinson's lemma shown below, with $a(x_1, \ldots, x_n) = h_i(x_1X_1, \ldots, x_nX_n)$, $b(x_1, \ldots, x_n) = f(x_1X_1, \ldots, x_nX_n)$ and $r = \prod_{j=1}^{n} X_j^{d_j(m-1)}$. It states that if the norm of $h_i(x_1X_1, \ldots, x_nX_n)$ is small enough, then it cannot be a multiple of f. Hence, thanks to lemma 3 (stated latter in this report), f and h_i are algebraically independent.

Lemma 2 (Hinek Stinson, see [16]). Let $a(x_1, \ldots, x_n)$ and $b(x_1, \ldots, x_n)$ be two non-zero polynomials over $\mathbb{Z}$ of maximum degree d in each variable, such that $b(x_1, \ldots, x_n)$ is a multiple of $a(x_1, \ldots, x_n)$ in $\mathbb{Z}[x_1, \ldots, x_n]$. Assume that $a(0, \ldots, 0) \neq 0$ and $b(x_1, \ldots, x_n)$ is divisible by a non-zero integer r such that $\gcd(r, a(0, \ldots, 0)) = 1$. Then:

$$\|b\|_2 \geq 2^{-(d+1)^n+1} |r| \|a\|_\infty$$

In the end, the two aforementioned conditions on the norm of the $n-1$ first basis vectors found by LLL are met (asymptotically in m and thanks to the choice of R) whenever the following condition holds:

$$\prod_{j=1}^{n} X_j^{s_j} < W^{s_W} \quad \text{where } s_j = \sum_{x_1^{i_1} \cdots x_n^{i_n} \in M \backslash S} i_j \text{ and } s_W = |S| \tag{3}$$

The idea of Jochemsz and May in [17] is to take for S and M respectively the sets of monomials of f^{m-1} and f^m. They also state an extended method where they include additional shifts in one variable in the set S. Namely, when we allow s extra shifts over the variable x_1, S and M become:

$$S = \bigcup_{j=0}^{s} \{x_1^{i_1+j} x_2^{i_2} \cdots x_n^{i_n} \mid x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \text{ is a monomial of } f^{m-1}\} \tag{4}$$

$$M = \{\text{monomials of } x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \cdot f \mid x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \in S\} \tag{5}$$

Their method to choose S and M provides a systematic way to adapt the method for specific shapes of f, and allows them to generalize all previously known bounds [3, 13, 7] on $X_1, \ldots, X_n$ that were tailored to specific shapes of polynomials. For instance, for a polynomial $f(x_1, \ldots, x_n)$ where the degree of x_i is $\lambda_i D$ (a so-called generalized rectangular polynomial), we obtain the simple bound on the X_i's:

$$X_1^{\lambda_1} \cdots X_n^{\lambda_n} < W^{\frac{2}{(n+1)D}} \tag{6}$$

And in the case where $f(x_1, \ldots, x_n)$ is of total degree D (a so-called triangular polynomial), the bound becomes:

$$X_1 \cdots X_n < W^{\frac{1}{D}} \tag{7}$$

Notice that the rectangular and triangular bounds degrade quickly as the degree and the number of variables grow.

3.2 Application to the Implicit Factoring Problem

Let $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$ be two n-bit RSA moduli, the q_i's being α-bit primes. We present in this subsection the recent results [25] of Sarkar and Maitra. In order to simplify the exposition, assume that p_1 and p_2 share t *most* significant bits, that are *unknown* to the attacker. Note that the result and method are very similar when p_1 and p_2 share t bits in total distributed between the MSBs and LSBs of p_1 and p_2. We can write $p_1 = p + \tilde{p_1}$ and $p_2 = p + \tilde{p_2}$, where p represents the t shared MSBs of p_1 and p_2, and $\tilde{p_1}, \tilde{p_2}$ are smaller than $2^{n-\alpha-t+1}$. Then, $N_1 = q_1(p + \tilde{p_1})$ and $N_2 = q_2(p + \tilde{p_2})$.

We now consider $q_1, q_2, p, \tilde{p_1}, \tilde{p_2}$ as indeterminates, and eliminate p by multiplying the two previous equalities respectively by q_2 and q_1, and finally subtracting them. We obtain:

$$q_2 N_1 - q_1 N_2 + q_1 q_2 (\tilde{p_2} - \tilde{p_1}) = 0$$

That is, $(q_1, q_2, \tilde{p_2} - \tilde{p_1})$ is a small integer solution of the polynomial equation $f'(x, y, z) = yN_1 - xN_2 + xyz = 0$. As f' has constant term zero, we solve instead

$$f(x, y, z) = f'(x, y - 1, z) = yN_1 - xN_2 - N_1 + xyz - yz = 0 \tag{8}$$

whose solutions are $(x_0, y_0, z_0) = (q_1, q_2 + 1, \tilde{p_2} - \tilde{p_1})$.

Let X, Y, Z be upper-bounds on $q_1, q_2 + 1$ and $|\tilde{p_2} - \tilde{p_1}|$ respectively. Namely, $X = Y = 2^{\alpha}$ and $Z = 2^{n-\alpha-t+1}$. Besides, $W = \|f(xX, yY, zZ)\|_{\infty} \geq N_1 Y \geq 2^{\alpha+n-1}$. Let's apply the bound for rectangular polynomials (equation 6), to derive straightforward bounds on t. In equation 6, we set $D = 1$, $n = 2$ and $\lambda_1 = \lambda_2 = \lambda_3 = 1$, and we obtain after doing the computations the following bound on t:

$$t \geq \frac{\alpha + n + 3}{2} \tag{9}$$

That means that as soon p_1 and p_2 share at least $\frac{\alpha+n+3}{2}$ most significant bits, we are able (heuristically) to factorize N_1 and N_2 in polynomial time. Note that this bound is already better than May and Ritzenhofen's results (which is $t \geq 2\alpha + 3$) as soon as $\alpha \geq \frac{n}{3}$.

The bound on t obtained by Sarkar and Maitra is a bit better (though much more complicated) than inequality 9 we showed for the sake of argument. Rather than simply applying the bound for rectangular-shaped polynomials, they evaluated the quantities s_j and s_W of equation 3 and used the sets S and M of equations 4, 5 from the extended method. That leads to the following theorem:

Theorem 2 (Sarkar, Maitra, [25]). *Let $N_1 = p_1 q_1$, $N_2 = p_2 q_2$ where p_1, p_2 are α-bit primes, and q_1, q_2 are primes. Assume that p_1 and p_2 share t_1 MSBs and t_2 LSBs, and define $t = t_1 + t_2$. Let $\beta = n - \alpha - t$. If*

$$-4\alpha^2 - 2\alpha\beta - \frac{1}{4}\beta^2 + 4\alpha n + \frac{5}{3}\beta n - n^2 < 0 \quad \textit{and} \quad n - \frac{3}{2}\beta - 2\alpha \geq 0$$

then one can factorize N_1, N_2 in polynomial time, under the assumption that one can retrieve the integer solutions of equation 8.

4 Generalization of Coppersmith's multivariate method to a system of polynomial equations

We present in this section how we generalized Coppersmith's method for finding integer roots of a multivariate polynomial to a system of polynomial equations.

The problem is as follows. We are given k algebraically independent polynomials $f_1, \ldots, f_k \in \mathbb{Z}[x_1, \ldots, x_n]$. Our goal is to find efficiently all integer solutions $(x_{1,0}, \ldots, x_{n,0})$ of the following system of equations:

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \qquad \vdots \\ f_k(x_1, \ldots, x_n) = 0 \end{cases} \tag{10}$$

satisfying $\forall i, |x_{i,0}| < X_i$, where the X_i's are bounds that we want to maximize. We furthermore assume that the f_i's are irreducible and that the GCD of the coefficients of each f_i is 1.

4.1 Description of our method

Our method is a generalization of Coron and May's formulation of Coppersmith method for multivariate polynomial that is described in section 3.1. The outline of the method is very similar to the one of section 3.1, except that we use all the input polynomials $f_1, \ldots, f_k$:

1. Using $f_1, \ldots, f_k$, construct a collection C of polynomials $g_1, \ldots, g_c \in \mathbb{Z}[x_1, \ldots, x_n]$ such that the g_i's vanish modulo R on the integer solutions $(x_{1,0}, \ldots, x_{n,0})$ of the system of polynomial equation satisfying: $\forall i, |x_{i,0}| < X_i$. R is a modulus which is chosen latter.

2. Find $n-k$ new polynomials $(h_1,\ldots,h_{n-k})$ which are linear combinations of the g_i's such that for every i, $|h_i(x_{1,0},\ldots,x_{n,0})|$ is strictly smaller than R (that is, the h_i's vanish on the solutions of the system of equation over the integers) and such that the variety defined by $(f_1,\ldots,f_k,h_1,\ldots,h_{n-k})$ is 0-dimensional. This step is carried out by LLL-reducing a lattice spanned by the coefficient vectors of the polynomials $g_i(x_1X_1,\ldots,x_nX_n)$.

3. Finally, compute all the integer solution of the system of equations within the bounds by enumerating the points of the 0-dimensional variety defined by $(f_1,\ldots,f_k,h_1,\ldots,h_{n-k})$.

As Coppersmith's method, our method is heuristic because we were unable to prove that the method yields a 0-dimensional variety under suitable bounds. We nevertheless give hints latter on how this could be proved for specific shapes of polynomials. A new difficulty is added: as the lattice created in step 2 is no longer constructed from a basis consisting of the row vectors of an upper-triangular matrix, it becomes difficult to evaluate its determinant as a function of the coefficients of the input polynomials $f_1,\ldots,f_k$. These two points imply that the experiments done in the next subsection are very important to know how the method behaves in practice.

Let's explain how C is constructed. Let m be an integer which controls the size of C. Define by S_i the sets of monomials of f_i^{m-1} and by M_i the monomials of f_i^m. Let $S=\bigcup_{i=1}^k S_i$, $M=\bigcup_{i=1}^k M_i$ and l_j be the largest exponent of x_j that appears in S. Let C be the collection defined by all these polynomials:

$$x_1^{i_1}x_2^{i_2}\cdots x_n^{i_n} f_i(x_1,\ldots,x_n)\prod_{j=1}^{n} X_j^{l_j-i_j} \qquad \text{for} \quad x_1^{i_1}x_2^{i_2}\cdots x_n^{i_n}\in S_i \text{ and } 1\le i\le k$$

$$x_1^{i_1}x_2^{i_2}\cdots x_n^{i_n} R \qquad \text{for} \quad x_1^{i_1}x_2^{i_2}\cdots x_n^{i_n}\in M\backslash S$$

Note that all the polynomials of C are defined over M. Finally, R is chosen to be $W\prod_{j=1}^n X_j^{l_j}$ where $W=\max_{i=1}^k \|f_i(x_1X_1,\ldots,x_nX_n)\|_\infty$.

The next step is to build a lattice L, spanned by all the coefficient vectors (over M) of the polynomials $\{g(x_1X_1,\ldots,x_nX_n)\mid g\in C\}$ and to reduce it using LLL. Note that the number of polynomials in C (and thus the number of vectors used to span L) is a bit greater than $|M|$.

In order to derive bounds under which the method works, two major steps are needed. Note that we weren't able to prove any rigorous bound in the end; however, the experimental results in the next section are encouraging and show that the method is of interest in practice. The first step is to ensure the the first $n-k$ vectors from the basis outputted by LLL are small enough so that Howgrave Graham's lemma (lemma 1) apply in order to satisfy the first condition of step 2. Using Theorem 9, we get the following sufficient condition on $\mathrm{Vol}(L)$ (and thus on the X_j's upon which $\mathrm{Vol}(L)$ depends):

$$\mathrm{Vol}(L) < 2^{-\frac{d(d-1)}{4}}\left(\frac{R}{\sqrt{d}}\right)^{d+1-n+k}$$

where $d=|M|$. Ignoring terms that do not depend on the size of the coefficients (which contribute to an asymptotically small error term), we get the following condition:

$$\mathrm{Vol}(L) < W\prod_{j=1}^{n} X_j^{l_j(d+1-n+k)}$$

Deriving bounds on the X_j would require to express $\mathrm{Vol}(L)$ in terms of the X_j's and the coefficients of the f_i's. Unfortunately, it turns out to be a difficult task, as the lattice L is no longer defined by an upper-triangular basis, due to the presence of several polynomials as inputs.

The second step would be to ensure that $f_1, \ldots, f_k, h_1, \ldots, h_{n-k}$ are algebraically independent. As in Coppersmith's original method in the case $n \geq 3$, we were not able to prove it. We nevertheless give some hints and ideas how this could be done. We observed in our experiments presented in the next section that this is the most restrictive condition to fulfill. The following lemma gives a criterion so that a new polynomial is algebraically independent from all the previous ones taken as a whole:

Lemma 3 (A proof can be found in [2]). Let $p_1, \ldots, p_l$ be polynomials of $\mathbb{Z}[x_1, \ldots, x_n]$ sharing a common root $(x_{1,0}, \ldots, x_{n,0})$. Define $I = \langle p_1, \ldots, p_{l-1} \rangle$. Suppose that I is a prime ideal and that $p_l \notin I$. Then, $p_1, \ldots, p_l$ are algebraically independent. In other words, the dimension of I is decremented by 1 if we add p_l to I.

Consider the special case where $k = n - 1$ (that is, the variety defined by $f_1, \ldots, f_k$ has dimension 1). Let $I = \langle f_1, \ldots, f_k \rangle$. We can suppose that I is a prime ideal, for otherwise we can replace it by one of the radical of the ideals in its primary decomposition. We can furthermore assume that $f_1, \ldots, f_k$ is a Gröbner basis for some monomial order $\prec$ as it is often the case in practice and it gives us a lead to tackle the problem. Assume also that M is compatible with the monomial ordering $\prec$ (see the definition in the preliminaries). The previous lemma tells us that the method of this section will succeed if the generated polynomial h_1 is not in I. Thanks to the assumption made, $h_1 \in I$ translates into the existence of polynomials $a_i \in \mathbb{Z}[x_1, \ldots, x_n]$ such that:

$$h_1 = a_1 f_1 + \cdots + a_k f_k \quad \text{and the } a_i f_i\text{'s are defined over } M$$

Note that it is the fact the $f_1, \ldots, f_k$ is a Gröbner basis for $\prec$ and that M and $\prec$ are compatible which guarantees that the $a_i f_i$'s are defined over M. This conveniently makes finite the set of monomials to which the a_i's belong. One way to obtain rigorous bounds in this case would thus to prove an analog to Hinek Stinson's lemma for multiple polynomials that would solve the following problem:

Problem 1. *Let $f_1, \ldots, f_k$ be polynomials of $\mathbb{Z}[x_1, \ldots, x_n]$, and M be a set of monomials. Suppose that:*

$$g = a_1 f_1 + \cdots + a_k f_k$$

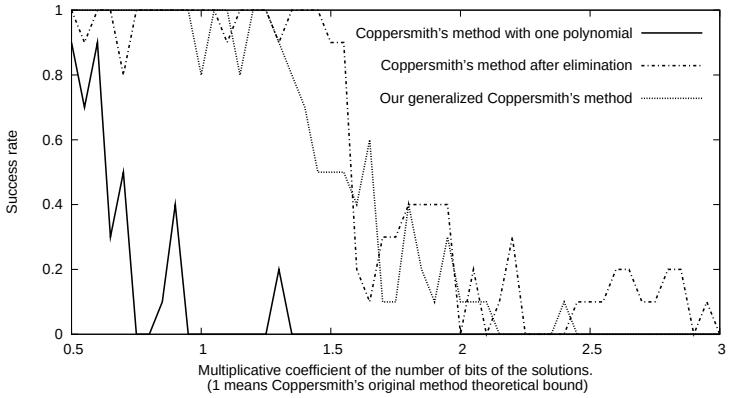
with the a_i being in $\mathbb{Z}[x_1, \ldots, x_n]$ and such that the $a_i f_i$'s are defined over M. Lower-bound *the norm of g as a function of the f_i's and M. We're especially interested in lower-bounds that increase as the coefficients of the f_i's increase.*

Note that we aren't aware of anybody having studied this problem in the general case $k \geq 2$. The major difficulty in order to come up with a lower-bound is the cancellation of the coefficients in front of monomials in the sum $a_1 f_1 + \cdots a_k f_k$. Besides, any useful lower-bound would heavily depend upon the shape of the f_i's. Let's take a trivial example. Suppose that $k = 2$, $f_1 = ax_1 + 1$ and $f_2 = (a+1)x_2 - 1$, where a is a big integer, and $M = \{x_1^2, x_2^2, x_1 x_2, x_1, x_2, 1\}$. Then $g = f_2 - f_1$ has arbitrarily tiny coefficients compared to a.

4.2 Experimental results

We tested our method with $n = 3$ and $n = 4$ (that is, 3 and 4 variables), and two or three polynomials $f_1, f_2, (f_3)$ vanishing on a common root as input. Indeed, we compared it to three other methods:

Table 2: Success rate of the different methods in the case $n = 3$, $k = 2$ and rectangular polynomials of degree 1 in each variable.



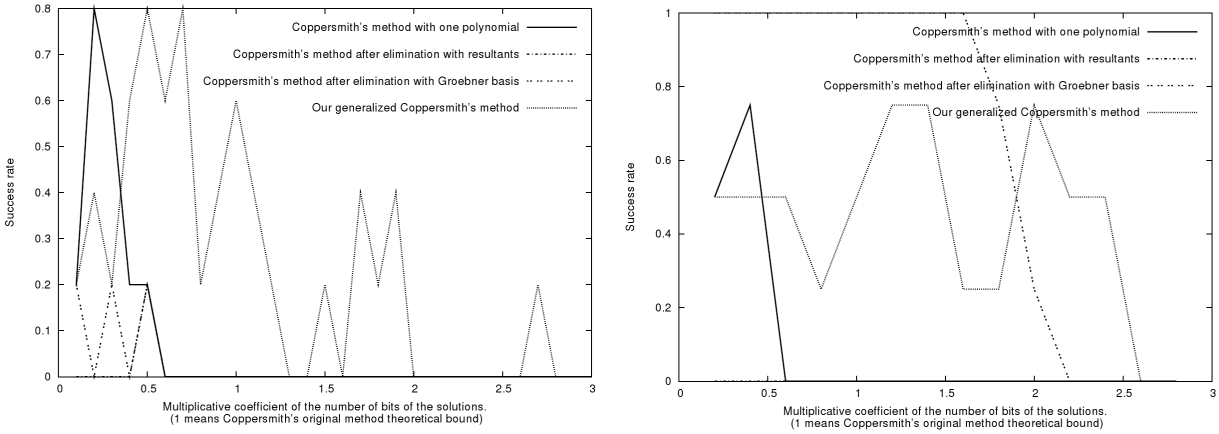
1. Use only one polynomial out of the two or three, and use the original Coppersmith's method described in section 3.

2. Eliminate one variable computing the resultant $g = \mathrm{Res}(f_1, f_2, x_1)$, and apply original Coppersmith's method on g. If we have three polynomials as input, we eliminate two variables computing $g = \mathrm{Res}(\mathrm{Res}(f_1, f_2, x_1), \mathrm{Res}(f_1, f_3, x_1), x_2)$. Note that there are several competing effects that increase or decrease the bounds on the roots that can be recovered. Decreasing the number of variable and g having greater coefficients than f_1 and f_2 improves the bounds, whereas g having a higher degree worsens the bounds. All in all, we can predict (by estimating experimentally the size of the coefficients of a resultant in term of the input polynomials) that computing the resultant increases the theoretical bounds on the solutions by factors between 2 and 6 (depending on the shape of the polynomials).

3. Eliminate $k - 1$ variables using Gröbner basis for a lexicographical ordering on the monomials and do the same as above.

We performed experiments for rectangular polynomials with maximum degree 1 in each variable, and lower-triangular polynomials with total degree 2. Random polynomials were generated with random 200-bit-sized coefficients, with the leading term being always non-zero, and a third of the remaining terms being zero, in order to somehow reproduce real-world instances. We carried out tests for various size of solutions. For a specific size, we carried out 5 tests, and called a test successful when the resulting variety was 0-dimensional (which is the strongest notion of success for those methods). We couldn't do more tests because of the time needed for one individual test to complete (several days). That explains partly the irregularities of the graphs. We show in figures 2 and 3 the success rate of the four methods as a function of the size of the roots of the polynomials. (when the two methods of elimination were the same, we show only one curve). In all this graphs, the x-axis represents the size (in number of bits) of the roots as a multiplicative factor of the theoretical bound of Coppersmith's original method using one polynomial (given by inequalities 6 and 7), and the y-axis represents the success rate. For instance, we can read in figure 3 (on the left) that our method has 20% success rate when the roots are 1.5 times bigger (in number of bits) than the theoretical bound of Coppersmith's original method applied to rectangular polynomials of degree 1 in each variable. Other graphs that do not give much additional useful information are shown in appendix B.

Table 3: Success rate of the different methods in the case $n = 4$, $k = 3$. Rectangular polynomials of degree 1 in each variables on the left, and triangular polynomials of total degree 2 on the right.



The methods involved LLL-reducing lattices of sizes between 100 and 400, and each individual test took up to a couple of days on Intel Xeon X5xxx or X7xxx processors belonging to the LIP6 computer cluster. We used Stehlé's fast implementation of LLL described in [23].

Overall, eliminating variables with Gröbner basis computation performs often much better than Coppersmith's method with one polynomial or after eliminating variables using resultants. With 3 variable and 2 polynomials as input, our generalized Coppersmith method performs a bit worse than Coppersmith's original method after having eliminated a variable. However, it can still be useful as it solves instances (15 out of the 35 cases where our method works, between the abscissas 1.5 and 2.1) that are unsolvable by all other methods (this is not shown on the graph). With 4 variables and 3 polynomials our method performs better (but not in all cases) than the aformentioned methods using 3 polynomials. In particular, in the rectangular case (figure 3, left side), our method performs well compared to other methods that uses 3 polynomials. On the other hand, it should be pointed out that our results show that using only two polynomials out of the three available (throwing out the third one for instance) surprisingly performs better than using directly the three polynomials.

As a conclusion, none of the methods using more than 2 polynomials were conclusive. One should instead use only two polynomials out of the k available. When using 2 polynomials, our method can in some cases solve instances that cannot be solved by existing methods. One should therefore always try all the methods in order to solve as many instances as possible.

Recall that there can be two reasons why our method fails on a particular instance. It can be either because the reduced basis is not short enough (and thus the associated polynomials do not vanish over the integer), or the new polynomials are not algebraically independent. It appeared in our experiments that failure was always due to the second reason.

Finally, even though all methods using more than two polynomials do not give good experimental results in the generic cases we tested, we still tried to apply our method the the implicit factoring problem in the next section. It turned out that this lead to surprisingly good results which we fully explain.

4.3 Application to the implicit factoring problem

Our main goal in this section is to generalize Sarkar and Maitra's ideas (described in section 3.2) to solve the implicit factoring problem to an arbitrary number k of moduli, by using our method to find integer roots of a system of integer polynomials. As we'll see, it turned out that our method performed well, but for an unforeseen reason. Understanding this allowed us to devise in section 5 a novel and direct lattice-based method that addresses the case where the p_i's share MSBs.

We present how we applied our method for finding integer roots of polynomial systems to the problem of implicit factoring. Consider three RSA moduli $N_i = p_i q_i$, $1 \leq i \leq 3$, with the p_i's all sharing t MSBs. As in section 3.2, we let p represent the t shared bits of the p_i's and write $p_i = p + \tilde{p_i}$ for all $1 \leq i \leq 3$. We thus have the equalities $N_i = q_1(p + \tilde{p_i})$. Let's turn the q_i's,$\tilde{p_i}$'s,p into indeterminates and introduce the following polynomials:

$$\begin{aligned} w_1(x_1, x_2, x_3, x_4, x_5, x_6, x_7) &= x_2(x_1 + x_3) - N_1 \\ w_2(x_1, x_2, x_3, x_4, x_5, x_6, x_7) &= x_4(x_1 + x_5) - N_2 \\ w_3(x_1, x_2, x_3, x_4, x_5, x_6, x_7) &= x_5(x_1 + x_7) - N_3 \end{aligned}$$

that vanish simultaneously in $(p, q_1, \tilde{p_1}, q_2, \tilde{p_2}, q_3, \tilde{p_3})$. Using Gröbner basis computations with a lexicographical ordering, we eliminate x_1, the variable representing p. Note that this is what Sarkar and Maitra already do in their paper, though with only two polynomials and manually. After elimination, we get three polynomials f_1, f_2, f_3 with 6 variables whose common root $(q_1, \tilde{p_1}, q_2, \tilde{p_2}, q_3, \tilde{p_3})$ reveals the factorization of the N_i's. We have thus reduced the implicit factoring problem with several RSA moduli to computing integer solutions of a polynomial system of equations. As the f_i's had no constant term, we substituted the x_i's by $x_i + 1$. We can apply our method we described in the previous sections.

Experimental results We ran our method with the polynomials f_1, f_2, f_3 as input. We randomly chose three unbalanced RSA moduli $N_i = p_i q_i$ with $n = 200$ bits, with the q_i being $\alpha = 50$ bits primes, and the p_i's sharing from $t = 120$ to $t = 60$ MSBs. Remember that Sarkar and Maitra heuristic method requires theoretically $t = 105$ shared MSBs in this cases (using Theorem 2), and that $t = 91$ is enough in their experiments. That led to lattices of dimension roughly 400 each reduced in a couple of days.

Define the ideal $I = \langle f_1, f_2, f_3 \rangle$. We first observed that the new polynomials h_i's generated by the method were always in I, which is apparently a failure. Nevertheless, we discovered that the new polynomials often contained a coefficient whose GCD with the N_i was not trivial. We were able in those cases to factorize all the N_i's. Surprisingly, we discovered that our method allowed us always to factorize the N_i's when the p_i shared at least 78 bits (which is significantly better than Sarkar and Maitra's experimental results), and that the method almost never worked when then p_i's shared less than 78 bits. This sharp cut-off certainly calls for a detailed explanation.

After examination, the input polynomials had this form:

$$\begin{aligned} f_1(x, y, z, t, u, v) &= xyz + xy - xzt - xt + p_2 q_2 x + yz + y - zt - p_1 q_1 z - t + (p_2 q_2 - p_1 q_1) \\ f_2(x, y, z, t, u, v) &= xyu + xy - xuv - xv + p_3 q_3 x + yu + y - uv - p_1 q_1 u - v + (p_3 q_3 - p_1 q_1) \\ f_3(x, y, z, t, u, v) &= ztu + zt - zuv - zv + p_3 q_3 z + tu + t - uv - p_2 q_2 u - v + (p_3 q_3 - p_2 q_2) \end{aligned}$$

Besides, we observed that the new polynomials h_i found by the method were not only in I, but linear combinations with *integer coefficients* of the f_i. We discovered that this linear short combination was always the same (for the "interesting" polynomials that yield a non-trivial factor of each N_i, and up to a multiplicative constant which is a by-product of our method):

$$h = q_3 f_1 - q_2 f_2 + q_1 f_3$$

Polynomials of this type lead immediately to the factorization of the N_i's, and always corresponded to a shortest vector of the LLL-reduced basis. Besides, we observed that the corresponding short vector was significantly smaller (when the number of shared bits was not too close to our experimental bound $t = 78$) than what the Gaussian heuristic predicted (see lemma 6), and that the estimated gap of the lattice (see definition 1) was unusually high. Let's examine the coefficients of h to see what happens. For instance, the coefficient in front of x is $q_2 q_3(p_2 - p_3)$ and the constant term is $q_1 q_3(p_3 - p_1) + q_1 q_2(p_1 - p_2)$. Indeed, all the coefficients of h are either of the type "$q_i q_j(p_i - p_j)$" or a sum of q_i's. Note how the relation $q_i N_j - q_j N_i = q_i q_j(p_j - p_i)$ harnesses the cancellation of the shared MSBs of the p_i. Thanks to this cancellation, the large coefficients of h have at most $2\alpha + n - t$ bits, which explains the fact that the associated vector of the lattice is unusually short when t is big enough.

All of this suggests that the shift polynomials are not used at all during the reduction of the lattice, and that we could obtain the same results much more efficiently by reducing directly the right sub-lattice. This is what we achieved after some thinking; we present our results in the next section.

5 Direct lattice-based method for the implicit factoring problem

The results described in this section constitute the main results of the internship and led to the writing of an article.

5.1 Implicit Factoring of Two RSA Moduli

As usual, we are given two n-bit RSA moduli $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$, where q_1 and q_2 are α-bit primes, given only the implicit hint that p_1 and p_2 share t most significant bits (MSBs) that are *unknown* to us.

The experiments of the previous section gave us the idea to craft a lattice whose shortest vector would harness the short linear combination $q_1 N_2 - q_2 N_1 = q_1 q_2(p_2 - p_1)$ of N_1 and N_2. The algebraic relation $q_1 N_2 - q_2 N_1 = q_1 q_2(p_2 - p_1)$ is specially interesting to craft a shortest vector that would reveal non-trivial factors of the N_i's because the shared most significant bits of p_1 and p_2 cancel out and thus $q_1 q_2(p_2 - p_1)$ has approximately $\alpha + n - t$ bits. It can therefore be made quite small compared to $N_1 \approx N_2 \approx 2^n$ for t large enough.

Reducing the row vectors of $\begin{pmatrix} N_1 \\ N_2 \end{pmatrix}$ would still however be non-sense and "lead" to the trivial relation $N_2 N_1 - N_1 N_2 = 0$. In order to control the size of the coefficients in front of N_1 and N_2 in a shortest vector, we concatenated the matrix $\begin{pmatrix} K & 0 \\ 0 & K \end{pmatrix}$ to the previous one, and chose $K \approx 2^{n-t}$ in order to "penalize"

coefficients in front of N_1, N_2 greater than the $q_i \approx 2^\alpha$. A basis of our lattice is now the row vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ of

$$\begin{pmatrix} K & 0 & N_1 \\ 0 & K & N_2 \end{pmatrix}$$

and we hope that $\mathbf{v_0} = q_2\mathbf{v_1} - q_1\mathbf{v_2} = (q_1K, q_2K, q_1q_2(p_1 - p_2))$ whose coefficients are all roughly $(n + \alpha - t)$-bit integers is a shortest vector of the lattice.

We now present rigorously our results and show that N_1 and N_2 can be factored in quadratic time as soon as $t \geq 2\alpha + 3$. By saying that the primes p_1, p_2 of maximal bit-size $n - \alpha + 1$ share t MSBs, we really mean that $|p_1 - p_2| \leq 2^{n-\alpha-t+1}$.

Let's consider the lattice L spanned by the row vectors (denoted by $\mathbf{v_1}$ and $\mathbf{v_2}$) of the following matrix:

$$\begin{pmatrix} K & 0 & N_2 \\ 0 & K & -N_1 \end{pmatrix} \quad \text{where } K = \lfloor 2^{n-t+\frac{1}{2}} \rfloor$$

and consider the following vector of L: $\mathbf{v_0} = q_1\mathbf{v_1} + q_2\mathbf{v_2} = (q_1K, q_2K, q_1q_2(p_2 - p_1))$.

Notice that the shared MSBs of p_1 and p_2 cancel each other out in the difference $p_2 - p_1$. Each of the coefficients of $\mathbf{v_0}$ are thus integers of roughly $(n + \alpha - t)$ bits. Provided that t is sufficiently large, $\pm\mathbf{v_0}$ may be a shortest vector of L that can be found using Lagrange reduction on L. Moreover, note that as soon as we retrieve $\mathbf{v_0}$ from L, factoring N_1 and N_2 is easily done by dividing the first two coordinates of $\mathbf{v_0}$ by K. Proving that $\mathbf{v_0}$ is a shortest vector of L under some conditions on t is therefore sufficient to factorize N_1 and N_2.

We first give an intuition on the bound on t that we can expect, and we give after that a proof that $\mathbf{v_0}$ is indeed a shortest vector of L under a similar condition.

The volume of L is the square root of the determinant of G, the Gramian matrix of L:

$$G = \begin{pmatrix} K^2 + N_2^2 & -N_1N_2 \\ -N_1N_2 & K^2 + N_1^2 \end{pmatrix}$$

That is,

$$\mathrm{Vol}(L) = K\sqrt{N_1^2 + N_2^2 + K^2} \approx 2^{2n-t} \tag{11}$$

because $K^2 \approx 2^{2(n-t)}$ is small compared to the $N_i^2 \approx 2^{2n}$.

The norm of $\mathbf{v_0}$ is approximately $2^{n+\alpha-t}$, because each of its coefficients have roughly $n+\alpha-t$ bits. If $\mathbf{v_0}$ is a shortest vector of L, it must be smaller than the Minkowski bound applied to L: $2^{n+\alpha-t} \approx \|\mathbf{v_0}\| \leq \sqrt{2}\,\mathrm{Vol}(L)^{1/2} \approx 2^{n-t/2}$, which happens when $t \geq 2\alpha$.

The following theorem states that $\mathbf{v_0}$ is indeed a shortest vector of L under a similar condition on t.

Theorem 3. *Let $N_1 = p_1q_1, N_2 = p_2q_2$ be two n-bit RSA moduli, where the q_i's are α-bit primes and the p_i's are primes that share t most significant bits. If $t \geq 2\alpha+3$, then N_1 and N_2 can be factored in quadratic time in n.*

Proof. Let L be the lattice generated by the row vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ of the following matrix:

$$\begin{pmatrix} K & 0 & N_2 \\ 0 & K & -N_1 \end{pmatrix} \quad \text{where } K = \lfloor 2^{n-t+\frac{1}{2}} \rfloor$$

and let

$$\mathbf{v_0} = q_1\mathbf{v_1} + q_2\mathbf{v_2} = (q_1K, q_2K, q_1q_2(p_2 - p_1))$$

Our goal is to prove that $\pm\mathbf{v_0}$ is a shortest vector of the lattice L.

Let $(\mathbf{b_1}, \mathbf{b_2})$ be the resulting basis from the Lagrange reduction on L. Computing $(\mathbf{b_1}, \mathbf{b_2})$ takes a quadratic time in n, as the norms of $\mathbf{v_1}$ and $\mathbf{v_2}$ are bounded by 2^{n+1}. This reduced basis verifies $\|\mathbf{b_1}\| = \lambda_1(L), \|\mathbf{b_2}\| = \lambda_2(L)$, and, by Hadamard's inequality:

$$\|\mathbf{b_1}\|\|\mathbf{b_2}\| \geq \mathrm{Vol}(L) \tag{12}$$

As $\mathbf{v_0}$ is in the lattice, $\|\mathbf{b_1}\| = \lambda_1(L) \leq \|\mathbf{v_0}\|$. Using (12), we get $\|\mathbf{b_2}\| \geq \frac{\mathrm{Vol}(L)}{\|\mathbf{v_0}\|}$. Moreover, if $\mathbf{v_0}$ is strictly shorter that $\mathbf{b_2}$, $\mathbf{v_0}$ is a multiple of $\mathbf{b_1}$; for otherwise $\mathbf{b_2}$ would not be the second minimum of the lattice. In this case, $\mathbf{v_0} = a\mathbf{b_1} = a(b\mathbf{v_1} + c\mathbf{v_2}), a, b, c \in \mathbb{Z}$, and looking at the first two coefficients of $\mathbf{v_0}$, we get that $ab = q_1$ and $ac = q_2$. Since the q_i's are prime, we conclude that $a = \pm 1$, that is, $\mathbf{v_0} = \pm\mathbf{b_1}$. Using the previous inequality, a condition for $\mathbf{v_0}$ to be strictly shorter than $\mathbf{b_2}$ is:

$$\|\mathbf{v_0}\|^2 < \mathrm{Vol}(L) \tag{13}$$

Let's upper-bound the norm of $\mathbf{v_0}$ and lower-bound $\mathrm{Vol}(L)$. We first provide simple bounds that proves the theorem when $t \geq 2\alpha + 4$ and derive secondly tighter bounds that require only $t \geq 2\alpha + 3$.

The p_i's have at most $n - \alpha + 1$ bits, and they share their t most significant bits so $|p_2 - p_1| \leq 2^{n-\alpha+1-t}$. We thus have the following inequality on the norm of $\mathbf{v_0}$:

$$\begin{aligned} \|\mathbf{v_0}\|^2 &\leq 2^{2(n-t)+1}(q_1^2 + q_2^2) + q_1^2q_2^2(p_1 - p_2)^2 \\ &\leq 2^{2(n+\alpha-t)+2} + 2^{2(\alpha+n+1-t)} \\ &\leq 2^{2(n+\alpha-t)+3} \end{aligned} \tag{14}$$

We can lower-bound the expression (11) for the volume of L, using that $N_1, N_2 \geq 2^{n-1}$ and $K^2 \geq 2^{2(n-t)}$:

$$\begin{aligned} \mathrm{Vol}(L)^2 &= K^2(N_1^2 + N_2^2 + 2^{2(n-t)}) \\ &> 2^{4n-2t-1} \end{aligned} \tag{15}$$

Using inequalities (14) and (15), (13) is true provided that:

$$2^{2(n+\alpha-t)+3} \leq 2^{2n-t-\frac{1}{2}}$$

which is equivalent to (as t and α are an integers):

$$t \geq 2\alpha + 4 \tag{16}$$

We have thus proved the theorem under condition 16.

We now refine the bounds on $\|\mathbf{v_0}\|$ and $\mathrm{Vol}(L)$ in order to prove the tight case.

q_1 and q_2 are α-bit primes, therefore $\forall i, q_i \leq 2^\alpha - 1$. Define ϵ_1 by $2^\alpha - 1 = 2^{\alpha-\epsilon_1}$. We get:

$$\forall i, q_i^2 \leq 2^{2\alpha-2\epsilon_1} \tag{17}$$

Moreover,

$$K^2 \leq 2^{2(n-t)+1} \tag{18}$$

because $K = \lfloor 2^{n-t+\frac{1}{2}} \rfloor$. We can therefore upper-bound $K^2 q_i^2$ using the inequalities (17) and (18):

$$\forall i, K^2 q_i^2 \leq 2^{2(n-t+\alpha)+1-2\epsilon_1} \tag{19}$$

The p_i's have at most $n - \alpha + 1$ bits and they share t bits, so $(p_2 - p_1)^2 \leq 2^{2(n-\alpha+1-t)}$. Using inequality (17), we can bound $q_1^2 q_2^2 (p_2 - p_1)^2$:

$$q_1^2 q_2^2 (p_2 - p_1)^2 \leq 2^{2(n-t+\alpha+1-2\epsilon_1)} \tag{20}$$

We can finally bound the norm of $\mathbf{v_0}$ using (20) and (19):

$$\begin{aligned} \|\mathbf{v_0}\|^2 &= K^2(q_1^2 + q_2^2) + q_1^2 q_2^2 (p_2 - p_1)^2 \\ &\leq 2^{2(n+\alpha-t)+2-2\epsilon_1} + 2^{2(n-t+\alpha+1-2\epsilon_1)} \\ &\leq 2^{2(n+\alpha-t)+3-\epsilon_1} \end{aligned} \tag{21}$$

Let's define ϵ_2 by the equality $2^{n-t+1/2} - 1 = 2^{n-t+1/2-\epsilon_2}$. We have that $K = \lfloor 2^{n-t+\frac{1}{2}} \rfloor \geq 2^{n-t+1/2-\epsilon_2}$ and $N_i^2 \geq 2^{2n-2}$. We can therefore lower-bound $\mathrm{Vol}(L)^2$:

$$\begin{aligned} \mathrm{Vol}(L)^2 &= K^2(N_1^2 + N_2^2 + 2^{2(n-t)}) \\ &> K^2(N_1^2 + N_2^2) \\ &> 2^{4n-2t-2\epsilon_2} \end{aligned} \tag{22}$$

Using the inequalities (21) and (22), the condition (13) is true under the new condition that:

$$2^{2(n+\alpha-t)+3-\epsilon_1} \leq 2^{2n-t-\epsilon_2}$$

that is:

$$t \geq 2\alpha + 3 + \epsilon_2 - \epsilon_1$$

It remains to show that $\epsilon_2 \leq \epsilon_1$. This comes from the fact that $\epsilon_1 = \log_2(\frac{1}{1-\frac{1}{2^\alpha}})$ and $\epsilon_2 = \log_2(\frac{1}{1-\frac{1}{2^{n-t+\frac{1}{2}}}})$, and that $\alpha \leq n - t$. □

Remark 1. For our analysis, the value $K = \lfloor 2^{n-t+\frac{1}{2}} \rfloor$ is indeed the best possible value. If we use $K = \lfloor 2^{n-t+\gamma} \rfloor$, we obtain the bound $t \geq 2\alpha + f(\gamma)$ where

$$f(\gamma) = \frac{3}{2} - \gamma + \log_2(2 + 2^{2\gamma})$$

The minimum of f is 3 and is attained in $\gamma = \frac{1}{2}$.

5.2 Implicit Factoring of k RSA Moduli

The construction of the lattice for 2 RSA moduli naturally generalizes to an arbitrary number k of moduli. Similarly, we show that a short vector $\mathbf{v_0}$ of the lattice allows us to recover the factorization of the N_i's. This vector takes advantage of the relations $q_iN_j - q_jN_i = q_iq_j(p_j - p_i)$ for all $i, j \in \{1, \ldots, k\}$. However, we were unable to prove that $\mathbf{v_0}$ is a shortest vector of the lattice. Therefore, our method relies on the Gaussian heuristic to estimate the conditions under which $\mathbf{v_0}$ should be a shortest vector of the lattice. Experimental data in the next section confirms that this heuristic is valid in nearly all the cases.

In this section, we are given k RSA moduli of n bits: $N_i = p_iq_i$, $1 \leq i \leq k$, where the q_i's are α-bit primes and the p_i's are primes that all share t most significant bits.

Let us construct a matrix M whose row vectors will form a basis of a lattice L. M will have k rows and $k + \binom{k}{2} = \frac{k(k+1)}{2}$ columns. Denote by $s_1, \ldots, s_m$ with $m = \binom{k}{2}$ all the subsets of cardinality 2 of $\{1, 2, \ldots, k\}$. To each of the s_i's, associate a column vector $\mathbf{c_i}$ of size k the following way. Let a,b be the two elements of s_i, with $a < b$. We set the a-th element of $\mathbf{c_i}$ to N_b, the b-th element of $\mathbf{c_i}$ to $-N_a$, and all other elements to zero. Finally, form M by concatenating column-wise the matrix $KI_{k\times k}$, where $I_{k\times k}$ is the identity matrix of size k, along with the m column vectors $\mathbf{c_1}, \ldots, \mathbf{c_m}$. K is chosen to be $\lfloor 2^{n-t+\frac{1}{2}} \rfloor$. We will call $\mathbf{v_1}, \ldots, \mathbf{v_k}$ the row vectors of M.

To make things more concrete, consider the example of $k = 4$. Up to a reordering of the columns (that changes nothing to the upcoming analysis),

$$M = \begin{pmatrix} K & 0 & 0 & 0 & N_2 & N_3 & N_4 & 0 & 0 & 0 \\ 0 & K & 0 & 0 & -N_1 & 0 & 0 & N_3 & N_4 & 0 \\ 0 & 0 & K & 0 & 0 & -N_1 & 0 & -N_2 & 0 & N_4 \\ 0 & 0 & 0 & K & 0 & 0 & -N_1 & 0 & -N_2 & -N_3 \end{pmatrix} \text{ where } K = \lfloor 2^{n-t+\frac{1}{2}} \rfloor \qquad (23)$$

Notice that the columns $k + 1$ to $k + m$ correspond to all the 2-subsets of $\{1, 2, 3, 4\}$.

Similarly to the case of 2 RSA moduli, L contains a short vector that allows us to factorize all the N_i's (by dividing its first k coordinates by K for instance):
Lemma 4. Let $\mathbf{v_0} = q_1\mathbf{v_1} + \cdots + q_k\mathbf{v_k}$. Then $\mathbf{v_0}$ can be rewritten as follows:

$$\mathbf{v_0} = (q_1K, \ldots, q_kK, \underbrace{\ldots, q_aq_b(p_b - p_a), \ldots}_{\forall \{a,b\} \subset \{1,\ldots,k\}})$$

Assumption 1. *If $\pm\mathbf{v_0}$ is shorter than the Gaussian heuristic applied to L: $\lambda_1(L) \approx \sqrt{\frac{k}{2\pi e}} \operatorname{Vol}(L)^{\frac{1}{k}}$ then it is a shortest vector of L.*

This assumption is backed by experimental data in the next section. We found it to be almost always true in practice. This condition can be seen as an analog of condition 13 of section 5.1 in the case of two RSA moduli.

Let's derive a bound on t so that $\mathbf{v_0}$ is smaller than the Gaussian heuristic applied to L. The norm of $\mathbf{v_0}$ can be computed and upper-bounded easily:

$$\|\mathbf{v_0}\|^2 = K^2 \left(\sum_{i=1}^{k} q_i^2 \right) + \sum_{\{i,j\} \subset \{1,\ldots,k\}} q_i^2 q_j^2 (p_i - p_j)^2 \leq k^2 2^{2(n+\alpha-t)+1}$$

Computing the volume of L is a bit more involved, we refer to lemma 6 of appendix C:

$$\mathrm{Vol}(L) = K\left(K^2 + \sum_{i=1}^{k} N_i^2\right)^{\frac{k-1}{2}} \geq 2^{n-t}\left(\sqrt{k}2^{n-1}\right)^{k-1}$$

We now seek the condition on t for the norm of $\mathbf{v_0}$ to be smaller than the Gaussian heuristic. Using the two previous inequalities, $\mathbf{v_0}$ is smaller than the Gaussian heuristic provided that:

$$t \geq \frac{k}{k-1}\alpha + 1 + \frac{k}{2(k-1)}\left(2 + \frac{\log_2(k)}{k} + \log_2(\pi e)\right) \tag{24}$$

When $k \geq 3$, we can derive the simpler and stricter bound: $t \geq \frac{k}{k-1}\alpha + 6$.

Finally, as $\pm\mathbf{v_0}$ is now a shortest vector of L under Assumption 1, it can be found in time $\mathcal{C}(k, \frac{k(k+1)}{2}, n)$ where $\mathcal{C}(k, s, B)$ is the time to find a shortest vector of a k-dimensional lattice of $\mathbb{Z}^s$ given by B-bit basis vectors. We just proved the following theorem:

Theorem 4. *Let $N_1 = p_1q_1, \ldots, N_k = p_kq_k$ be k n-bit RSA moduli, with the q_i's being α-bit primes, and the p_i's being primes that all share t most significant bits. Under Assumption 1, the N_i's can be factored in time $\mathcal{C}(k, \frac{k(k+1)}{2}, n)$, as soon as t verifies equation* (24)*.*

Remark 2. Note that we can find a shortest vector of the lattice of Theorem 4 using Kannan's algorithm (Theorem 8) in time $\mathcal{O}(\mathcal{P}(n,k)\,k^{\frac{k}{2e}+o(k)})$ where $\mathcal{P}$ is a polynomial. It implies that we can factorize all $N_1, \ldots, N_k$ in time polynomial in n as soon as k is constant or k^k is a polynomial in n. Unfortunately, to the best of our knowledge,this algorithme is not implemented in the computer algebra system Magma [5] on which we implemented our methods. We used instead Schnorr-Euchner's enumeration which is well known [14, 15] to perform well beyond small dimension (≤ 50). For example, we found that finding a shortest vector of the lattice using Schnorr-Euchner's algorithm takes less than 1 minute for $k \leq 40$. One may also reduce the lattice using LLL algorithm instead of Schnorr-Euchner's. If t is not too close to the bound of Theorem 4, the Gaussian heuristic suggests that the gap of the lattice is large, and thus LLL may be able to find a shortest vector of L even in medium dimension (50–200).

Similarly to the case of 2 RSA moduli, $K = \lfloor 2^{n-t+\frac{1}{2}} \rfloor$ is optimal for our analysis. Indeed, if we redo the analysis with $K = \lfloor 2^{n-t+\gamma} \rfloor$, we find that the optimal value for γ is the one that minimizes the function $f_k = \gamma \mapsto \frac{1}{2}k\log_2(k - 1 + 2^{2\gamma-1}) - \gamma$, which is $\gamma = \frac{1}{2}$ regardless of k.

5.3 Experimental results

In order to check the validity of Assumption 1 and the quality of our bounds on t, we implemented the method on Magma 2.15 [5]. We generated many random 1024-bit RSA moduli, for various values of α and t. We observed that the results were similar for other values of n. In the case where $k = 2$, we used the Gaussian reduction to find with certainty a shortest vector of the lattice, and for $3 \leq k \leq 40$ we compared Schnorr-Euchner's algorithm (that provably outputs a shortest vector of the lattice) with LLL (that gives an exponential approximation of a shortest vector). We used only LLL for $k = 80$.

We conducted experiments for $k = 2, 3, 10, 40$ and 80, and for several values for α. For specific values of k, α and t, we said that a test was successful when the first vector of the reduced basis of the lattice was

of the form $\pm\mathbf{v_0}$ (that is, it satisfies Assumption 1 in the heuristic case $k \geq 3$). For each k and each α, we generated 100 tests and found experimentally the best (lowest) value of t that had 100% success rate. We compared this experimental value to the bounds we obtained in theorems 4 and 3. For the first value of t that doesn't have 100% success rate and for $k \geq 3$, we analyzed the rate of failures due to Assumption 1 not being valid. Note that failures can be of two different kinds: the first possibility is that $\|\mathbf{v_0}\|$ is greater than the Gaussian heuristic, and the second one is that $\|\mathbf{v_0}\|$ is smaller than the Gaussian heuristic yet $\mathbf{v_0}$ is not a shortest vector of the lattice (that is, Assumption 1 does not hold). We wrote down the percentage of the cases where Assumption 1 was not valid among all the cases where $\|\mathbf{v_0}\|$ was smaller than the Gaussian heuristic. These results are shown in tables 4 to 8. Let's take an example. For $k = 10$ and $\alpha = 200$ (second line of table 6), Theorem 4 predicts that $\mathbf{v_0}$ is a shortest vector of the lattice as soon as $t \geq 227$. It turned out that it was always the case as soon as $t \geq 225$, which is better than expected. For $t = 224$, Assumption 1 was not valid in 3% of the cases.

Let's analyze the results now. In the rigorous case $k = 2$, we observe that the attack consistently goes one bit further with 100% success rate than our bound in Theorem 3.

In all our experiments concerning the heuristic cases $k \geq 3$, we observed that we had 100% success rate (thus, Assumption 1 was always true) when t was within the bound (24) of Theorem 4. That means that Theorem 4 was always true in our experiments. Moreover, we were often able to go a few bits (up to 3) beyond the theoretical bound on t. When the success rate was not 100% (that is, beyond our experimental bounds on t), we found that Assumption 1 was not true in a very limited number of the cases (less than 3%). Finally, up to dimension 80, LLL was always sufficient to find $\mathbf{v_0}$ when t was within the bound of Theorem 4, and Schnorr-Euchner's algorithm allowed us to go one bit further than LLL in dimension 40.

Additionally, we show in table 9 the lowest value of t with 100% success rate and the running-time of LLL and Schnorr-Euchner's algorithm for several values of k. For each k, we show the worst running-time we encountered when running 10 tests on an Intel Xeon E5420 at 2.5Ghz. We see that all individual tests completed in less than 1 second for $2 \leq k \leq 20$. We used Schnorr-Euchner's algorithm up to $k = 60$ where it took at most 6200 seconds. LLL completes under one minute for $20 \leq k \leq 40$ and in less than 30 minutes for $40 \leq k \leq 80$.

Table 4: Results for $k = 2$ and 1024-bit RSA moduli

α (bit-size of the q_i's)	Bound of Theorem 3 $t \geq 2\alpha + 3$	Best experimental t (number of bits shared among the p_i's)
150	303	302
200	403	402
250	503	502
300	603	602

Acknowledgements

We are indebted to Bernardo Freitas Paulo da Costa and Saskia Bosma for their useful comments and suggestions improving the editorial quality of this article.

Table 5: Results for $k = 3$ and 1024-bit RSA moduli

α (bit-size of the q_i's)	Bound $t \geq \frac{3}{2}\alpha + 5.2\ldots$ of Theorem 4	Best experimental t (number of bits shared among the p_i's) using LLL algorithm	Best experimental t using Kannan's algorithm	Failure rate of Assumption 1
150	231	228	228	0% ($t = 227$)
200	306	303	303	0% ($t = 302$)
250	381	378	378	0% ($t = 377$)
300	456	453	453	0% ($t = 452$)
350	531	528	528	0% ($t = 527$)
400	606	603	603	0% ($t = 602$)

Table 6: Results for $k = 10$ and 1024-bit RSA moduli

α (bit-size of the q_i's)	Bound $t \geq \frac{10}{9}\alpha + 4.01\ldots$ of Theorem 4	Best experimental t (number of bits shared among the p_i's) using LLL algorithm	Best experimental t using Schnorr-Euchner's algorithm	Failure rate of Assumption 1
150	171	169	169	0% ($t = 168$)
200	227	225	225	3% ($t = 224$)
250	282	280	280	3% ($t = 279$)
300	338	336	336	1% ($t = 335$)
350	393	391	391	2% ($t = 390$)
400	449	447	447	0% ($t = 446$)

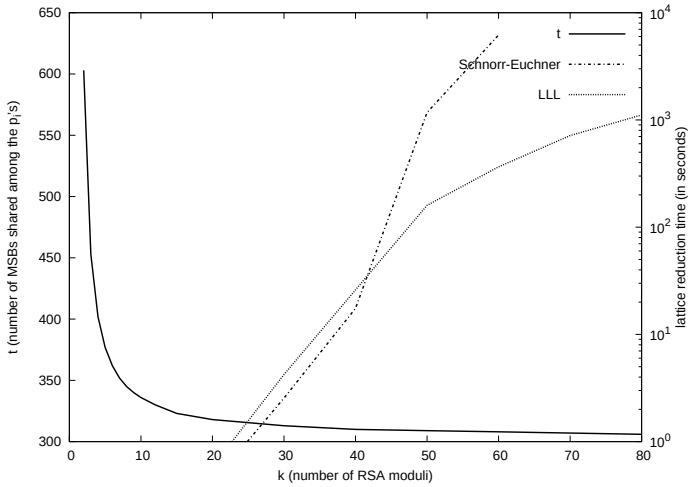
Table 7: Results for $k = 40$ and 1024-bit RSA moduli

α (bit-size of the q_i's)	Bound $t \geq \frac{40}{39}\alpha + 3.68\ldots$ of Theorem 4	Best experimental t (number of bits shared among the p_i's) using LLL algorithm	Best experimental t using Schnorr-Euchner's algorithm	Failure rate of Assumption 1
150	158	156	155	2% ($t = 154$)
200	209	208	207	3% ($t = 206$)
250	261	259	258	1% ($t = 257$)
300	312	310	309	1% ($t = 308$)
350	363	362	361	0% ($t = 360$)
400	414	413	412	2% ($t = 411$)

Table 8: Results for $k = 80$ and 1024-bit RSA moduli

α (bit-size of the q_i's)	Bound $t \geq \frac{80}{79}\alpha + 3.62\ldots$ of Theorem 4	Best experimental t (number of bits shared among the p_i's) using LLL algorithm
150	156	155
200	207	206
250	257	257
300	308	307
350	359	358
400	409	409

Table 9: Running time of LLL and Schnorr-Euchner's algorithm, and bound on t as k grows.
($\alpha = 300$ and $n = 1024$)



References

[1] Miklós Ajtai. Generating random lattices according to the invariant distribution. draft, March 2006.

[2] Aurélie Bauer and Antoine Joux. Toward a rigorous variation of coppersmith's algorithm on three variables. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 361–378. Springer, 2007.

[3] Johannes Blömer and Alexander May. A tool kit for finding small roots of bivariate polynomials over the integers. In Cramer [12], pages 251–267.

[4] Dan Boneh and Glenn Durfee. Cryptanalysis of rsa with private key d less than $n^{0.292}$.

[5] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system I: The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[6] D. Coppersmith. Finding a small root of a univariate modular equation. *Lecture Notes in Computer Science*, pages 155–165, 1996.

[7] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.

[8] Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *EUROCRYPT*, pages 178–189, 1996.

[9] Jean-Sébastien Coron. Finding small roots of bivariate integer polynomial equations revisited. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 492–505. Springer, 2004.

[10] Jean-Sébastien Coron. Finding small roots of bivariate integer polynomial equations: A direct approach. In Menezes [21], pages 379–394.

[11] D.A. Cox, J.B. Little, and D. O'Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Verlag, 2007.

[12] Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

[13] Matthias Ernst, Ellen Jochemsz, Alexander May, and Benne de Weger. Partial key exposure attacks on rsa up to full size exponents. In Cramer [12], pages 371–386.

[14] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In *EUROCRYPT*, pages 31–51, 2008.

[15] Guillaume Hanrot and Damien Stehlé. Improved analysis of kannan's shortest lattice vector algorithm. In Menezes [21], pages 170–186.

[16] M.J. Hinek and D.R. Stinson. An inequality about factors of multivariate polynomials. *University Waterloo*.

[17] Ellen Jochemsz and Alexander May. A strategy for finding roots of multivariate polynomials with new applications in attacking rsa variants. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 267–282. Springer, 2006.

[18] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *STOC*, pages 193–206, 1983.

[19] A. May. Using LLL-reduction for solving RSA and factorization problems: a survey. In *LLL+ 25 Conference in honour of the 25th birthday of the LLL algorithm*, 2007.

[20] Alexander May and Maike Ritzenhofen. Implicit factoring: On polynomial time factoring given only an implicit hint. In *Public Key Cryptography*, pages 1–14, 2009.

[21] Alfred Menezes, editor. *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*. Springer, 2007.

[22] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.

[23] Phong Q. Nguyen and Damien Stehlé. Floating-point lll revisited. In Cramer [12], pages 215–233.

[24] Xavier Pujol and Damien Stehlé. Rigorous and efficient short lattice vectors enumeration. In *ASIACRYPT*, pages 390–405, 2008.

[25] S. Sarkar and S. Maitra. Further Results on Implicit Factoring in Polynomial Time. *Advances in Mathematics of Communications*, 3(2):205–217, 2009.

[26] MJ Wiener, B.N.R. Ltd, and O. Ottawa. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36(3):553–558, 1990.

A Preliminaries on lattice

An integer lattice L is an additive subgroup of $\mathbb{Z}^n$. Equivalently, it can be defined as the set of all integer linear combinations of d independent vectors $\mathbf{b_1}, \ldots, \mathbf{b_d}$ of $\mathbb{Z}^n$. The integer d is called the *dimension* of L, and $B = (\mathbf{b_1}, \ldots, \mathbf{b_d})$ is one of its *basis*. All the bases of L are related by a unimodular transformation. The *volume* (or *determinant*) of L is the d-dimensional volume of the parallelepiped spanned by the vectors of a basis of L and is equal to the square root of the determinant of the Gramian matrix of B. It does not depend upon the choice of B. We denote it by $\mathrm{Vol}(L)$.

We state (without proofs) common results on lattices that will be used throughout this paper. Readers interested in getting more details and proofs can refer to [22].

Definition 1. For $1 \leq r \leq d$, let $\lambda_r(L)$ be the least real number such that there exist at least r linearly independent vectors of L of Euclidean norm smaller than or equal to $\lambda_r(L)$. We call $\lambda_1(L), \ldots, \lambda_d(L)$ the d *minima* of L, and we call $g(L) = \frac{\lambda_2(L)}{\lambda_1(L)} \geq 1$ the *gap* of L.

Lemma 5 (Hadamard). Let $B = (\mathbf{b_1}, \ldots, \mathbf{b_d})$ be a basis of a d-dimensional integer lattice of $\mathbb{Z}^n$. Then the following inequality holds:

$$\prod_{i=1}^{d} \|\mathbf{b_i}\| \geq \mathrm{Vol}(L)$$

Theorem 5 (Minkowski)**.** *Let L be a d-dimensional lattice of $\mathbb{Z}^n$. Then there exists a non zero vector $\mathbf{v}$ in L with the following property:*

$$\|\mathbf{v}\| \leq \sqrt{d}\,\mathrm{Vol}(L)^{\frac{1}{d}}$$

An immediate consequence of Minkowski's theorem is that:

$$\lambda_1(L) \leq \sqrt{d}\,\mathrm{Vol}(L)^{\frac{1}{d}}$$

Theorem 6 (Gaussian heuristic, see [1])**.** *Let L be a random d-dimensional lattice of $\mathbb{Z}^n$. Then, with overwhelming probability, all the minima of L are asymptotically close to:*

$$\sqrt{\frac{d}{2\pi e}}\,\mathrm{Vol}(L)^{\frac{1}{d}}$$

Theorem 7 (Lagrange reduction)**.** *Let L be a 2-dimensional lattice of $\mathbb{Z}^n$, given by a basis $B = (\mathbf{b_1}, \mathbf{b_2})$. Then one can compute a Lagrange-reduced basis $B' = (\mathbf{v_1}, \mathbf{v_2})$ of L in time:*

$$\mathcal{O}(n \log^2(\max(\|\mathbf{b_1}\|, \|\mathbf{b_2}\|)))$$

Besides, it verifies:

$$\|\mathbf{v_1}\| = \lambda_1(L) \quad \textit{and} \quad \|\mathbf{v_2}\| = \lambda_2(L)$$

More information about the running time of the Lagrange reduction may be found in [22].

Theorem 8 (Kannan's algorithm, see [18, 24, 15])**.** *Let L be a d-dimensional lattice of $\mathbb{Z}^n$ given by a basis $(\mathbf{b_1}, \ldots, \mathbf{b_d})$. One can compute a shortest vector of L (with norm equal to $\lambda_1(L)$) in time*

$$\mathcal{O}(\mathcal{P}(\log B, n)\, d^{\frac{d}{2e}+o(d)})$$

where $\mathcal{P}$ is a polynomial and $B = \max_i(\|\mathbf{b_i}\|)$. This is done by computing a HKZ-reduced basis of L.

The time complexity of finding a shortest vector of L will be denoted by $\mathcal{C}(d, n, B)$ (hence $\mathcal{C}(d, n, B) \leq \mathcal{O}(\mathcal{P}(\log B, n)\, d^{\frac{d}{2e}+o(d)})$).

Theorem 9 (LLL)**.** *Let L be a d-dimensional lattice of $\mathbb{Z}^n$ given by a basis $(\mathbf{b_1}, \ldots, \mathbf{b_d})$. Then LLL algorithm computes a reduced basis $(\mathbf{v_1}, \ldots, \mathbf{v_d})$ that satisfy:*

$$\forall i,\ \|\mathbf{v_i}\| \leq 2^{\frac{d(d-1)}{4(d+1-i)}} \operatorname{Vol}(L)^{\frac{1}{d+1-i}}$$

The running time of Nguyen and Stehlé's version is $\mathcal{O}(d^5(d + \log B) \log B)$ where $B = \max_i(\|\mathbf{b_i}\|)$, see [23].

In practice, LLL algorithm is known to perform much better than expected. It has been experimentally established in [14] that we can expect the following bound on $\|\mathbf{v_1}\|$ on random lattices:

$$\|\mathbf{v_1}\| \leq 1.0219^d \operatorname{Vol}(L)^{\frac{1}{d}}$$

and that finding a shortest vector of a lattice with gap greater than 1.0219^d should be easy using LLL.

B Additional experimental results of the generalized Coppersmith's multivariate method

This appendix contains additional experimental results of our generalized Coppersmith's multivariate method described in section 4.

Table 10: Success rate of the different methods in the case $n = 4$, $k = 2$ and rectangular polynomials of degree 1 in each variable.

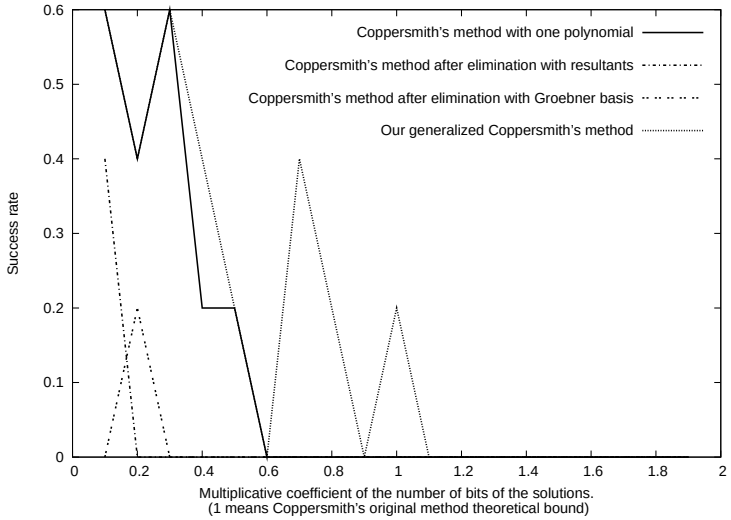
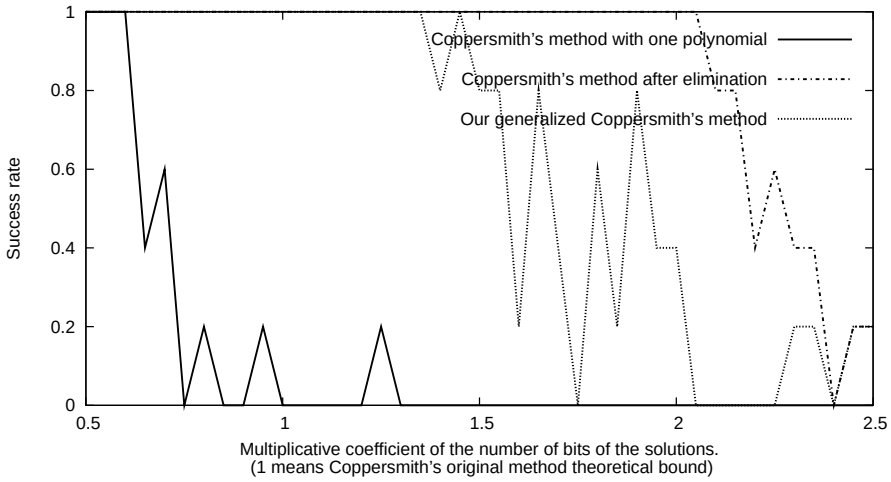


Table 11: Success rate of the different methods in the case $n = 3$, $k = 2$ and triangular polynomials of total degree 2.



C Exact computation of the Volume of lattice L of section 5.2

In this section, we compute exactly the volume of the lattice L defined at the beginning of section 5.2. As a visual example of the construction of this lattice, the reader may take a look at the matrix defined in equation (23) in the case of $k = 4$. We use the notations of section 5.2.

Lemma 6. Let L be the lattice whose construction is described at the beginning of section 5.2. Then its volume is:

$$\mathrm{Vol}(L) = K \left(K^2 + \sum_{i=1}^{k} N_i^2 \right)^{\frac{k-1}{2}}$$

Proof. Let G be the Gramian matrix (of size $k \times k$) of L. Its diagonal terms are:

$$\langle \mathbf{v_i}, \mathbf{v_i} \rangle = K^2 + \sum_{\substack{u=1 \\ u \neq i}}^{k} N_u^2$$

and its other terms are:

$$\langle \mathbf{v_i}, \mathbf{v_j} \rangle = -N_i N_j$$

Observe that we can rewrite G as follows:

$$G = \left(K^2 + \sum_{i=1}^{k} N_i^2\right) I_{k\times k} + J$$

where $I_{k\times k}$ is the identity matrix of size k and J is the $k \times k$ matrix with terms $-N_i N_j$. If we let χ_J be the characteristic polynomial of J and $\lambda_0 = K^2 + \sum_{i=1}^{k} N_i^2$, we observe that $\det(G) = \chi_J(-\lambda_0)$.

All the columns of J are multiples of $\begin{pmatrix} N_1 \\ N_2 \\ \vdots \\ N_k \end{pmatrix}$. The rank of J is thus 1. J has therefore the eigenvalue 0 with multiplicity $k-1$. The last eigenvalue is computed using its trace: $Tr(J) = -\sum_{i=1}^{k} N_i^2$. Therefore, up to a sign,

$$\chi_J(X) = X^{k-1}\left(X + \sum_{i=1}^{k} N_i^2\right)$$

We conclude that

$$\det(G) = \chi_J\left(-K^2 - \sum_{i=1}^{k} N_i^2\right) = K^2\left(K^2 + \sum_{i=1}^{k} N_i^2\right)^{k-1}$$

and that

$$\mathrm{Vol}(L) = \sqrt{\det(G)} = K\left(K^2 + \sum_{i=1}^{k} N_i^2\right)^{\frac{k-1}{2}}$$

□