Using LLL-Reduction for Solving RSA and Factorization Problems

Alexander May

Abstract 25 years ago, Lenstra, Lenstra and Lovász presented their celebrated LLL lattice reduction algorithm. Among the various applications of the LLL algorithm is a method due to Coppersmith for finding small roots of polynomial equations. We give a survey of the applications of this root finding method to the problem of inverting the RSA function and the factorization problem. As we will see, most of the results are of a dual nature, they can either be interpreted as cryptanalytic results or as hardness/security results.

Keywords: LLL reduction, Coppersmith's method, RSA, Factoring, Oracles

1 Introduction

The RSA cryptosystem invented by Rivest, Shamir and Adleman in 1977 [76] is today's most important public-key cryptosystem. Let us denote by $N = pq$ an RSA-modulus which is the product of two primes p,q of the same bit-size. Let e be an integer co-prime to Euler's totient function $\phi(N) = (p-1)(q-1)$. The RSA encryption function takes a message m to the e^{th} power in the ring $\mathbb{Z}_N$. The security of RSA relies on the difficulty of inverting the RSA encryption function on the average, i.e. extracting e^{th} roots in the ring $\mathbb{Z}_N$. We call this problem the RSA inversion problem or the RSA problem for short.

Let d be the inverse of e modulo $\phi(N)$. Computing d^{th} powers in $\mathbb{Z}_N$ inverts the RSA encryption function. Since d can be easily computed when the prime factorization of N is known, the RSA cryptosystem is at most as secure as the problem of computing d and the problem of factoring N. Indeed, we will see that the last two

Alexander May
Horst Görtz Institute for IT-Security
Faculty of Mathematics
Ruhr-University Bochum, Germany
e-mail: alex.may@ruhr-uni-bochum.de

problems are polynomial time equivalent. However, it is one of the most challenging problems to prove or disprove the polynomial time equivalence of the RSA problem and the problem of factoring N. There are results that these problems are not equivalent under restricted reductions [20]. On the other hand, one can show that in restricted generic attack models both problems appear to be equivalent [21, 58].

Despite considerable efforts to attack RSA (see [11, 54] for surveys), the currently best way is still to factor the RSA modulus. Consequently, researchers focussed for a long time on the construction of factorization algorithms for attacking RSA. In this factorization line of research, the goal is to minimize the computational complexity in the common Turing machine model. The most important milestones in the construction of factorization algorithms in the 80s and 90s are the invention of the Quadratic Sieve [73], the Elliptic Curve Method [59] and the Number Field Sieve (NFS) [60]. The NFS is currently the best algorithm for factoring RSA moduli. It factors N in subexponential time and space $L_N[\frac{1}{3},c] = \mathcal{O}(\exp(c(\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}))$ for $c \approx 1.9$.

Of course, the ultimate cryptanalyst's goal is the construction of a polynomial time algorithm for either the RSA problem or the factorization problem. Since it is unknown whether there exist algorithms for these problems with Turing complexity $L_N[\alpha,c]$ for $\alpha < \frac{1}{3}$, one might ask for polynomial time algorithms in *other machine models* or for interesting *relaxations* of the RSA and factorization problem.

In 1994, Shor [79] presented an algorithm for solving the factorization problem in time and space polynomial in the bit-length of N, provided that the model of Turing machines is replaced by the model of quantum Turing machines. This ground-breaking theoretical result led to intensive engineering efforts for building quantum computers in practice. However, today it is still unclear whether quantum computers with a large number of quantum bits can ever be constructed.

In the 90s, another interesting line of research evolved, which uses *polynomial time* algorithms in the *Turing machine model*. However, in order to achieve polynomial complexity one has to relax the RSA and factorization problem. So instead of changing the model of computation, one relaxes the problems themselves by looking at restricted instances. The most natural restriction is realized by limiting the parameter set of the instances to an interval which is smaller than in the general setting, but still of exponential size.

A variation of this limiting approach addresses full parameter sets, but allows additional access to an oracle for parts of the solution, e.g. for some of the bits. Notice that the oracle queries have the effect of cutting down the search space for the solution. The so-called *oracle complexity* measures the number of oracle queries that is required in order to solve the underlying problem in polynomial time. Of course, one is interested in minimizing the number of oracle queries and in restricting the oracle's power, i.e. the type of queries that an oracle replies to. Oracles are motivated by other cryptographical mechanisms, so-called side-channel attacks, that often leak partial information of the secrets, and therefore behave in practice like an oracle.

In the following we will call both approaches, limiting the parameter sets and allowing for an oracle, *relaxations* of the problem instances. In order to solve these relaxed instances, one models them as a polynomial equation and tries to find the integer solutions.

Let us illustrate this approach by a simple example. The RSA factorization problem is the problem of finding p,q on input N. This can be modeled by a polynomial equation $f(x,y) = N - xy$. The positive integer roots of this polynomial equation are $(1,N),(p,q),(q,p),(N,1)$. Since we assume that p,q are of the same bit-size, finding all integer solutions which are in absolute value smaller than roughly $\sqrt{N}$ suffices to solve the factorization problem. Thus, one only has to find *small solutions*, where *small* means that the size of the root is small compared to the size of the coefficients of the polynomial. Naturally, one can define upper bounds X,Y for the size of the roots in x,y, respectively. The ultimate goal is to find a polynomial time algorithm which succeeds whenever $XY \leq N$. Since we do not know how to achieve this bound, we relax the factorization problem.

A natural relaxation of this problem is to narrow down the search space for the prime factors. Assume that we are given oracle access to the most significant bits of p. This allows us to compute an approximation $\tilde{p}$ of p such that $|p - \tilde{p}|$ is significantly smaller than $\sqrt{N}$. Then $\tilde{q} = \frac{N}{\tilde{p}}$ defines an approximation of q. Therefore, we obtain the polynomial equation $f(x,y) = N - (\tilde{p}+x)(\tilde{q}+y)$ with a small root $(p-\tilde{p}, q-\tilde{q})$, where the size of the root depends on the quality of the approximation. It was shown by Coppersmith in 1996 [24], that the solution of this problem can be found in polynomial time if $XY \leq N^{\frac{1}{2}}$.

Building on works in the late 80's [42, 37], Coppersmith [23, 24, 25, 26] derived a general algorithm for finding small roots of polynomial equations. This root finding algorithm in turn is essentially based on the famous LLL-reduction algorithm by Lenstra, Lenstra and Lovász [61]. The key idea is to encode polynomial equations with small solutions as coefficient vectors that have a small Euclidean norm. These coefficient vectors can efficiently be found by an application of the LLL-reduction algorithm.

We will survey several applications of Coppersmith's algorithm to relaxations of the RSA problem and the factorization problem. Many of these applications naturally allow for a dual interpretation, both as a cryptanalytic result and as a security result. Let us give an example for this duality. In 1996, Coppersmith [23] showed that for RSA with $e = 3$, an attacker who knows $2/3$ of an RSA-encrypted message m can recover the remaining third from the ciphertext in polynomial time. The cryptanalytic interpretation is that knowing only a $2/3$-fraction of the plaintext is already enough to recover the whole. The security interpretation is that recovering a $2/3$-fraction must be hard, provided that solving the RSA problem for $e = 3$ is hard. Thus, this result establishes the security of a $2/3$-fraction of the underlying plaintext under the RSA assumption. This security interpretation was used by Shoup [80] to show the security of RSA-OAEP for $e = 3$ under chosen ciphertext attacks. We will elaborate a bit more on this duality effect in the paper.

This survey is organized as follows. We start in Section 2 by giving a high-level description of Coppersmith's algorithm for finding small roots of univariate modular polynomials. We state a theorem which provides us with an upper bound for the size of the roots of a univariate polynomial that can efficiently be found.

The details of the theorem's proof are given in Section 3. This section is devoted to people who are interested in the technical details of the method, and those who want to implement a Coppersmith-type univariate root finding algorithm. It is the only section that requires some basic knowledge of lattice theory from the reader. People that are mainly interested in the applications of Coppersmith's method can proceed to the subsequent section.

In Section 4, we will extensively use our theorem for finding small roots. We will model certain relaxed RSA and factorization problems as univariate polynomial equations. For instance, we present Coppersmith's attack on RSA with stereotyped messages [23] and show its dual use in Shoup's security proof [80] and for the construction of an RSA-based pseudorandom number generator proposed by Steinfeld, Pieprzyk and Wang [82]. Moreover, we will show a generalization of Håstad's broadcast attack [42] on RSA-encrypted, polynomially related messages that provides a natural link to Coppersmith's attack on stereotyped RSA messages.

We then describe the *factoring with high bits known* results from Coppersmith [24] and Boneh, Durfee, Howgrave-Graham [18]. Furthermore, we show a deterministic polynomial time reduction of factoring to computing d [66, 30], which establishes the hardness of the so-called RSA secret key recovery problem under the factorization assumption. We conclude this section by stating Boneh's algorithm [12] for finding smooth integers in short intervals. The problem of finding smooth integers is related to classical factorization algorithms such as the Number Field Sieve.

In Section 5, we will turn our focus to multivariate extensions of Coppersmith's LLL-based method. We present Wiener's attack [88] on RSA with $d \leq N^{\frac{1}{4}}$ as a bivariate linear equation, which was originally phrased in terms of the continued fraction algorithm. We then present the bivariate polynomial equation of Boneh and Durfee [14, 15] that led to a heuristic improvement of the bound to $d \leq N^{0.292}$. As an example of an application with more variables, we present a heuristic polynomial time attack of Jochemsz, May [52] for RSA with so-called CRT-exponents $d \bmod p-1, d \bmod q-1$ smaller than $N^{0.073}$. Dually to these attacks, the server-based RSA signature generation proposals of Boneh, Durfee, Frankel [16] and Steinfeld, Zheng [83] are constructive security applications.

Since the number of applications of Coppersmith's LLL-based method for the RSA/factorization problem is already far too large to capture all the different results in this survey, we try to provide a more comprehensive list of references in Section 6. We are aware of the fact that it is impossible to achieve completeness of such a list, but our references will serve the purpose of a good starting point for further reading.

In Section 7, we give some open problems in this area and try to speculate in which direction this line of research will go. Especially, we discuss to which extent we can go from relaxed instances towards general problem instances, and where the limits of the method are. This discussion naturally leads to speculations whether any small root finding algorithm based on LLL-reduction will eventually have the potential to solve general instances of the RSA problem or the factorization problem in polynomial time.

2 How to Find Small Roots: The Univariate Case

We first introduce the problem of finding solutions of a modular univariate polynomial equation. Then we argue that this approach extends to polynomials in more variables in a heuristic manner.

Let N be a positive integer of unknown factorization with divisor $b \geq N^{\beta}$, $0 < \beta \leq 1$. [1] Let $f(x)$ be a monic univariate polynomial of degree δ. We are looking for all small roots of the polynomial f modulo b. I.e., we want to efficiently find all solutions x_0 satisfying

$$f(x_0) = 0 \bmod b \quad \text{with} \quad |x_0| \leq X,$$

where X is an upper bound on the size of the solutions. Our goal is to maximize the bound X, with the restriction that the running time of our method should be polynomial in the input size, i.e. polynomial in the parameters $(\log N, \delta)$.

We would like to stress that N is an integer of *unknown* factorization, which makes the above root finding problem hard to solve. If the prime factors of N are given, efficient algorithms with finite field arithmetic are known for the problem.

In 1996, Coppersmith [25] proposed an elegant LLL-based method for finding small solutions of univariate polynomial equations. Here, we describe his approach using the notion of Howgrave-Graham's reformulation [48] of the method. Coppersmith's approach is basically a reduction of solving modular polynomial equations to solving univariate polynomials over the integers. I.e. one constructs from $f(x)$ another univariate polynomial $g(x)$ that contains all the small modular roots of $f(x)$ over the integers:

$$f(x_0) = 0 \bmod b \quad \Rightarrow \quad g(x_0) = 0 \text{ over } \mathbb{Z} \quad \text{for all } |x_0| \leq X.$$

The algorithmic idea for the construction of $g(x)$ from $f(x)$ can be described via the following two steps:

(1) Fix an integer m. Construct a collection C of polynomials $f_1(x), f_2(x), \ldots, f_n(x)$ that all have the small roots x_0 modulo b^m. As an example, take the collection

$$\begin{aligned} f_i(x) &= N^{m-i} f^i(x) \quad && \text{for } i = 1, \ldots, m \\ f_{m+i}(x) &= x^i f^m(x) \quad && \text{for } i = 1, \ldots, m. \end{aligned}$$

(2) Construct an integer linear combination $g(x) = \sum_{i=1}^{n} a_i f_i(x)$, $a_i \in \mathbb{Z}$ such that the condition

$$|g(x_0)| < b^m$$

holds. Notice that b^m divides all $f_i(x_0)$ by construction. Therefore, b^m also divides $g(x_0)$. But then $g(x_0) = 0 \bmod b^m$ and $|g(x_0)| < b^m$, which implies that $g(x_0) = 0$ over the integers.

---

[1] An important special case is $b = N$, i.e. $\beta = 1$.

The construction in step (2) is realized by an LLL-based approach. Namely, one can easily show that every polynomial g whose coefficient vector of $g(xX)$ has sufficiently small norm fulfills the condition $|g(x_0)| < b^m$. The integer linear combinations of the coefficient vectors of $f_i(xX)$, $i = 1 \ldots n$, form a lattice L. Applying a lattice basis reduction algorithm to a basis of L yields a small norm coefficient vector $g(xX)$. One can show that in our case the LLL-reduction algorithm of Lenstra, Lenstra and Lovász [61] outputs a sufficiently small vector. Therefore, $g(x)$ can be computed in polynomial time via LLL-reduction.

Eventually, one has to find the roots of $g(x)$ over the integers. This can be done by standard polynomial factorization methods like the Berlekamp-Zassenhaus algorithm. Interestingly, the initial application of the LLL algorithm was a deterministic polynomial time algorithm [61] for factoring polynomials in $\mathbb{Q}[X]$. In 2001, van Hoeij [46, 47] proposed an improved, highly efficient LLL-based factorization algorithm (see [55] for an introduction). Thus we cannot only use LLL to construct g but also to find its integer roots.

The details of the proof of the following result can be found in Section 3.

Theorem 1 *Let N be an integer of unknown factorization, which has a divisor $b \geq N^\beta$, $0 < \beta \leq 1$. Let $f(x)$ be a univariate monic polynomial of degree δ and let $c \geq 1$. Then we can find all solutions x_0 of the equation*

$$f(x) = 0 \bmod b \quad \textit{with} \quad |x_0| \leq cN^{\frac{\beta^2}{\delta}}$$

in time $\mathcal{O}(c\delta^5 \log^9 N)$.

Although LLL reduction only approximates a shortest vector up to some factor that is exponential in the lattice dimension, it is important to point out that lattice reduction techniques which give better approximations do not help to improve the bound given in Theorem 1.

Coppersmith proved this result for the special case $\beta = 1$, i.e. $b = N$. The term β^2 first appeared in Howgrave-Graham's work [50] for the special case $\delta = 1$, i.e. for a linear polynomial. A proof of Theorem 1 first appeared in [65].

Coppersmith's method generalizes in a natural way to modular multivariate polynomials $f(x_1, \ldots, x_\ell)$. The idea is to construct ℓ *algebraically independent* polynomials $g^{(1)}, \ldots, g^{(\ell)}$ that all share the desired small roots over the integers. The roots are then computed by resultant computations. For $\ell \geq 2$, this is a heuristic method because although the LLL-algorithm guarantees linear independence of the coefficient vectors, it does not guarantee algebraic independence of the corresponding polynomials.

The case of solving multivariate polynomial equations *over the integers* – not modular – uses similar techniques. In the integer case, the method of finding small roots of bivariate polynomials $f(x, y)$ is rigorous, whereas the extension to more than two variables is again a heuristic. Coron showed in [28, 29], that the case of solving integer polynomials can in principle be reduced to the case of solving modular polynomials.

3 Proof of Theorem 1 & Algorithmic Considerations

In this section, we will give a complete proof of Theorem 1. Readers who are mainly interested in the methods' applications can skip this section and proceed to Section 4.

We provide an algorithm that on input

- an integer N of unknown factorization,
- a monic, univariate polynomial $f(x)$ of degree δ
- a bound $\beta \in (0,1]$, such that $b \geq N^{\beta}$ for some divisor b of N

outputs in time polynomial in $\log N$ and δ all solutions x_0 such that

- $f(x_0) = 0 \bmod b$ and
- $|x_0| \leq N^{\frac{\beta^2}{\delta}}$.

Normally, the property that $f(x)$ is monic is no restriction in practice. Assume that $f(x)$ has a leading coefficient $a_{\delta} \neq 1$. Then we can either make $f(x)$ monic by multiplying with the inverse of a_{δ} modulo N, or we find a non-trivial factorization of N. In the latter case, we can work modulo the factors of N.

The following lemma of Howgrave-Graham [48] gives us two criteria under which we can find a polynomial $g(x)$ that evaluates to zero over the integers at small roots.

Theorem 2 (Howgrave-Graham) *Let $g(x)$ be an univariate polynomial with n monomials. Further, let m be a positive integer. Suppose that*

(1) $g(x_0) = 0 \bmod b^m$ *where* $|x_0| \leq X$
(2) $\|g(xX)\| < \frac{b^m}{\sqrt{n}}$

Then $g(x_0) = 0$ holds over the integers.

Proof: We have

$$\begin{aligned} |g(x_0)| &= \sum_i c_i x_0^i \leq \sum_i |c_i x_0^i| \\ &\leq \sum_i |c_i| X^i \leq \sqrt{n}\|g(xX)\| < b^m. \end{aligned}$$

But $g(x_0)$ is a multiple of b^m and therefore it must be zero. ◻

Using powers of f, we construct a collection $f_1(x), \ldots, f_n(x)$ of polynomials that all have the desired roots x_0 modulo b^m. Thus, for every integer linear combination g we have

$$g(x_0) = \sum_{i=1}^{n} a_i f_i(x_0) = 0 \bmod b^m, \quad a_i \in \mathbb{Z}.$$

Hence every integer linear combination satisfies condition (1) of Lemma 2. Among all integer linear combinations, we search for one that also satisfies condition (2). In other words, we have to search among all integer linear combinations of the coefficient vectors $f_i(xX)$ for a vector with Euclidean norm smaller than $\frac{b^m}{\sqrt{n}}$. This can be achieved by finding a short vector in the lattice L spanned by the coefficient vectors of $f_i(xX)$.

Our goal is to ensure that the LLL algorithm finds a vector v with $\|v\| < \frac{b^m}{\sqrt{n}}$ in L. By a theorem of Lenstra, Lenstra and Lovász [61], the norm of a shortest vector v in an LLL-reduced lattice basis can by related to the determinant $\det(L)$ of the corresponding lattice L with dimension n via

$$\|v\| \leq 2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}}.$$

The determinant $\det(L)$ can be easily computed from the coefficient vectors of $f_i(xX)$. If we could satisfy the condition

$$2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}} < \frac{N^{\beta m}}{\sqrt{n}}, \tag{1}$$

then we obtain the desired inequality $\|v\| < \frac{N^{\beta m}}{\sqrt{n}} \leq \frac{b^m}{\sqrt{n}}$.

Neglecting low-order terms in (1), i.e. terms that do not depend on N, we obtain the simplified condition

$$\det(L) < N^{\beta m n}.$$

Let L be a lattice of dimension n with basis B satisfying this condition. Then on average a basis vector $v \in B$ contributes to the determinant with a factor less than $N^{\beta m}$. We call such a basis vector a *helpful vector*. Helpful vectors will play a central role for the construction of an optimized lattice basis.

The following theorem of Coppersmith states that for a monic polynomial $f(x)$ of degree δ, all roots x_0 with $|x_0| \leq \frac{1}{2} N^{\frac{\beta^2}{\delta} - \varepsilon}$ can be found in polynomial time. We will later show that the error term ε and the factor $\frac{1}{2}$ can be easily eliminated, which will lead to a proof of Theorem 1.

Theorem 3 (Coppersmith) *Let N be an integer of unknown factorization, which has a divisor $b \geq N^\beta$, $0 < \beta \leq 1$. Let $0 < \varepsilon \leq \frac{1}{7}\beta$. Furthermore, let $f(x)$ be a univariate monic polynomial of degree δ. Then we can find all solutions x_0 for the equation*

$$f(x) = 0 \bmod b \quad \textit{with} \quad |x_0| \leq \frac{1}{2} N^{\frac{\beta^2}{\delta} - \varepsilon}.$$

The running time is dominated by the time to LLL-reduce a lattice basis of dimension $\mathcal{O}(\varepsilon^{-1}\delta)$ with entries of bit-size $\mathcal{O}(\varepsilon^{-1} \log N)$. This can be achieved in time $\mathcal{O}(\varepsilon^{-7}\delta^5 \log^2 N)$.

Proof: Define $X := \frac{1}{2} N^{\frac{\beta^2}{\delta} - \varepsilon}$. Let us apply the two steps of Coppersmith's method as described in Section 2. In the first step, we fix

$$m = \left\lceil \frac{\beta^2}{\delta\varepsilon} \right\rceil. \tag{2}$$

Next, we choose a collection C of polynomials, where each polynomial has a root x_0 modulo b^m whenever $f(x)$ has the root x_0 modulo b. In our case, we include in C the polynomials

$$\begin{array}{lllll}
N^m, & xN^m, & x^2N^m, & \ldots & x^{\delta-1}N^m, \\
N^{m-1}f, & xN^{m-1}f, & x^2N^{m-1}f, & \ldots & x^{\delta-1}N^{m-1}f, \\
N^{m-2}f^2, & xN^{m-2}f^2, & x^2N^{m-2}f^2, & \ldots & x^{\delta-1}N^{m-2}f^2, \\
\vdots & \vdots & \vdots & & \vdots \\
Nf^{m-1}, & xNf^{m-1}, & x^2Nf^{m-1}, & \ldots & x^{\delta-1}Nf^{m-1}.
\end{array}$$

Additionally, we take the polynomials

$$f^m, xf^m, x^2f^m, \ldots, x^{t-1}f^m$$

for some t that has to be optimized as a function of m.

Note that by our ordering the k^{th} polynomial of C is a polynomial of degree k. Thus, it introduces the new monomial x^k. We could also write the choice of our polynomials in C in a more compact form. Namely, we have chosen the polynomials

$$\begin{array}{lll}
g_{i,j}(x) & = x^jN^if^{m-i}(x) & \text{for } i = 0,\ldots,m-1,\ j = 0,\ldots,\delta-1 \text{ and} \\
h_i(x) & = x^if^m(x) & \text{for } i = 0,\ldots,t-1.
\end{array}$$

In Step 2 of Coppersmith's method, we construct the lattice L that is spanned by the coefficient vectors of $g_{i,j}(xX)$ and $h_i(xX)$. As we noticed before, we can order the polynomials $g_{i,j}$ and h_i in strictly increasing order of their degree k. Therefore, the basis B of L, that has as row vectors the coefficient vectors of $g_{i,j}(xX)$ and $h_i(xX)$, can be written as a lower triangular matrix. Let $n := \delta m + t$, then we write B as the $(n \times n)$-matrix given in Table 3.

Since B is in lower triangular form, $\det(L)$ is simply the product of all entries on the diagonal:

$$\det(L) = N^{\frac{1}{2}\delta m(m+1)} X^{\frac{1}{2}n(n-1)}. \tag{3}$$

Now we want to optimize the parameter t, which is equivalent to the optimization of $n = \delta m + t$. Remember that we argued before that every vector which contributes to the determinant by a factor less than $N^{\beta m}$ is *helpful*. In our setting this means that we have to ensure that the entries of the coefficient vectors $h_i(xX)$ on the diagonal are all less than $N^{\beta m}$, i.e., we have the condition

$$X^{n-1} < N^{\beta m}.$$

Since $X^{n-1} < N^{(\frac{\beta^2}{\delta} - \varepsilon)(n-1)} < N^{\frac{\beta^2}{\delta} n}$ this condition is satisfied for the choice

$$
\left(\begin{array}{ccccccccccccc}
N^m &&&&&&&&&&&& \\
& N^m X &&&&&&&&&&& \\
&& \ddots &&&&&&&&&& \\
&&& N^m X^{\delta-1} &&&&&&&&& \\
\hline
& \ddots & \ddots & \ddots & \ddots &&&&&&&& \\
\hline
- & - & \dots & - & \dots & NX^{\delta m-\delta} &&&&&&& \\
& - & \dots & - & \dots & - & NX^{\delta m-\delta+1} &&&&&& \\
&& \ddots & \ddots & \ddots & \ddots & \ddots & \ddots &&&&& \\
&&& - & \dots & - & - & \dots & NX^{\delta m-1} &&&& \\
\hline\hline
- & - & \dots & - & \dots & - & - & - & - & X^{\delta m} &&& \\
& - & \dots & - & \dots & - & - & - & - & - & X^{\delta m+1} && \\
&&&& \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
&&&&& - & - & - & - & - & - & \dots & X^{\delta m+t-1}
\end{array}\right)
$$

Table 1 Basis B of the lattice L. We use the following notation: Every non-specified entry is zero. The entries marked with "-" may be non-zero, but the determinant of the lattice does not depend on these values.

$$n \leq \frac{\delta}{\beta} m. \tag{4}$$

According to (2), we know that $m \leq \frac{\beta^2}{\delta\varepsilon} + 1$. Then, we immediately have a bound for the lattice dimension

$$n \leq \frac{\beta}{\varepsilon} + \frac{\delta}{\beta}.$$

Using $7\beta^{-1} \leq \varepsilon^{-1}$, we obtain $n = \mathscr{O}(\varepsilon^{-1}\delta)$. We choose n as the maximal integer that satisfies inequality (4). This yields a lower bound of

$$n > \frac{\delta}{\beta} m - 1 \geq \frac{\beta}{\varepsilon} - 1 \geq 6.$$

In order to prove the running time, we also need to upper-bound the bit-size of the entries in B. Notice that for every power f^{m-i} in the definition of $g_{i,j}$ and h_i, we can reduce the coefficients modulo N^{m-i}, since x_0 must be a root modulo N^{m-i}. Thus the largest coefficient in a product $N^i f^{m-i}$ has a bit-size of at most $m \log(N) = \mathscr{O}(\varepsilon^{-1} \log N)$. Powers of $X = \frac{1}{2} N^{\frac{\beta^2}{\delta} - \varepsilon}$ occur with exponents smaller than n. Thus the bit-size of powers of X can also be upperbounded by

$$n \cdot \frac{\beta^2}{\delta} \log N = \mathscr{O}\left(\frac{\delta}{\varepsilon} \cdot \frac{\beta^2}{\delta}\right) \log N = \mathscr{O}\left(\varepsilon^{-1} \log N\right).$$

Nguyen and Stehlé [72, 81] recently proposed a modified version of the LLL-algorithm called L^2-algorithm. The L^2-algorithm achieves the same approximation quality for a shortest vector as the LLL algorithm, but has an improved worst case running time analysis. It takes time $\mathscr{O}(n^5(n+\log b_m)\log b_m)$, where $\log b_m$ is the maximal bit-size of an entry in B. Thus, we obtain for our method a running time of

$$\mathscr{O}\left(\left(\frac{\delta}{\varepsilon}\right)^5\left(\frac{\delta}{\varepsilon}+\frac{\log N}{\varepsilon}\right)\frac{\log N}{\varepsilon}\right).$$

Notice that we can assume $\delta \leq \log N$, since otherwise our bound $|x_0| \leq N^{\frac{\beta^2}{\delta}-\varepsilon}$ is vacuous. Therefore, we obtain a running time of $\mathscr{O}(\varepsilon^{-7}\delta^5\log^2 N)$.

It remains to show that LLL's approximation quality is sufficient for our purpose. In order to apply the theorem of Howgrave-Graham (Theorem 2), we have to ensure that the LLL algorithm finds a vector in L with norm smaller than $\frac{b^m}{\sqrt{n}}$. Since the LLL algorithm finds a vector v in an n-dimensional lattice with $\|v\| \leq 2^{\frac{n-1}{4}}\det(L)^{\frac{1}{n}}$, we have to satisfy the condition

$$2^{\frac{n-1}{4}}\det(L)^{\frac{1}{n}} < \frac{b^m}{\sqrt{n}}.$$

Using the term for $\det(L)$ in (3) and the fact $b \geq N^\beta$, we obtain the new condition

$$N^{\frac{\delta m(m+1)}{2n}} X^{\frac{n-1}{2}} \leq 2^{-\frac{n-1}{4}} n^{-\frac{1}{2}} N^{\beta m}.$$

This gives us a condition on the size of X:

$$X \leq 2^{-\frac{1}{2}} n^{-\frac{1}{n-1}} N^{\frac{2\beta m}{n-1}-\frac{\delta m(m+1)}{n(n-1)}}.$$

Notice that $n^{-\frac{1}{n-1}} = 2^{-\frac{\log n}{n-1}} \geq 2^{-\frac{1}{2}}$ for $n > 6$. Therefore, our condition simplifies to

$$X \leq \frac{1}{2} N^{\frac{2\beta m}{n-1}-\frac{\delta m(m+1)}{n(n-1)}}.$$

Remember that we made the choice $X = \frac{1}{2}N^{\frac{\beta^2}{\delta}-\varepsilon}$. Hence in order to finish the proof of the theorem, it suffices to show that

$$\frac{2\beta m}{n-1} - \frac{\delta m^2(1+\frac{1}{m})}{n(n-1)} \geq \frac{\beta^2}{\delta} - \varepsilon.$$

We obtain a lower bound for the left-hand side by multiplying with $\frac{n-1}{n}$. Then, we use $n \leq \frac{\delta}{\beta}m$ which gives us

$$2\frac{\beta^2}{\delta} - \frac{\beta^2}{\delta}\left(1+\frac{1}{m}\right) \geq \frac{\beta^2}{\delta} - \varepsilon.$$

This simplifies to

$$-\frac{\beta^2}{\delta} \cdot \frac{1}{m} \geq -\varepsilon.$$

This in turn gives us the condition $m \geq \frac{\beta^2}{\delta\varepsilon}$, which holds by the choice of m that we made in (2).

Let us briefly summarize the whole algorithm which finds all roots of $f(x)$ modulo b that are in absolute value smaller than X.

Coppersmith's method in the univariate case

INPUT: Polynomial $f(x)$ of degree δ, modulus N of unknown factorization that is a multiple of b, a lower bound $b \geq N^{\beta}$, $\varepsilon \leq \frac{1}{7}\beta$

Step 1: Choose $m = \lceil \frac{\beta^2}{\delta\varepsilon} \rceil$ and $t = \lfloor \delta m(\frac{1}{\beta} - 1) \rfloor$.
Compute the polynomials

$$\begin{aligned} g_{i,j}(x) &= x^j N^i f^{m-i}(x) \quad \text{for } i = 0, \ldots, m-1, \ j = 0, \ldots, \delta - 1 \text{ and} \\ h_i(x) &= x^i f^m(x) \qquad\quad\ \text{for } i = 0, \ldots, t-1. \end{aligned}$$

Step 2: Compute the bound $X = \frac{1}{2}\lceil N^{\frac{\beta^2}{\delta} - \varepsilon} \rceil$. Construct the lattice basis B, where the basis vectors of B are the coefficient vectors of $g_{i,j}(xX)$ and $h_i(xX)$.

Step 3: Apply the LLL algorithm to the lattice basis B. Let v be the shortest vector in the LLL reduced basis. The vector v is the coefficient vector of some polynomial $g(xX)$. Construct $g(x)$ from v.

Step 4: Find the set R of all roots of $g(x)$ over the integers using standard methods. For every root $x_0 \in R$ check whether $\gcd(N, f(x_0)) \geq N^{\beta}$. If this condition is not satisfied then remove x_0 from R.

OUTPUT: Set R, where $x_0 \in R$ whenever $f(x_0) = 0 \bmod b$ for an $|x_0| \leq X$.

As we noticed before, all steps of the algorithm can be done in time $\mathcal{O}(\varepsilon^{-7}\delta^5 \log^2 N)$, which concludes the proof of the theorem. □

One should remark that the polynomial $g(x)$ that we construct in Coppersmith's method may contain integer roots that are not roots of $f(x)$ modulo b. Therefore, we use in Step 4 of the above algorithm a simple test whether $f(x_0)$ contains a divisor of N of size at least N^{β}.

It is also worth noticing the following point: The LLL approximation factor of $2^{\frac{n-1}{4}}$ for the shortest vector is exponentially in the lattice dimension n, but this factor essentially translates in the analysis of Theorem 3 to the term $\frac{1}{2}$ for the upper bound of the size of the roots x_0. Thus, computing a shortest vector instead of an LLL approximate version would only improve the bound by a factor of roughly 2 (i.e., only one bit).

Moreover, Theorem 1 is a direct implication of Theorem 3 and shows that we can avoid the terms $\frac{1}{2}$ and ε from the upper bound on x_0. The proof uses a simple brute-force search.

Theorem 1 *Let N be an integer of unknown factorization, which has a divisor $b \geq N^{\beta}$, $0 < \beta \leq 1$. Furthermore, let $f(x)$ be a univariate monic polynomial of degree δ. Then we can find all solutions x_0 for the equation*

$$f(x) = 0 \bmod b \quad \textit{with} \quad |x_0| \leq cN^{\frac{\beta^2}{\delta}}.$$

in time $\mathcal{O}(c\delta^5 \log^9 N)$.

Proof: An application of Theorem 3 with the parameter choice $\varepsilon = \frac{1}{\log N}$ shows that we can find all roots x_0 with

$$|x_0| \leq \frac{1}{4} N^{\frac{\beta^2}{\delta}}$$

in time $\mathcal{O}(\delta^5 \log^9 N)$.

In order to find all roots that are of size at most $cN^{\frac{\beta^2}{\delta}}$ in absolute value, we divide the interval $[-cN^{\frac{\beta^2}{\delta}}, cN^{\frac{\beta^2}{\delta}}]$ into $4c$ subintervals of size $\frac{1}{2}N^{\frac{\beta^2}{\delta}}$ centered at some x_i. For each subinterval with center x_i, we apply the algorithm of Theorem 3 to the polynomial $f(x - x_i)$ and output the roots in this subinterval. $\square$

For completeness reasons and since it is one of the most interesting cases of Coppersmith's method, we explicitly state the special case $b = N$ and $c = 1$, which is given in the work of Coppersmith [25].

Theorem 4 (Coppersmith) *Let N be an integer of unknown factorization. Furthermore, let $f_N(x)$ be a univariate monic polynomial of degree δ. Then we can find all solutions x_0 for the equation*

$$f_N(x) = 0 \bmod N \quad \textit{with} \quad |x_0| \leq N^{\frac{1}{\delta}}$$

in time $\mathcal{O}(\delta^5 \log^9 N)$.

4 Modeling RSA Problems as Univariate Root Finding Problems

We address several RSA related problems that can be solved by finding small roots of univariate modular polynomial equations. Throughout this section, we will assume that $N = pq$ is a product of two primes, and that $e \in \mathbb{Z}^*_{\phi(N)}$. Both N and e are publically known.

4.1 Relaxed RSA Problem: Stereotyped Messages

The RSA problem is the problem of inverting the RSA function. Given $m^e \bmod N$, one has to find the unique e^{th} root $m \in \mathbb{Z}_N$. The *RSA assumption* states that the RSA problem is difficult to solve for randomly chosen $m \in \mathbb{Z}_N$.

Notice that the RSA problem is trivial to solve for *small m and small e*. Namely, if $m < N^{\frac{1}{e}}$ then $m^e \bmod N = m^e$ over $\mathbb{Z}$. Therefore, computation of the e^{th} roots over the integers yields the desired root.

RSA problem

Given: $m^e \bmod N$
Find : $m \in \mathbb{Z}_N$

Relaxed RSA problem: Small e, High Bits Known

Given: $m^e, \tilde{m}$ with $|m - \tilde{m}| \le N^{\frac{1}{e}}$
Find : $m \in \mathbb{Z}_N$

Coppersmith extended this result to the case where m is not small but we know m up to a small part. Namely, we assume knowledge of an approximation $\tilde{m}$ such that $m = \tilde{m} + x_0$ for some unknown part $|x_0| \le N^{\frac{1}{e}}$. This can be modeled as the polynomial equation

$$f(x) = (\tilde{m} + x)^e - m^e \bmod N.$$

Let us apply Theorem 1. We set $\beta = 1$, $\delta = e$ and $c = 1$. Therefore, we can recover x_0 as long as $|x_0| \le N^{\frac{1}{e}}$. This extends the trivial attack where m is small to the inhomogenous case: The most significant bits of m are not zero, but they are known to an attacker.

Clearly, one can interpret this as a cryptanalytic result. E.g., if $e = 3$ then an attacker who can guess the first $2/3$-fraction of the message m is able to reconstruct the last $1/3$-fraction of m in polynomial time. This might happen in situations were the plaintext has a stereotype form like “The password for today is: xxxx”. Therefore, this is often called an *attack on stereotyped messages*. Loosely speaking, the cryptanalytic meaning is that an attacker gets an $\frac{1}{e}$*-fraction of the RSA message efficiently*. We will see in Section 4.4 that this cryptanalytic interpretation can be generalized to the case where the same message m is sent several times.

On the other hand, one can interpret this result in a dual sense as a security result for a $2/3$-fraction of the plaintext bits in an RSA ciphertext. It is as difficult to compute a $2/3$-fraction of m as inverting the RSA problem for $e = 3$. In general, there is a tight reduction from the RSA problem to the problem of finding an $\frac{e-1}{e}$-fraction of the most significant bits. Under the RSA assumption, this shows that the most significant bits of an RSA plaintext are hard to find. Even stronger results on the security of RSA bits were given by Håstad and Näslund [43].

4.2 Constructive Applications of the Relaxed RSA Problem: RSA-OAEP and RSA Pseudorandom Generator

The dual security interpretation of the Relaxed RSA problem was used by Shoup [80] in 2001. He gave a security proof of the padding scheme OAEP [2] when instantiated with the RSA trapdoor function. Here, we only sketch Shoup's proof. More details on the proof and on cryptographic security notations can be found in Gentry's survey [40].

In RSA-OAEP, the plaintext is split into two parts s and t. The first part s depends on the message m, a fixed padding and some randomization parameter r of length k bits. The fixed padding ensures that s fulfills a well-defined format that can be checked. The second part t is simply $h(s) \oplus r$ for some hash function h, which is modeled as a random oracle. One encrypts the padded message $s \cdot 2^k + t$. Let c be the corresponding ciphertext.

Bellare and Rogaway [2] showed that RSA-OAEP is CCA1-secure, i.e. secure against so-called lunch-time attacks. It was widely believed that RSA-OAEP is also CCA2-secure, i.e. that it provides security against adaptive chosen ciphertext attacks. In 2001, Shoup [80] showed that the original proof of Bellare and Rogaway does not offer this level of security. However, using an analogous reasoning as in the *stereotyped message attack*, he could easily derive CCA2-security for RSA-OAEP with exponent 3.

In order to prove CCA2-security, we assume the existence of an adversary that successfully attacks RSA-OAEP under chosen ciphertext attacks. This adversary is then used to invert the RSA function. One defines a simulator in order to answer the adversary's decryption and hash queries. Shoup showed that any adversary that never explicitly queries h on s has a negligible probability to pass the format check for the s-part. Thus, one can assume that the first part s has to appear among the attacker's queries. This in turn is already sufficient to extract t as a root of

$$f(t) = (s \cdot 2^k + t)^e - c \bmod N,$$

provided that $|t| < N^{\frac{1}{e}}$ which is fulfilled whenever $k < \log N/e$. This condition is satisfied for $e = 3$ by the RSA-OAEP parameters. One should notice the correspondence to the Relaxed RSA problem: s plays the role of the known message part $\tilde{m}$, whereas t is the small unknown part.

We have reduced the RSA problem to an algorithm for attacking RSA-OAEP. The reduction is tight up to a factor of q_h, the number of hash queries an adversary is allowed to ask. Namely, the running time is q_h times the time to run the LLL-based algorithm for finding small e^{th} roots. The success probability of the RSA inverter is roughly the same as the success probability of the adversary. This reduction is tighter than the original reduction by Bellare-Rogaway for CCA1-security.

RSA-OAEP was shown to be CCA2-secure for arbitrary e by Fujisaki et al [36] in 2001, using a 2-dimensional lattice technique. However, their reduction is also less tight than Shoup's: If the RSA attacker has success probability ε, then the RSA inversion algorithm of [36] has success probability only ε^2.

Another constructive application of Coppersmith's attack on stereotyped messages is used for the definition of an efficient RSA-based pseudorandom number generator (PRNG) in a recent paper by Steinfeld, Pieprzyk and Wang [82], which in turn builds on a work of Fischlin and Schnorr [34]. In the Fischlin-Schnorr RSA-PRNG, one starts with a random seed x_0 and generates a sequence $x_1, x_2, \ldots$ by successively applying the RSA function, i.e. $x_i = x_{i-1}^e \bmod N$. In each iteration, one outputs the r least significant bits of x_i.

In the security proof, Fischlin and Schnorr show that any efficient algorithm that distinguishes the generator's output from the uniform distribution can be used to invert the RSA function, i.e. to solve the RSA problem. However, the reduction is not tight. Namely, if T_D is the running time of the distinguisher, then the inversion algorithm's running time is roughly $2^{2r}T_D$. Therefore, one can only output $r = \mathscr{O}(\log\log N)$ in each iteration in order to preserve a polynomial reduction.

In 2006, Steinfeld, Pieprzyk and Wang showed that one can securely output $\Theta(\log N)$ bits if one replaces the RSA assumption in the Fischlin-Schnorr proof by a relaxed RSA inversion assumption. Namely, we already know that one can recover an $\frac{1}{e}$-fraction of the message from an RSA ciphertext given the rest of the plaintext. Steinfeld et al. make the assumption that this bound is essentially tight. More precisely, they assume that any algorithm that recovers an $\frac{1}{e} + \varepsilon$-fraction for some constant ε already requires at least the same running time as the best factoring algorithm for N.

In fact, one replaces the RSA assumption by a stronger assumption which states that the bound $\frac{1}{e}$ for the Coppersmith attack on stereotyped messages cannot be significantly improved. This stronger assumption is sufficient to increase the generator's output rate from $r = \mathscr{O}(\log\log N)$ to the full-size of $r = \Theta(\log N)$ bits. The efficiency of the Steinfeld, Piepryzk, Wang construction is comparable to the efficiency of the Micali-Schnorr generator [69] from 1988, but uses a weaker assumption than in [69].

Another construction of an efficient PRNG and a MAC based on small root problems was proposed by Boneh, Halevi and Howgrave-Graham [19]. Its security is proved under the hardness of the so-called modular inversion hidden number problem. The best algorithmic bound for attacking this problem is based on an LLL-approach. The security proofs for the PRNG and the MAC again assume that one cannot go significantly beyond this bound.

4.3 Affine Padding: Franklin-Reiter's Attack

The following attack was presented by Franklin and Reiter [35] in 1995. The attack was one year later extended by Coppersmith, Franklin, Patarin and Reiter [27].

Assume that two RSA plaintexts m, m' satisfy an affine relation $m' = m + r$. Let $c = m^3 \bmod N$ and $c' = (m+r)^3 \bmod N$ their RSA ciphertexts, respectively. Franklin and Reiter showed that any attacker with knowledge of c, c', r and N can efficiently recover m by carrying out the simple computation

$$\frac{c'r+2cr-r^4}{c'-c+2r^3} = \frac{3m^3r+3m^2r^2+3mr^3}{3m^2r+3mr^2+3r^3} = m \bmod N.$$

What happens in the case where r is unknown but small?

Affine related messages

Given: $c_1 = m^e \bmod N, c_2 = (m+r)^e \bmod N$ with $|r| \le N^{\frac{1}{e^2}}$
Find : m

If one is able to determine r from the ciphertexts, then m can be computed efficiently. The resultant computation

$$\mathrm{Res}_m(c-m^3, c'-(m+r)^3) = r^9 + 3(c-c')r^6 + 3(c^2+c'^2+7cc')r^3 + (c-c')^3 \bmod N$$

yields a monic univariate polynomial $f(r)$ of degree 9. An application of Theorem 1 shows that r can be recovered as long as $|r| \le N^{\frac{1}{9}}$. For arbitrary e, the bound generalizes to $|r| \le N^{\frac{1}{e^2}}$.

4.4 Related RSA messages: Extending Håstad's attack

Assume that we want to broadcast a plain RSA encrypted message to a group of k receivers all having public exponent e and co-prime moduli $N_1, \ldots, N_k$. I.e., we send the messages $m^e \bmod N_1, \ldots, m^e \bmod N_k$. From this information an attacker can compute $m^e \bmod \prod_{i=1}^k N_i$. If m^e is smaller than the product of the moduli, he can compute m by e^{th} root computation over the integers. If all N_i are of the same bit-size we need $k \ge e$ RSA encrypted messages in order to recover m.

So naturally, an attacker gains more and more information by receiving different encryptions of the same message. Notice that this observation nicely links with the attack on stereotyped RSA messages from Section 4.1. Recall that the cryptanalytic interpretation of the attack in Section 4.1 was that one gets an $\frac{1}{e}$-fraction of the plaintext efficiently. The above broadcast attack can thus be interpreted as an accumulation of this result. If one gets $k \ge e$ times an $\frac{1}{e}$-fraction of m efficiently, then one eventually obtains the whole m.

The question is whether this is still true when the public exponents are different and when the messages are preprocessed by simple padding techniques, e.g. an affine transformation with a fixed known padding pattern. We show that whenever the messages are polynomially related, then the underlying plaintext can still be discovered given sufficiently many encryptions. This result is an extension of Håstad's original result [42] due to May, Ritzenhofen [68].

Assume that the message m is smaller than $\min_j\{N_j\}$. We preprocess the message by known polynomial relations $g_1, \ldots, g_k$ with degrees $\delta_1, \ldots, \delta_k$, respectively.

Polynomially related RSA messages

Given: $c_i = g_i(m)^{e_i} \bmod N_i$ for $i = 1, \ldots, k$ with $\sum_{i=1}^{k} \frac{1}{\delta_i e_i} \geq 1$.
Find : m

Assume that $g_i(x)$ has leading coefficient $a_i \neq 1$. Compute $a_i^{-1} \bmod N_i$. If this computation fails, we obtain the factorization of N_i, which enables us to compute m. Otherwise we replace c_i and $g_i(x)$ by $a_i^{-e_i} c_i$ and $a_i^{-1} g_i(x)$, respectively. This makes all $g_i(x)$ monic.

Let $\delta = \mathrm{lcm}_i\{\delta_i e_i\}$ be the least common multiple of all $\delta_i e_i$. Define $N = \prod_{i=1}^{k} N_i^{\frac{\delta}{\delta_i e_i}}$. We know that for all $i = 1, \ldots, k$ we have

$$(g_i(m)^{e_i} - c_i)^{\frac{\delta}{\delta_i e_i}} = 0 \bmod N_i^{\frac{\delta}{\delta_i e_i}}.$$

Let us compute by Chinese Remaindering a polynomial

$$f(x) = \sum_{i=1}^{k} b_i (g_i(x)^{e_i} - c_i)^{\frac{\delta}{\delta_i e_i}} \bmod N,$$

where the b_i are the Chinese remainder coefficients satisfying $b_i \bmod N_j = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{else} \end{cases}$.

Notice that $f(m) = 0 \bmod N$ and that $f(x)$ is by construction a univariate monic polynomial of degree δ. Let us now upper-bound the size of our desired root m. Using the condition $1 \leq \sum_{i=1}^{k} \frac{1}{\delta_i e_i}$, we obtain

$$m < \min_j\{N_j\} \leq \left(\min_j\{N_j\}\right)^{\sum_{i=1}^{k} \frac{1}{\delta_i e_i}} \leq \prod_{i=1}^{k} N_i^{\frac{1}{\delta_i e_i}}.$$

By applying Theorem 1 with the parameters $\beta, c = 1$, we can find all roots m up to the same bound

$$m \leq N^{\frac{1}{\delta}} = \prod_{i=1}^{k} N_i^{\frac{1}{\delta_i e_i}},$$

which completes the description of the attack.

Let us look at our condition $\sum_{i=1}^{k} \frac{1}{\delta_i e_i} \geq 1$ when we encrypt the plain message m without any further transformation. Then $g_i(x) = x$ is the identity with degree $\delta_i = 1$, i.e. we obtain the simplified condition

$$\sum_{i=1}^{k} \frac{1}{e_i} \geq 1.$$

Again this can be interpreted as an accumulation of the results for stereotyped RSA messages in Section 4.1. Recall that for each encryption of m under exponent e_i, we can compute an $\frac{1}{e_i}$-fraction of m efficiently. This information accumulates such that whenever the sum $\sum_i \frac{1}{e_i}$ of all the fractions exceeds 1, we eventually obtain the whole plaintext m.

4.5 Factoring with High Bits Known

Let $N = pq$, w.l.o.g. $p > q$. Assume that we are given an oracle for the most significant bits of p. Our task is to find the factorization of N in time polynomial in the bit-size of N with a minimal number of queries to the oracle, i.e. we want to minimize the oracle complexity.

One can view this problem as a natural relaxation of the factorization problem. Without knowing any bits of the prime factor p, i.e. without using the oracle, we have to solve the general factorization problem. For the general problem it is unclear whether there exists a polynomial time algorithm in the Turing machine model. So we provide the attacker with an additional sufficiently strong hint given by the oracle answers that allows him to find the factorization in polynomial time.

In 1985, Rivest and Shamir [75] published an algorithm that factors N given a $\frac{2}{3}$-fraction of the bits of p. Coppersmith [22] improved this bound to $\frac{3}{5}$ in 1995. One year later, Coppersmith [24, 25] gave an algorithm using only half of the bits of p.

The *factoring with high bits known* problem can again be reduced to the problem of solving modular univariate polynomial equations with the LLL algorithm. Let us assume that we are given half of the high-order bits of p. Omitting constants, we know an approximation $\tilde{p}$ of p that satisfies $|p - \tilde{p}| \leq N^{\frac{1}{4}}$.

Factorization problem

Given: $N = pq$
Find : p

Relaxed Factorization: High Bits Known

Given: $N = pq, \tilde{p}$ with $|p - \tilde{p}| \leq N^{\frac{1}{4}}$
Find : p

Our goal is to recover the least-significant bits of p, i.e. we want to find the root of the univariate, linear modular polynomial

$$f(x) = \tilde{p} + x \bmod p.$$

Observe that $p - \tilde{p}$ is a root of $f(x)$ with absolute value smaller than $N^{\frac{1}{4}}$.

We apply Theorem 1 with $f(x) = \tilde{p} + x$, i.e. we have degree $\delta = 1$, $\beta = \frac{1}{2}$ and $c = 1$. Therefore, we can find all roots x_0 with size

$$|x_0| \leq N^{\frac{\beta^2}{\delta}} = N^{\frac{1}{4}}.$$

This enables us to recover the low-order bits of p in polynomial time with the LLL algorithm, which yields the factorization.

The *factorization with high bits known* approach can be extended to moduli $N = p^r q$, where p and q have the same bit-size. This extension was proposed by Boneh, Durfee and Howgrave-Graham [18]. For simplicity, we assume that p and q are of the same bit size. For fixed bit-size of N and growing r, these moduli should be – from an information theoretical point of view – easier to factor than usual RSA moduli. Moreover, an attacker should learn from an approximation of p more information than in the standard RSA case. This intuition turns out to be true.
We model this variant of the factorization problem as the univariate polynomial

$$f(x) = (\tilde{p} + x)^r \bmod p^r.$$

Set $\beta = \frac{r}{r+1}$, $\delta = r$ and $c = 1$. An application of Theorem 1 shows that the LLL algorithm recovers all roots x_0 with

$$|x_0| \leq M^{\frac{\beta^2}{\delta}} = N^{\frac{r}{(r+1)^2}}.$$

Since N is roughly of size p^{r+1}, this means that we need an approximation $\tilde{p}$ with $|p - \tilde{p}| \leq p^{\frac{r}{r+1}}$. Or in other words, we need a $\frac{1}{r+1}$-fraction of the most significant bits in order to factor N in polynomial time. I.e., for the RSA case $r = 1$ we need half of the bits, whereas e.g. for $r = 2$ we only need a third of the most significant bits of p. For $r = \Omega(\sqrt{\frac{\log N}{\log\log N}})$, one only has to guess $\mathcal{O}(\log\log N)$ bits of p, which can be done in polynomial time.

4.6 Computing $d \equiv$ Factoring

Our next application of the LLL algorithm addresses the difficulty of computing the RSA secret exponent from the public information (N, e). We show that any algorithm that computes d in *deterministic polynomial time* can be transformed into an algorithm that factors N in *deterministic polynomial time*.

Let $N = pq$ be an RSA-modulus. Let $e, d \in \mathbb{Z}_{\phi(N)}$ be the public/secret exponents, satisfying the equation $ed = 1 \bmod \phi(N)$. If we are given the public information (N, e) and the factorization of N, then d can be computed in polynomial time using the Euclidean algorithm. Rivest, Shamir and Adleman showed that the converse is also true: Given (N, e, d), one can factor N in *probabilistic* polynomial time by an algorithm due to Miller [70].

In 2004, it was shown in [66, 30] that there is also a *deterministic* reduction of factoring to computing d using Coppersmith's method. This establishes the *deterministic polynomial time equivalence* of both problems.

It is not hard to see that the knowledge of $\phi(N) = N - (p+q-1)$ yields the factorization of N in polynomial-time. Our goal is to compute $\phi(N)$. Since p, q are of the same bit-size, the term N is an approximation of $\phi(N)$ up to roughly $N^{\frac{1}{2}}$. Therefore, the polynomial

$$f(x) = N - x \mod \phi(N)$$

has a root $x_0 = p+q-1$ of size $N^{\frac{1}{2}}$. Let $M = ed - 1 = N^{\alpha}$ for some $\alpha \leq 2$. We know that M is a multiple of $\phi(N)$.

Now, we can apply the LLL algorithm via Theorem 1 with the parameter setting $\delta, c = 1$, $b = \phi(N)$, $M = N^{\alpha}$ the integer of unknown factorization and $\beta = \frac{1}{\alpha}$. We conclude that we can find all roots x_0 within the bound

$$|x_0| \leq M^{\frac{\beta^2}{\delta}} = (N^{\alpha})^{\frac{1}{\alpha^2}} = N^{\frac{1}{\alpha}}.$$

Since $\alpha \leq 2$, we can find all roots within the bound $N^{\frac{1}{2}}$, as desired.

4.7 Finding Smooth Numbers and Factoring

The following link between finding smooth integers with Coppersmith's LLL-based algorithm and factoring composite integers N was introduced by Boneh [12] in 2001.

Many classic factorization algorithms such as the Quadratic Sieve and the Number Field Sieve have to find values slightly larger than $\sqrt{N}$ such that their square modulo N is *B-smooth*. A number is called B-smooth if it splits into prime factors $p_1, p_2, \ldots, p_n$ smaller than B. We can model this by a univariate polynomial equation

$$f_c(x) = (x + \sqrt{cN})^2 - cN,$$

for small values of c. Given an interval size X, the task is to find all solutions $|x_0| \leq X$ such that $f_c(x_0)$ has a large B-smooth factor. Whenever this factor is as large as $f_c(x_0)$ itself, then $f_c(x_0)$ factors completely over the factor base $p_1, \ldots, p_n$.

Finding Integers with Large Smooth Factor

Given: $f_c(x), B, X$
Find : $|x_0| \leq X$ such that $f_c(x_0)$ has a large B-smooth factor.

Let us define $P = \prod_{i=1}^{n} p_i^{e_i}$. For simplicity reasons, we will assume here $e_i = 1$ for all exponents, although we could handle arbitrary multiplicities as well. We are interested in integers x_0 such that many p_i divide $f_c(x_0)$, i.e. $f_c(x_0) = 0 \bmod b$ for a modulus $b = \prod_{i \in I} p_i$, where $I \subseteq \{1, \ldots, n\}$ is a large index set.

Applying Theorem 1, it is easy to see that $b \geq P^{\sqrt{\frac{2\log X}{\log P}}}$ is sufficient to find all

$$|x_0| \leq P^{\frac{\beta^2}{\delta}} = P^{\frac{2\log X}{2\log P}} = 2^{\log X} = X.$$

Boneh [12] illustrates his result by giving numerical examples were just one application of LLL on a 50-dimensional lattice yields all numbers in an interval of size $X = 2^{500}$ that have a sufficiently large smooth factor.

At the moment however, the technique does not lead to improvements to classical factorization algorithms, since it is unlikely that randomly chosen intervals of the given size contain sufficiently many smooth numbers. Moreover, classical algorithms usually need fully smooth numbers, whereas with the present method one only finds numbers with a large smooth factor.

5 Applications of Finding Roots of Multivariate Equations

In this section, we study applications of the LLL algorithm for solving multivariate polynomial equations. We start by presenting the two most famous RSA applications for solving bivariate modular polynomial equations: The attacks of Wiener [88] and Boneh-Durfee [14] on RSA with small secret exponent d.

RSA Key Recovery Problem

Given: N, e
Find : d with $ed = 1 \bmod \phi(N)$

Relaxed RSA Key Recovery Problem: Small key

Given: N, e with $ed = 1 \bmod \phi(N)$ for some $d \leq N^{\delta}$
Find : d

Let us briefly describe Wiener's polynomial time attack on RSA for secret keys $d \leq N^{\frac{1}{4}}$. Although this attack was originally presented using continued fractions, we will describe it within the framework of small solutions to linear bivariate equations. We can write the RSA key equation $ed = 1 \bmod \phi(N)$ in the form

$$ed + k(p+q-1) - 1 = kN, \tag{5}$$

for some $k \in \mathbb{N}$. This leads to a linear bivariate polynomial $f(x,y) = ex + y$ that has the root $(x_0, y_0) = (d, k(p+q-1)-1)$ modulo N. It is not hard to see that $k < d$. In the case of balanced prime factors, we have $p+q \approx \sqrt{N}$. For $d \leq N^{\frac{1}{4}}$, the product $x_0 y_0$ of the desired roots can therefore be upper-bounded by N.

It is well-known that linear modular polynomial equations can be heuristically solved by lattice reduction whenever the product of the unknowns is smaller than the modulus. For the bivariate case, this lattice technique can be made rigorous. In our case, one has to find a shortest vector in the lattice L spanned by the row vectors of the following lattice basis

$$B = \begin{pmatrix} NX & 0 \\ eX & Y \end{pmatrix}, \quad \text{where } X = N^{\frac{1}{4}} \text{ and } Y = N^{\frac{3}{4}}.$$

Using an argumentation similar to the one in Section 2, one can see that a shortest vector $v = (c_0, c_1) \cdot B$ yields a polynomial $c_0 Nx + c_1 f(x,y)$ that evaluates to zero over the integers at the point $(x_0, y_0) = (d, k(p+q-1)-1)$. Since $f(x_0, y_0) = kN$, we have

$$c_0 Nd = -c_1 Nk.$$

Because v is a shortest vector, the coefficients c_0 and c_1 must be co-prime. Therefore, we conclude that $|c_0| = k$ and $|c_1| = d$. From this information, we can derive via Eq. (5) the term $p+q$ which in turn yields the factorization of N in polynomial time.

Instead of using a two-dimensional lattice, one could compute the tuple (k,d) by looking at all convergents of the continued fraction expansion of e and N. This approach was taken in Wiener's original work.

In 1999, Boneh and Durfee improved Wiener's bound to $d \leq N^{1-\sqrt{\frac{1}{2}}} \approx N^{0.292}$. This result was achieved by writing the RSA equation as

$$k(N+1-(p+q))+1 = ed.$$

This in turn yields a bivariate polynomial $f(x,y) = x(N+1-y)+1$ with the root $(x_0, y_0) = (k, p+q)$ modulo e. Notice that f has the monomials x, xy and 1. As in Wiener's attack, the product $x_0 \cdot x_0 y_0$ can be bounded by N whenever $d \leq N^{\frac{1}{4}}$. Thus for e of size roughly N, we obtain the same bound as in the Wiener attack if we linearize the polynomial. However, Boneh and Durfee used the polynomial structure of $f(x,y)$ in order to improve the bound to $N^{0.292}$ by a Coppersmith-type approach.

Wiener as well as Boneh and Durfee posed the question whether there is also a polynomial time attack for RSA with small secret CRT-exponent d. We call d a small CRT-exponent if the values $d_p = d \bmod p-1$ and $d_q = d \bmod q-1$ are small. This enables a receiver to efficiently decrypt modulo p and q and combine the results using the Chinese remainder theorem (CRT) [74].

RSA Key Recovery Problem

Given: N, e
Find : d with $ed = 1 \bmod \phi(N)$

Relaxed RSA Key Recovery Problem: Small CRT-key

Given: N, e with $ed_p = 1 \bmod p-1$ and $ed_q = 1 \bmod q-1$ for $d_p, d_q \leq N^{\delta}$
Find : d with $d = d_p \bmod p-1$ and $d = d_q \bmod q-1$

Recently, Jochemsz and May[52] presented a polynomial time attack for RSA with $d_p, d_q \leq N^{0.073}$, building on an attack of Bleichenbacher and May [5]. The basic idea is to write the RSA key equation in the form

$$\left| \begin{array}{l} ed_p + k_p - 1 = k_p p \\ ed_q + k_q - 1 = k_q q \end{array} \right|,$$

with the unknowns d_p, d_q, k_p, k_q, p and q. We eliminate the unknowns p, q by multiplying both equations. Rearranging terms yields

$$e^2 d_p d_q + e(d_p(k_q - 1) + d_q(k_p - 1)) + k_p k_q (1 - N) + (k_p + k_q + 1) = 0.$$

In [5], the authors linearize this equation and derive attacks for variants of the RSA cryptosystem where e is significantly smaller than N. In [52], the full polynomial structure is exploited using a Coppersmith technique in order to extend the linearization attack to full size e.

By assigning the variables x_1, x_2, x_3, x_4 to the unknowns d_p, d_q, k_p, k_q, respectively, one obtains a 4-variate polynomial equation which evaluates to zero over the integers. A Coppersmith-type analysis results in a heuristic polynomial time attack that works for $d_p, d_q \leq N^{0.073}$.

Several results in the literature also address the inhomogenous case of small RSA secret key relaxations, where d is not small but parts of d's bits are known to an attacker. Boneh, Durfee and Frankel introduced several of these so-called Partial Key Exposure attacks, which were later extended in Blömer, May [7] and EJMW [33]. In the latter work, the authors showed that the Boneh-Durfee attack naturally extends to the inhomogenous case for all d smaller than $\phi(N)$. The larger d is, the more bits of d an attacker has to know (see Fig. 1).

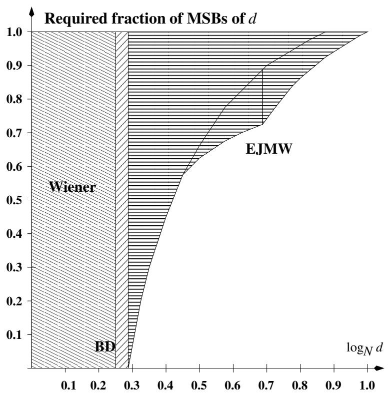

Fig. 1 Partial Key Exposure attack

Again, the former cryptanalytic results have a dual interpretation as security results. They establish the security of certain parts of the bits of the RSA secret key. More precisely, the results state that recovering these bits is as hard as factoring the RSA modulus given only the public information (N, e). This opens the possibility to publish the remaining bits of the secret key, which can be used e.g. in server-aided RSA systems, where parts of an RSA signature computation are outsourced to an untrusted server. This dual application was first proposed by Boneh, Durfee and Frankel [18]. Later, Steinfeld and

Zheng [83] proposed another server-based RSA system, which provides provable security against Partial Key Exposure attacks.

6 Survey and References for LLL-based RSA and Factoring Results

The following table gives an overview and references of various applications of Coppersmith's LLL-based methods for finding small roots when applied to relaxed RSA or factorization problems. Although not all of these results are originally described in terms of small root problems, they all more or less fit in this framework and might serve as useful pointers for further reading.

Method	
Method/variants	Håstad 88[42], Girault,Toffin,Vallée 88[37] Coppersmith 96,97,01[23, 24, 25, 26], Howgrave-Graham 98,01[48, 49], Jutla 98[53], May 03[65], Bernstein 04[3], Coron 05,07[28, 29], Bauer, Joux 07[1]
Optimize bounds	Blömer,May[10] , Jochemsz, May[52]
RSA	
Inverting RSA	Håstad 89[42], Coppersmith 96[23, 25], May, Ritzenhofen 08[68]
Small d	Wiener 90[88], Boneh, Durfee 98[14, 15], Durfee, Nguyen 00[32], Blömer, May 01[9], de Weger 02[87], Hinek 02[45], May 04[67]
Known bits of d	Boneh, Durfee, Frankel 96[16], Blömer, May 03[7], Ernst, Jochemsz, May, de Weger 05[33]
Key recovery	May 04[66], Coron, May 07[30], Kunihiro, Kurosawa 07[57]
Small CRT-d	May 02[64], Hinek, Sun, Wu 05[85], Galbraith, Heneghan, McKee 05[38, 39], Bleichenbacher, May 06[5], Jochemsz, May 06[51, 52]
Proving Security	Shoup 01[80], Boneh 01[13], Steinfeld, Zheng 04[83]
PRNG, MAC	Boneh, Halevi, Howgrave-Graham 99[19], Steinfeld, Pieprzyk, Wang 06[82]
Factoring	
High Bits known	Rivest, Shamir 86[75], Coppersmith 95,96[22, 23], Boneh, Durfee, Howgrave-Graham 99[18], Crépeau, Slakmon 03[31], Santoso, Kunihiro, Kanayama, Ohta 06[77], Herrmann, May 08[44]
Finding relations	Schnorr 01[78], Boneh 00[12]

7 Open Problems & Speculations

7.1 Optimizing bounds: On Newton polytopes and Error Terms

In this section, we will explain how to optimize the upper bounds up to which small roots can be found. Here, a polynomial's Newton polytope will play a central role. We will also see that the upper bounds usually incorporate some error term, which in many cases can be eliminated by splitting the search interval into smaller pieces and treating each subinterval separately (see e.g. the proof of Theorem 1).

In all applications of Coppersmith's method, one starts with either a polynomial modular equation $f(x_1,\ldots,x_m) = 0 \bmod b$ or a polynomial integer equation $f(x_1,\ldots,x_m) = 0$. Using this equation, one defines algebraic multiples $f_1,\ldots,f_n$ of f which contain the same small roots. For instance, if f is a univariate polynomial equation in x as in Section 2, this is done by multiplying f with powers of x and by taking powers of f itself. In the univariate case, it is clear which set of algebraic multiples maximizes the size of the roots x_0 that we can efficiently recover. Indeed, we will argue in Section 7.3, that the bound $|x_0| \le N^{\frac{\beta^2}{\delta}}$ from Section 2 cannot be improved in general, since beyond this bound f may have too many roots to output them in polynomial time.

For univariate modular polynomial equations $f(x)$, one looks for an integer linear combination $g(x) = \sum_i a_i f_i(x)$, $a_i \in \mathbb{Z}$, such that $g(x_0) = 0$ over the integers for all sufficiently small roots. These roots can then be found by standard root finding methods.

For irreducible bivariate polynomials $f(x,y)$, one similarly defines algebraic multiples $f_1(x,y),\ldots,f_n(x,y)$. The goal is to find a polynomial $g(x,y) = \sum_i a_i f_i(x,y)$ by LLL-reduction such that $g_i(x,y)$ is not a multiple of $f(x,y)$. Then the roots can be found by resultant computations.

Whereas the choice of the algebraic multiples is quite straightforward for univariate polynomials, for multivariate polynomials the choice of the algebraic multiples appears to be a complex optimization problem. The bounds for the roots that one computes mainly depend on the largest coefficient of the polynomial and the polynomial's Newton polytope - i.e. the convex hull of the monomials' exponents when regarded as points in the Euclidean space.

Let us give an example for this. As explained in Section 4, we can factor $N = pq$ with known high bits of p by using the univariate polynomial equation $f(x) = \tilde{p} + x \bmod p$, where $\tilde{p}$ is an approximation of p up to $N^{\frac{1}{4}}$. The same result can be achieved by computing $\tilde{q} = \frac{N}{\tilde{p}}$ and solving the bivariate integer polynomial $f(x,y) = (\tilde{p}+x)(\tilde{q}+y) - N$. The largest coefficient in this equation is $\tilde{p}\tilde{q} - N$, which is roughly of size $W = N^{\frac{3}{4}}$. The monomials of $f(x,y)$ are $1, x, y$ and xy, i.e. the Newton polytope is a square defined by the points $(0,0), (0,1), (1,0)$ and $(1,1)$. Optimizing the upper bounds X, Y for the size of the roots in x, y, respectively, yields the condition $XY \le W^{\frac{2}{3}}$. This is equivalent to $XY \le N^{\frac{1}{2}}$ or $X, Y \le N^{\frac{1}{4}}$. Thus, we achieve the same result as in the univariate modular case.

We could however also look at the bivariate polynomial $f(x,y) = (\tilde{p}+x)y - N$. The largest coefficient is $W = N$ and the Newton polytope defined by $(0,0),(0,1),(1,1)$ is a triangle. Optimizing the bounds for this shape of the Newton polytope yields the condition $(XY)^4 \leq W^3$. Setting $Y = N^{\frac{1}{2}}$ and $W = N$ yields $X^4 \leq N$ which leads again to $X \leq N^{\frac{1}{4}}$.

Interestingly, we do not need the approximation of q for achieving the same result. Since we do not need the bits of q, one should ask whether we indeed need to know the bits of p. Let us look at the polynomial equation $f(x,y) = xy - N$, where $W = N$ and the Newton polytope is a line formed by $(0,0)$ and $(1,1)$. Applying a Coppersmith-type analysis to this polynomial yields the bound $XY \leq W^{1-\varepsilon} = N^{1-\varepsilon}$, for some error term ε. Notice that a bound of $XY \leq 2N$ would easily allow to factor $N = pq$ if p,q have equal bit-size, since $p \leq \sqrt{N} = X$ and $q \leq 2\sqrt{N} = Y$.

What does the bound $XY \leq N^{1-\varepsilon}$ imply? Can we remove the error term ε and derive the desired bound by running the algorithm on $2N^{\varepsilon}$ copies, where we search in each copy for the roots in an interval of size $N^{1-\varepsilon}$? I.e. can we factor in time $\tilde{\mathcal{O}}(N^{\varepsilon})$? And provided that the error term ε satisfies $\varepsilon = \mathcal{O}(\frac{1}{\log N})$, can we factor in polynomial time?

(Un)fortunately, the answer is NO, at least with this approach. The reason is that as opposed to other polynomials, we cannot simply guess a few bits of the desired small root $(x_0,y_0) = (p,q)$, since this would either change the structure of the Newton polytope or the size of W. If we guess bits of x_0, we introduce the monomial y and symmetrically for y_0 we introduce the x-monomial. But as shown above, this changes our bound to an inferior $XY \leq N^{\frac{3}{4}}$. On the other hand, if we guess bits of x_0y_0 our largest coefficient decreases accordingly.

Notice that e.g. for the polynomial $(\tilde{p}+x)y - N$ guessing bits of x_0 is doable since the guessing does not introduce new monomials. Thus, in this case a small error term in the bound can be easily eliminated by a brute-force search technique.

7.2 Applying the Method to Multivariate Polynomial Equations

Another challenging problem is to obtain provability of the algorithm in the multivariate setting. This problem is not only of theoretical interest. There have been cases reported, where the heuristic method – which computes the roots by resultant computations – for multivariate polynomials systematically fails [9].

Let us see why the method provably works for the bivariate integer case and what causes problems when extending it to a third variable. Coppersmith's original method for bivariate integer polynomials constructs on input $f(x,y)$ a polynomial $g(x,y)$, such that $g(x,y)$ cannot be a polynomial multiple of $f(x,y)$. In other words, $g(x,y)$ does not lie in the ideal $\langle f \rangle$ generated by f and therefore the resultant of f and g cannot be the zero polynomial.

Heuristically, one extends this approach to three variables by constructing two polynomials g_1, g_2 with LLL-reduction. The resultants $r_1 = \mathrm{Res}(f, g_1)$ and $r_2 =$

$\mathrm{Res}(f, g_2)$ are bivariate polynomials. The resultant $\mathrm{Res}(r_1, r_2)$ is then univariate and yields one coordinate of the roots, provided that the resultant does not vanish. The other coordinates of the roots can be found by back-substitution. The resultant is non-vanishing iff g_1 and g_2 are algebraically independent.

Recently, Bauer and Joux [1] proposed a twist in the above construction which in some cases enables to guarantee algebraic independence also for polynomials in three and more variables. Basically, their approach is an iterative application of Coppersmith's original technique for bivariate polynomials. Given a trivariate polynomial $f(x,y,z)$, one constructs a polynomial $g(x,y,z)$ such that g does not lie in $\langle f \rangle$. Afterwards, one uses a Gröbner Basis approach and another iteration of the LLL procedure to construct a third polynomial $h(x,y,z)$, which does not lie in $\langle f, g \rangle$.

Unfortunately, Bauer and Joux's approach still incorporates a heuristic assumption. For trivariate polynomials of a special shape however, the approach can be made fully rigorous.

7.3 What Are the Limitations of the Method?

Coppersmith's method outputs *all* sufficiently small solutions of a polynomial equation. Since the method runs in polynomial time, it can only output a polynomial number of solutions. Thus, the method proves in a constructive way a limit for the number of roots within a certain interval. This limit matches for univariate modular polynomials the bounds by Konyagin, Steeger [56]. The number of roots of each polynomial equation thus limits the size of the interval that we are able to search through in polynomial time. Let us demonstrate this effect for univariate modular polynomial equations.

Let $N = p^r$. Assume that we want to solve the equation $f(x) = x^r \mod N$. Clearly, all $x_0 = kp, k \in \mathbb{N}$, are solutions. Hence solving this equation for solutions $|x_0| \le p^{1+\varepsilon}$ would imply that one has to output p^{ε} solutions, an exponential number.

This argument serves as an explanation why the bound $|x_0| = N^{\frac{1}{\delta}}$ from Section 2 cannot be improved in general. On the other hand, for the following two reasons this argument does not fundamentally rule out improvements for any of the applications' current bounds mentioned in this survey.

First, the factorization of $N = p^r$ can be easily determined. Hence, there might be an improved method which exploits this additional information. Indeed, Bleichenbacher and Nguyen [6] describe a lattice-based method for *Chinese remaindering with errors* that goes beyond the Coppersmith-type bound in cases where the factorization of the modulus is known.

Second, in all the applications we studied so far, an improvement of the bound would not immediately imply an exponential number of solutions. Look for instance at the *factoring with high bits problem* and let us take the polynomial $f(x) = \tilde{p} + x \bmod p$. The solution of this polynomial is unique up to the bound $|x_0| \le p$. So although we have no clue how to solve the factorization problem with the help of lattice reduction techniques, there is also no limiting argument which tells us that it is impossible to extend our bounds to the general case.

As a second example, look at the Boneh-Durfee attack on RSA with $d \leq N^{0.292}$ which introduces the bivariate polynomial equations $f(x,y) = x(N+1-y)+1 \bmod e$. Assume that e is roughly of size N. Since y is the variable for $p+q$, its size can be roughly bounded by $\sqrt{N}$. Assume that for a fixed candidate y the mapping $g : x \mapsto x(N+1-y)+1 \bmod e$ takes on random values in $\mathbb{Z}_e$. If we map $\sqrt{N}$ candidates for x for every of the $\sqrt{N}$ choices of y, we expect to map to zero at most a constant number of times.

This counting argument let Boneh and Durfee conjecture that one can achieve a bound of $d \leq \sqrt{N}$ in polynomial time attacks on RSA with small secret d. Moreover, if one used the fact that y represents $p+q$, which implies that y_0 is already fully determined by N, then the counting argument would not rule out a bound beyond $\sqrt{N}$. If we could make use of this information about y_0, then there would be a unique candidate for x_0 in $\mathbb{Z}_{\phi(N)}$ and recovering this candidate would solve the RSA problem as well as the factorization problem. However, despite considerable research efforts the bound $d \leq N^{0.292}$ is still the best bound known today. It remains an open problem to further push it.

8 Summary

The invention of the LLL algorithm in 1982 was the basis for the construction of an efficient algorithm due to Coppersmith for finding small solutions of polynomial equations in 1996. This in turn opened a whole new line of research and enabled new directions for tackling challenging problems such as the RSA problem or the factorization problem from a completely different angle. As opposed to traditional approaches like the Elliptic Curve Method and the Number Field Sieve, the LLL-based approach is polynomial time but solves only relaxed versions of the RSA and the factorization problem.

Today, the relaxed versions are still pretty far away from the general instances. But there appears to be a steady progress in finding new interesting applications and the existing bounds are continuously pushed. From a research point of view, it is likely that the young field of LLL-based root finding still hides many fascinating results that await their discovery.

Acknowledgments: The author thanks Mathias Herrmann, Ellen Jochemsz, Phong Nguyen, Maike Ritzenhofen and Damien Stehlé for comments and discussions.

References

1. A. Bauer, A. Joux, "Toward a rigorous variation of Coppersmith's algorithm on three variables", Advances in Cryptology – Eurocrypt 2007, Lecture Notes in Computer Science, Springer-Verlag, 2007
2. M. Bellare, P. Rogaway, "Optimal Asymmetric Encryption", Advances in Cryptology – Eurocrypt '94, Lecture Notes in Computer Science Vol. 950, Springer-Verlag, pp. 92–111, 1994
3. D. Bernstein, "Reducing lattice bases to find small-height values of univariate polynomials."Algorithmic number theory, 2004
4. D. Bleichenbacher, "On the Security of the KMOV public key cryptosystem", Advances in Cryptology – Crypto '97, Lecture Notes in Computer Science Vol. 1294, Springer-Verlag, pp. 235–248, 1997
5. D. Bleichenbacher, A. May, "New Attacks on RSA with Small Secret CRT-Exponents", Practice and Theory in Public Key Cryptography – PKC 2006, Lecture Notes in Computer Science Vol. 3958, pp. 1–13, Springer-Verlag, 2006
6. D. Bleichenbacher, P.G. Nguyen, "Noisy Polynomial Interpolation and Noisy Chinese Remaindering", Advances in Cryptology – Eurocrypt 2000, Lecture Notes in Computer Science, pp. 53–69, Springer-Verlag, 2000
7. J. Blömer, A. May, "New Partial Key Exposure Attacks on RSA", Advances in Cryptology – Crypto 2003, Lecture Notes in Computer Science Vol. 2729, pp. 27–43, Springer-Verlag, 2003
8. J. Blömer, A. May, "A Generalized Wiener Attack on RSA", Practice and Theory in Public Key Cryptography – PKC 2004, Lecture Notes in Computer Science Vol. 2947, pp. 1–13, Springer-Verlag, 2004
9. J. Blömer, A. May, "Low Secret Exponent RSA Revisited", Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science Volume 2146, Springer-Verlag, pp. 4–19, 2001.
10. J. Blömer, A. May, "A Tool Kit for Finding Small Roots of Bivariate Polynomials over the Integers", Advances in Cryptology – Eurocrypt 2005, Lecture Notes in Computer Science Vol. 3494, pp. 251–267, Springer-Verlag, 2005
11. D. Boneh, "Twenty years of attacks on the RSA cryptosystem", Notices of the AMS, 1999
12. D. Boneh, "Finding smooth integers in short intervals using CRT decoding", STOC, pp. 265–272, 2000
13. D. Boneh, "Simplified OAEP for the RSA and Rabin Functions", Advances in Cryptology – Crypto 2001, Lecture Notes in Computer Science Vol. 2139, pp. 275–291, Springer-Verlag, 2001
14. D. Boneh, G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$", Advances in Cryptology – Eurocrypt'99, Lecture Notes in Computer Science Vol. 1592, Springer-Verlag, pp. 1–11, 1999.
15. D. Boneh, G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$", IEEE Trans. on Information Theory, Vol. 46(4), pp. 1339–1349, 2000
16. D. Boneh, G. Durfee, Y. Frankel, "An attack on RSA given a small fraction of the private key bits", Advances in Cryptology – Asiacrypt '98, Lecture Notes in Computer Science Vol. 1514, Springer-Verlag, pp. 25–34, 1998
17. D. Boneh, G. Durfee, Y. Frankel, "Exposing an RSA Private Key Given a Small Fraction of its Bits", Full version of the work from Asiacrypt'98, available at `http://crypto.stanford.edu/~dabo/abstracts/bits_of_d.html`, 1998
18. D. Boneh, G. Durfee, and N. Howgrave-Graham, "Factoring $N = p^r q$ for large r", Advances in Cryptology – Crypto '99, Lecture Notes in Computer Science Vol. 1666, Springer-Verlag, pp. 326–337, 1999
19. D. Boneh, S. Halevi, N. Howgrave-Graham, "The Modular Inversion Hidden Number Problem", Advances in Cryptology – Asiacrypt 2001, Lecture Notes in Computer Science Vol. 2248, Springer-Verlag, pp. 36–51, 2001
20. D. Boneh, Venkatesan, " Breaking RSA may not be equivalent to factoring", Advances in Cryptology – Eurocrypt '98, Lecture Notes in Computer Science Vol. 1233, Springer-Verlag, pp. 59–71, 1998

21. D. Brown, "Breaking RSA May Be As Difficult As Factoring", Cryptology ePrint Archive Report 2005/380, 2005
22. D. Coppersmith, "Factoring with a hint", IBM Research Report RC 19905, 1995
23. D. Coppersmith, "Finding a Small Root of a Univariate Modular Equation", Advances in Cryptology – Eurocrypt '96, Lecture Notes in Computer Science Vol. 1070, Springer-Verlag, pp. 155–165, 1996
24. D. Coppersmith, "Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known", Advances in Cryptology – Eurocrypt '96, Lecture Notes in Computer Science Vol. 1070, Springer-Verlag, pp. 178–189, 1996
25. D. Coppersmith, "Small solutions to polynomial equations and low exponent vulnerabilities", Journal of Cryptology, Vol. 10(4), pp. 223–260, 1997.
26. D. Coppersmith, "Finding Small Solutions to Small Degree Polynomials", Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science Volume 2146, Springer-Verlag, pp. 20–31, 2001.
27. D. Coppersmith, M. K. Franklin, J. Patarin, and M. K. Reiter, "Low-Exponent RSA with Related Messages", Advances in Cryptology – Eurocrypt '96, Lecture Notes in Computer Science Vol. 1070, Springer-Verlag, pp. 1–9, 1996
28. J.-S. Coron, "Finding Small Roots of Bivariate Integer Polynomial Equations Revisited", Advances in Cryptology – Eurocrypt 2005, Lecture Notes in Computer Science Vol. 3027, Springer-Verlag, 2005
29. J.-S. Coron, "Finding Small Roots of Bivariate Integer Polynomial Equations: A Direct Approach", Advances in Cryptology – Crypto 2007, Lecture Notes in Computer Science, Springer-Verlag, 2007
30. J.-S. Coron, A. May, "Deterministic Polynomial Time Equivalent of Computing the RSA Secret Key and Factoring", Journal of Cryptology, 2007
31. C. Crépeau, A. Slakmon, "Simple Backdoors for RSA Key Generation", Topics in Cryptology – CT-RSA 2003, Lecture Notes in Computer Science Vol. 2612, pp. 403–416, Springer-Verlag, 2003
32. G. Durfee, P. Nguyen, "Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt '99", Advances in Cryptology – Asiacrypt 2000, Lecture Notes in Computer Science Vol. 1976, Springer, pp. 14–29, 2000
33. M. Ernst, E. Jochemsz, A. May, B. de Weger, "Partial Key Exposure Attacks on RSA up to Full Size Exponents", Advances in Cryptology – Eurocrypt 2005, Lecture Notes in Computer Science Vol. 3494, pp. 371–386, Springer-Verlag, 2005
34. R. Fischlin, C.-P. Schnorr, "Stronger Security Proofs for RSA and Rabin Bits", Journal of Cryptology Vol. 13(2), pp. 221–244, 2000
35. M. K. Franklin, M. K. Reiter, "A linear protocol failure for RSA with exponent three", Rump Session Crypto '95, 1995
36. E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern, "RSA-OAEP Is Secure under the RSA Assumption", Advances in Cryptology – Crypto 2001, Lecture Notes in Computer Science Vol. 2139, Springer-Verlag, pp. 260–274, 2001
37. M. Girault, P. Toffin, B. Vallée, "Computation of Approximate L-th Roots Modulo n and Application to Cryptography", Advances in Cryptology – Crypto 1988, Lecture Notes in Computer Science Vol. 403, pp. 100-117, 1988
38. S. D. Galbraith, C. Heneghan, and J. F. McKee, "Tunable Balancing of RSA", Proceedings of ACISP 2005, Lecture Notes in Computer Science Vol. 3574, pp. 280–292, 2005
39. S. D. Galbraith, C. Heneghan, and J. F. McKee, "Tunable Balancing of RSA", full version of [38], online available at `http://www.isg.rhul.ac.uk/~sdg/full-tunable-rsa.pdf`
40. G. Gentry, "The Geometry of Provable Security: Some Proofs of Security in which Lattices Make a Surprise Appearance", LLL+25 Conference in honour of the 25th birthday of the LLL algorithm, 2007
41. V. Guruswami, M. Sudan, "Improved decoding for Reed-Solomon and algebraic geometric codes", IEEE Trans. on Information Theory, Vol. 45(6), pp. 1757–1767, 1999

42. J. Håstad, “Solving Simultaneous Modular Equations of Low Degree”, Siam J. Computing, 1988
43. J. Håstad, M. Näslund, “The security of all RSA and discrete log bits.”, Journal of the ACM Vol. 51(2), pp. 187–230, 2004
44. M. Herrmann, A. May, “Solving Linear Equations Modulo Divisors: On Factoring Given Any Bits”, Advances in Cryptology – Asiacrypt 2008, Lecture Notes in Computer Science , Springer, 2008
45. M.J. Hinek, “Another Look at Small RSA Exponents”, Topics in Cryptology – CT-RSA 2006, Lecture Notes in Computer Science Vol. 3860, pp. 82–98, 2006
46. M. van Hoeij, “Factoring Polynomials and 0-1 Vectors”, Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science Vol. 2146, Springer-Verlag, pp. 45–50, 2001
47. M. van Hoeij, “Factoring polynomials and the knapsack problem”, J. Number Theory Vol. 95, pp. 167–189, 2002
48. N. Howgrave-Graham, “Finding small roots of univariate modular equations revisited”, Proceedings of Cryptography and Coding, Lecture Notes in Computer Science Vol. 1355, Springer-Verlag, pp. 131–142, 1997
49. N. Howgrave-Graham, “Computational Mathematics Inspired by RSA”, PhD thesis, University of Bath, 1998
50. N. Howgrave-Graham, “Approximate Integer Common Divisors”, Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science Vol. 2146, Springer-Verlag, pp. 51–66, 2001
51. E. Jochemsz, A. May, “A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants”, Advances in Cryptology – Asiacrypt 2006, Lecture Notes in Computer Science Vol. 4284, pp. 267–282, Springer-Verlag, 2006
52. E. Jochemsz, A. May, “A Polynomial Time Attack on Standard RSA with Private CRT-Exponents Smaller than $N^{0.073}$”, Advances in Cryptology – Crypto 2007, Lecture Notes in Computer Science Vol. 4622, pp. 395–411, Springer-Verlag, 2007
53. C. Jutla, “On finding small solutions of modular multivariate polynomial equations”, Advances in Cryptology – Eurocrypt ’98, Lecture Notes in Computer Science Vol. 1403, Springer-Verlag, pp. 158–170, 1998
54. S. Katzenbeisser, “Recent Advances in RSA Cryptography”, Advances in Information Security, Kluwer Academic Publishers, 2001
55. J. Klüners, “The van Hoeij algorithm for factoring polynomials”, LLL+25 Conference in honour of the 25th birthday of the LLL algorithm, 2007
56. S.V. Konyagin, T. Steger, “On polynomial congruences”, Mathematical Notes Vol. 55(6), pp.596–600, 1994
57. N. Kunihiro, K. Kurosawa, “Deterministic Polynomial Time Equivalence between Factoring and Key-Recovery Attack on Takagi’s RSA”, Practice and Theory in Public Key Cryptography – PKC 2007, Lecture Notes in Computer Science, Springer-Verlag, 2007
58. G. Leander, A. Rupp, “On the Equivalence of RSA and Factoring Regarding Generic Ring Algorithms”, Advances in Cryptology – Asiacrypt 2006, Lecture Notes in Computer Science Vol. 4284, pp. 241–251, Springer-Verlag, 2006
59. H. W. Lenstra, “Factoring Integers with Elliptic Curves”, Mathematische Annalen, Vol. 126, pp. 649–673, 1987
60. A.K. Lenstra, H.W. Lenstra, “The Development of the Number Field Sieve”, Springer-Verlag, 1993
61. A. K. Lenstra, H. W. Lenstra, and L. Lovász, ”Factoring polynomials with rational coefficients,” Mathematische Annalen, Vol. 261, pp. 513–534, 1982
62. A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, and J. M. Pollard, “The number field sieve”, In Proceedings of the 22nd Annual ACM Symposium on the Theory of Computation, pages 564–572, 1990
63. Mandelbaum, “Further results on decoding arithmetic residue codes”, IEEE Transaction on Information Theory, Vol. 24(5), pp.643–644, 1978

64. A. May, “Cryptanalysis of Unbalanced RSA with Small CRT-Exponent”, Advances in Cryptology – Crypto 2002, Lecture Notes in Computer Science Vol. 2442, Springer-Verlag, pp. 242–256, 2002
65. A. May, “New RSA Vulnerabilities Using Lattice Reduction Methods”, PhD thesis, University of Paderborn, 2003
66. A. May, “Computing the RSA Secret Key is Deterministic Polynomial Time Equivalent to Factoring”, Advances in Cryptology (Crypto 2004), Lecture Notes in Computer Science Volume 3152, pages 213–219, Springer-Verlag, 2004.
67. A. May, “Secret Exponent Attacks on RSA-type Schemes with Moduli $N = p^r q$”, Practice and Theory in Public Key Cryptography – PKC 2004, Lecture Notes in Computer Science Vol. 2947, Springer-Verlag, pp. 218–230, 2004
68. A. May, M. Ritzenhofen, “Solving Systems of Modular Equations in One Variable: How Many RSA-Encrypted Messages Does Eve Need to Know?”, Practice and Theory in Public Key Cryptography – PKC 2008, Lecture Notes in Computer Science Vol. 4939, Springer-Verlag, pp. 37–46, 2008
69. S. Micali, C.P. Schnorr, “Efficient, Perfect Random Number Generators”, Advances in Cryptology – Crypto 1988, Lecture Notes in Computer Science Vol. 403, pp. 173–198, Springer-Verlag, 1988
70. G.L. Miller, “Riemann’s hypothesis and test for primality”, STOC, pp. 234–239, 1975
71. P. Nguyen, J. Stern, “The Two Faces of Lattices in Cryptology”, Cryptography and Lattices – Proceedings of CALC, Lecture Notes in Computer Science Vol. 2146, 2001
72. P. Nguyen, D. Stehlé, “Floating-Point LLL Revisited”, Advances in Cryptology – Eurocrypt 2005, Lecture Notes in Computer Science Vol. 3494, pp. 215–233, Springer-Verlag, 2005
73. C. Pomerance, “The Quadratic Sieve Factoring Algorithm”, Advances in Cryptology – Eurocrypt 84, Lecture Notes in Computer Science, pp. 169–182, 1985
74. J.-J. Quisquater, C. Couvreur, “Fast decipherment algorithm for RSA public-key cryptosystem”, Electronic Letters 18 (21), pp. 905–907, 1982
75. R. Rivest, A. Shamir, “Efficient factoring based on partial information”, Advances in Cryptology (Eurocrypt ’85), Lecture Notes in Computer Science Volume 219, pp. 31–34, Springer-Verlag, 1986
76. R. Rivest, A. Shamir and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, Communications of the ACM, Vol. 21(2), pp.120–126, 1978
77. B. Santoso, N. Kunihiro, N. Kanayama, K. Ohta, “Factorization of Square-Free Integers with High Bits Known”, Progress in Cryptology, VIETCRYPT 2006, Lecture Notes in Computer Science Vol. 4341, pp. 115–130, 2006
78. C.P. Schnorr, “Factoring Integers and Computing Discrete Logarithms via Diophantine Approximations”, Advances in Cryptology – Eurocrypt 1991, Lecture Notes in Computer Science, pp. 281–293, Springer-Verlag, 1991
79. P. W. Shor, “Algorithms for quantum computation: Discrete log and factoring”, In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pages 124–134, 1994
80. V. Shoup, “OAEP Reconsidered”, Advances in Cryptology – Crypto 2001, Lecture Notes in Computer Science Vol. 2139, Springer-Verlag, pp. 239–259, 2001
81. D. Stehlé, “Floating-Point LLL: Theoretical and Practical Aspects”, LLL+25 Conference in honour of the 25th birthday of the LLL algorithm, 2007
82. R. Steinfeld, J. Pieprzyk, H. Wang, “On the Provable Security of an Efficient RSA-Based Pseudorandom Generator”, Advances in Cryptology – Asiacrypt 2006, Lecture Notes in Computer Science Vol. 4284, pp. 194–209, Springer-Verlag, 2006
83. R. Steinfeld, Y. Zheng, “On the Security of RSA with Primes Sharing Least-Significant Bits”, Applicable Algebra in Engineering, Communication and Computing Vol. 15(3-4), pp. 179–200, 2004
84. H.-M. Sun, M.-E. Wu, “An Approach Towards Rebalanced RSA-CRT with Short Public Exponent”, Cryptology ePrint Archive: Report 2005/053, online available at `http://eprint.iacr.org/2005/053`

85. H.-M. Sun, M. J. Hinek, and M.-E. Wu, "An Approach Towards Rebalanced RSA-CRT with Short Public Exponent", revised version of [84], online available at http://www.cacr.math.uwaterloo.ca/techreports/2005/cacr2005-35.pdf
86. S. A. Vanstone, R. J. Zuccherato, "Short RSA Keys and Their Generation", Journal of Cryptology 8(2), pp. 101–114, 1995
87. B. de Weger, "Cryptanalysis of RSA with small prime difference", Applicable Algebra in Engineering, Communication and Computing ,Vol. 13(1), Springer-Verlag, pp. 17–28, 2002
88. M. Wiener, "Cryptanalysis of short RSA secret exponents", IEEE Transactions on Information Theory, Vol. 36, pp. 553–558, 1990