Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt '99

Glenn Durfee*
Stanford University
gdurf@cs.stanford.edu

Phong Q. Nguyen
École Normale Supérieure
pnguyen@ens.fr

Abstract

At Asiacrypt '99, Sun, Yang and Laih proposed three RSA variants with short secret exponent that resisted all known attacks, including the recent Boneh-Durfee attack from Eurocrypt '99 that improved Wiener's attack on RSA with short secret exponent. The resistance comes from the use of unbalanced primes p and q. In this paper, we extend the Boneh-Durfee attack to break two out of the three proposed variants. While the Boneh-Durfee attack was based on Coppersmith's lattice-based technique for finding small roots to bivariate modular polynomial equations, our attack is based on its generalization to trivariate modular polynomial equations. The attack is heuristic but works well in practice, as the Boneh-Durfee attack. In particular, we were able to break in a few minutes the numerical examples proposed by Sun, Yang and Laih. The results illustrate once again the fact that one should be very cautious when using short secret exponent with RSA.

1 Introduction

The RSA [13] cryptosystem is the most widely used public-key cryptosystem. However, RSA is computationally expensive, as it requires exponentiations modulo N, where N is a large integer (at least 1024 bits due to recent progress in integer factorization [4]) product of two primes p and q. Consequently, speeding up RSA has been a stimulating area of research since the invention of RSA. Perhaps the simplest method to speed up RSA consists of shortening the exponents of the modular exponentiations. If e is the RSA public exponent and d is the RSA secret exponent, one can either choose a small e or a small d. The choice of a small d is especially interesting when the device performing secret operations (signature generation or decryption) has limited computed power, such as smartcards. Unfortunately, Wiener [20] showed over 10 years ago that if $d \leq N^{0.25}$, then one could (easily) recover d (and hence, the secret primes p and q) in polynomial time from e and N using the continued fractions algorithm. Verheul and van Tilborg [19] slightly improved the bound in 1997, by showing that Wiener's attack could be applied to larger d, provided an exhaustive search on about $2\log_2(d/N^{0.25})$ bits. At Eurocrypt '99, Boneh and Durfee [3] presented the first substantial improvement over Wiener's bound. Their attack can (heuristically) recover p and q in polynomial time if $d \leq N^{0.292}$. The attack is heuristic because it is based on the seminal lattice-based work by Coppersmith [5] on finding small roots to low-degree modular polynomial equations, in the bivariate case.[1] However, it should be emphasized that the attack works very well in practice.

*Supported by a NSF Graduate Research Fellowship.

[1] The bivariate case is only heuristic for now, as opposed to the (simpler) univariate case, for which the method can be proved rigorously. For more information, see [5, 2, 12].

At Asiacrypt '99, Sun, Yang and Laih [18] noticed that all those attacks on RSA with short secret exponent required some (natural) assumptions on the public modulus N. For instance, the Wiener's bound $N^{0.25}$ only holds if $p + q = O(\sqrt{N})$, and e is not too large. Similar restrictions apply to the extension to Wiener's attack by Verheul-van Tilborg [19], and to the Boneh-Durfee attack [3]. This led Sun, Yang and Laih to propose in [18] simple variants of RSA using a short secret exponent that, *a priori*, foiled all such attacks due to the previous restrictions. More precisely, they proposed three RSA schemes, in which only the (usual) RSA key generation is modified. In the first scheme, one chooses p and q of greatly different size, and a small exponent d in such a way that the previous attacks cannot apply. In particular, d can even be smaller than $N^{0.25}$ if p and q are unbalanced enough. The second scheme consists of a tricky construction that selects slightly unbalanced p and q in such a way that both e and d are small, roughly around $\sqrt{N}$. The third scheme is a mix of the first two schemes, which allows a trade-off between the sizes of e and d. Sakai, Morii and Kasahara [14] earlier proposed a different key generation scheme which achieves similar results to the third scheme, but that scheme can easily been shown insecure (see [18]).

In this paper, we show that the first and third schemes of [18] are insecure, by extending the Boneh-Durfee attack. Our attack can also break the second scheme, but only if the parameters are carelessly chosen. Boneh and Durfee reduced the problem of recovering the factors p and q to finding small roots of a particular bivariate modular polynomial equation derived from the basic equation $ed \equiv 1 \pmod{\phi(N)}$. Next, they applied an optimized version (for that particular equation) of Coppersmith's generic technique [5] for such problems. However, when p and q are unbalanced, the particular equation used by Boneh and Durfee is not enough, because it has no longer any "small" root. Our attack extends the Boneh-Durfee method by taking into account the equation $N = pq$. We work with a system of two modular equations with three unknowns; interestingly, when p and q are imbalanced, this approach leads to an attack on systems with d even larger than the $N^{0.292}$ bound of Boneh and Durfee. The attack is extremely efficient in practice: for typical instances of two of the schemes of [18], this approach breaks the schemes within several minutes. Also, our "triviariate" version of Coppersmith's technique we use may be of independent interest.

The remainder of this paper is organized as follows. In Section 2, we briefly review former attacks on RSA with short secret exponents, recalling necessary background on lattice theory and Coppersmith's method to find small roots of low-degree modular polynomial equations. This is useful to explain our attacks. In Section 3, we describe the RSA schemes with short secret exponent of [18]. In Section 4, we present the new attack using the trivariate approach. We discuss an implementation of the attack and its running time on typical instances of the RSA variants in Section 5.

2 Former Attacks on RSA with Short Secret Exponent

All known attacks on RSA with short secret exponent focus on the equation $ed \equiv 1 \bmod \phi(N)$ (where $\phi(N) = N - (p + q) + 1$) rewritten as:

$$ed = 1 + k\left(\frac{N+1}{2} - s\right) \tag{1}$$

where k is an unknown integer and $s = (p + q)/2$. The primes p and q can be recovered from either d or s. Note that k and d are coprime.

2.1 The Wiener Attack

Wiener's attack [20] is based on the continued fractions algorithm. Recall that if two (unknown) coprime integers A and B satisfy $|x - \frac{B}{A}| < \frac{1}{2A^2}$ where x is a known rational, then $\frac{B}{A}$ can be obtained in polynomial time as a convergent of the continued fraction expansion of x. Here, (1) implies that

$$\left|\frac{2e}{N} - \frac{k}{d}\right| = \frac{|2 + k(1-2s)|}{Nd}.$$

Therefore, if $\frac{k(2s-1)-2}{N} < \frac{1}{2d}$, d can be recovered in polynomial time from e and N, as k/d is a convergent of the continued fraction expansion of $2e/N$. That condition can roughly be simplified to $ksd = O(N)$, and is therefore satisfied if k, s and d are all sufficiently small. In the usual RSA key generation, $s = O(\sqrt{N})$ and $k = O(d)$, which leads to the approximate condition $d = O(N^{0.25})$. But the condition gets worse if p and q are unbalanced, making s much larger than $\sqrt{N}$. For instance, if $p = O(N^{0.25})$, the condition becomes $d = O(N^{0.125})$.

The extension of Wiener's attack by Verheul and van Tilborg [19] applies to $d > N^{0.25}$ provided exhaustive search on $O(\log_2(d/N^{0.25}))$ bits if p and q are balanced. Naturally, the attack requires much more exhaustive search if p and q are unbalanced.

2.2 The Boneh-Durfee Attack

2.2.1 The Small Inverse Problem.

The Boneh-Durfee attack [3] looks at the equation (1) modulo e:

$$-k\left(\frac{N+1}{2} - s\right) \equiv 1 \pmod{e}. \qquad (2)$$

Assume that the usual RSA key generation is used, so that $|s| < \sqrt{e}$ and $|k| < d$ (ignoring small constants). The problem of finding such a small root (s, k) of that bivariate modular equation was called the *small inverse problem* in [3], since one is looking for a number $(N+1)/2 - s$ close to $(N+1)/2$ such that its inverse $-k$ modulo e is rather small. Note that heuristically, the small inverse problem is expected to have a unique solution whenever $|k| < d \leq N^{0.5}$. This led Boneh and Durfee to conjecture that RSA with $d \leq N^{0.5}$ is insecure.

Coppersmith [5] devised a general lattice-based technique to find sufficiently small roots of low-degree modular polynomial equations, which we will review in the next subsections, as it is the core of our attacks. By optimizing that technique to the specific polynomial of (2), Boneh and Durfee showed that one could solve the small inverse problem (and hence, break RSA) when $d \leq N^{0.292}$. This bound corresponds to the usual case of balanced p and q. It gets worse as p and q are unbalanced (see [3, 18]), because s becomes larger.

2.2.2 Lattice theory.

Coppersmith's technique, like many public-key cryptanalyses, is based on lattice basis reduction. We only review what is strictly necessary for this paper. Additional information on lattice theory can be found in numerous textbooks, such as [6, 17]. For the important topic of lattice-based cryptanalysis, we refer to the recent survey [12].

We will call *lattice* any subgroup of some $(\mathbb{Z}^n, +)$, which corresponds to the case of integer lattices in the literature. Consequently, for any integer vectors $\mathbf{b}_1, \ldots, \mathbf{b}_r$, the set $L(\mathbf{b}_1, \ldots, \mathbf{b}_r) = \{\sum_{i=1}^r n_i \mathbf{b}_i \mid n_i \in \mathbb{Z}\}$ of all integer linear combinations of the $\mathbf{b}_i$'s is a lattice, called the lattice *spanned* by the $\mathbf{b}_i$'s. In fact, all lattices are of that form. When $L = L(\mathbf{b}_1, \ldots, \mathbf{b}_r)$ and the $\mathbf{b}_i$'s are further linearly independent (over $\mathbb{Z}$), then $(\mathbf{b}_1, \ldots, \mathbf{b}_r)$ is called a *basis* of L. Any lattice L has infinitely many bases. However, any two bases share some things in common, notably the number of elements r and the Gram determinant $\det_{1 \le i,j \le r} \langle \mathbf{b}_i, \mathbf{b}_j \rangle$ (where $\langle , \rangle$ denotes the Euclidean dot product). The parameter r is called the lattice *dimension* (or *rank*), while the square root of the Gram determinant is the lattice *volume* (or *determinant*), denoted by $\mathrm{vol}(L)$. The name volume comes from the fact that the volume matches the r-dimensional volume of the parallelepiped spanned by the $\mathbf{b}_i$'s. In the important case of full-dimensional lattices (r equal to n), the volume is also the absolute value of the determinant of any basis (hence the name determinant). In general, it is hard to give a "simple" expression for the lattice volume, and one contents oneself with the Hadamard's inequality to estimate the volume:

$$\mathrm{vol}(L) \le \prod_{i=1}^{r} \|\mathbf{b}_i\|.$$

Fortunately, sometimes, the lattice is full-dimensional and we know a specific basis which is triangular, making the volume easy to compute.

The volume is important because it enables one to estimate the size of short lattice vectors. A well-known result by Minkowski shows that in any r-dimensional lattice L, there exists a non-zero $\mathbf{x} \in L$ such that $\|\mathbf{x}\| \le \sqrt{r} \cdot \mathrm{vol}(L)^{1/r}$, where $\|.\|$ denotes the Euclidean norm. That bound is in some (natural) sense the best possible. The LLL algorithm [9] can be viewed, from a qualitative point of view, as a constructive version of Minkowski's result. Given any basis of some lattice L, the LLL algorithm outputs in polynomial time a so-called *LLL-reduced* basis of L. The exact definition of an LLL-reduced basis is beyond the scope of this paper, we only mention the properties that are of interest here:

Fact 2.1 *Any LLL-reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_r)$ of a lattice L in $\mathbb{Z}^n$ satisfies:*

$$\|\mathbf{b}_1\| \le 2^{r/2} \mathrm{vol}(L)^{1/r} \quad \textit{and} \quad \|\mathbf{b}_2\| \le 2^{(r-1)/2} \mathrm{vol}(L)^{1/(r-1)}.$$

2.2.3 Coppersmith's technique.

For a discussion and a general exposition of Coppersmith's technique [5], see the recent surveys [2, 12]. We describe the technique in the bivariate case, following a simplified approach due to Howgrave-Graham [7].

Let e be a large integer of possibly unknown factorization. Assume that one would like to find all small roots of $f(x, y) \equiv 0 \pmod{e}$, where $f(x, y)$ is an integer bivariate polynomial with at least one monomial of maximal total degree which is monic. If one could obtain two algebraically independent integral bivariate polynomial equations satisfied by all sufficiently small modular roots (x, y), then one could compute (by resultant) a univariate integral polynomial equation satisfied by x, and hence find efficiently all small (x, y). Coppersmith's method tries to obtain such equations from reasonably short vectors in a certain lattice. The lattice comes from the linearization of a set of equations of the form $x^u y^v f(x, y)^w \equiv 0 \pmod{e^w}$ for appropriate integral values of u, v and w. Such equations are satisfied by any solution of $f(x, y) \equiv 0 \pmod{e}$. Small solutions (x_0, y_0) give rise to unusually short solutions to the resulting linear system, hence short vectors in the lattice. To transform modular equations into integer equations, one uses the following elementary lemma, with the (natural) notation $\|h(x,y)\| = \sqrt{\sum_{i,j} a_{i,j}^2}$ for $h(x,y) = \sum_{i,j} a_{i,j} x^i y^j$:

Lemma 2.2 *Let $h(x,y) \in \mathbb{Z}[x,y]$ be a polynomial which is a sum of at most r monomials. Suppose that $h(x_0,y_0) \equiv 0 \bmod e^m$ for some positive integer m where $|x_0| < X$ and $|y_0| < Y$, and $\|h(xX,yY)\| < e^m/\sqrt{r}$. Then $h(x_0,y_0) = 0$ holds over the integers.*

Now the trick is to, given a parameter m, consider the polynomials

$$h_{u_1,u_2,v}(x,y) = e^{m-v} x^{u_1} y^{u_2} f(x,y)^v.$$

where u_1, u_2 and v are integers. Notice that any root (x_0,y_0) of $f(x,y)$ modulo e is a root modulo e^m of $h_{u_1,u_2,v}(x,y)$, and therefore, of any integer linear combination $h(x,y)$ of the $h_{u_1,u_2,v}(x,y)$'s. If such a combination $h(x,y)$ further satisfies $\|h(xX,yY)\| < e^m/\sqrt{r}$, where r is the number of monomials of h, then by Lemma 2.2, the integer equation $h(x,y) = 0$ is satisfied by all sufficiently small modular roots of h modulo e. Thus, it suffices to find two algebraically independent such equations $h_1(x,y)$ and $h_2(x,y)$.

The use of integer linear combination suggests that we represent the polynomials as vectors in a lattice, so that finding polynomials with small norm reduces to finding short vectors in a lattice. More precisely, let $\mathcal{S}$ be a set of indices (u_1,u_2,v), and choose a representation of the polynomials $h_{u_1,u_2,v}(x,y)$ with $(u_1,u_2,v) \in \mathcal{S}$ as n-dimensional integer vectors for some n. Let L be the lattice in $\mathbb{Z}^n$ spanned by the vectors corresponding to $h_{u_1,u_2,v}(xX,yY)$ with $(u_1,u_2,v) \in \mathcal{S}$. Apply the LLL algorithm on the lattice, and let $h_1(xX,yY)$ and $h_2(xX,yY)$ be the polynomials corresponding to the first two vectors of the reduced basis obtained. Denoting by r the dimension of L, one deduces from the LLL theoretical bounds that:

$$\|h_1(xX,yY)\| \leq 2^{r/2}\mathrm{vol}(L)^{1/r} \text{ and } \|h_2(xX,yY)\| \leq 2^{(r-1)/2}\mathrm{vol}(L)^{1/(r-1)}.$$

To apply Lemma 2.2, we want both of these upper bounds to be less than $e^m/\sqrt{n}$; since the factor 2^r is negligible with respect to e^m, this amounts to saying

$$\mathrm{vol}(L) \ll e^{mr}. \tag{3}$$

There are two problems. The first problem is that even if this condition is satisfied, so that Lemma 2.2 applies, we are not guaranteed that the integer equations $h_1(x,y) = 0$ and $h_2(x,y) = 0$ obtained are algebraically independent. In other words, h_2 will provide no additional information beyond h_1 if the two *linearly* independent short basis vectors do not also yield *algebraically* independent equations. It is still an open problem to state precisely when this can be guaranteed, although all experiments to date suggest this is an accurate heuristic assumption to make when inequality (3) holds. We note that a similar assumption is used in the work of Bleichenbacher [1] and Jutla [8].

The second problem is more down-to-earth: how can we make sure that $\mathrm{vol}(L)$ is small enough to satisfy inequality (3) ? Note that Hadamard's bound is unlikely to be useful. Indeed, in general, some of the coefficients of $f(x,y)$ are about the size of e, so that $\|h_{u_1,u_2,v}(xX,yY)\|$ is at least e^m. To address this problem, one must choose in a clever way the set of indices $\mathcal{S}$ to have a close estimate on $\mathrm{vol}(L)$. The simplest solution is to choose $\mathcal{S}$ so that L is full-dimensional (r equal to n) and the $h_{u_1,u_2,v}(xX,yY)$'s form a triangular matrix for some ordering on the polynomials and on the monomials (the vector coordinates). Since we want $\mathrm{vol}(L)$ to be small, each coefficient on the diagonal should be the smallest one of $h_{u_1,u_2,v}(xX,yY) = e^{m-v}(xX)^{u_1}(yY)^{u_2}f(xX,yY)^v$, which is likely to be the one corresponding to the monic monomial of maximal total degree of $f(x,y)$.

In the general case, $f(x,y)$ may have several monomials of maximal total degree, and the only simple choice of $\mathcal{S}$ is to cover all the monomials of total degree less than some parametrized bound. More precisely, if Δ is the total degree of $f(x,y)$, and $x^a y^{\Delta-a}$ is a monic monomial of $f(x,y)$, one defines $\mathcal{S}$ as the set of (u_1, u_2, v) such that $u_1 + u_2 + \Delta v \leq h\Delta$ and $u_1, u_2, v \geq 0$ with $u_1 < a$ or $u_2 < \Delta - a$. Then the volume of the corresponding lattice can be computed exactly, and it turns out that (3) is satisfied whenever $XY < e^{1/\Delta - \varepsilon}$ for and m is sufficiently large.

However, depending on the shape of $f(x,y)$ (represent each monomial $x^i y^j$ by the point (i,j)), other choices of $\mathcal{S}$ might lead to improved bounds. Boneh and Durfee applied such tricks to the polynomial (2). In [3], they discussed several choices of $\mathcal{S}$. Using certain sets $\mathcal{S}$ for which the lattice is full-dimensional and one knows a triangular lattice basis, they obtained a first bound $d \leq N^{0.284}$ for their attack. Next, they showed that using a slightly different $\mathcal{S}$ for which the lattice is no longer full-dimensional, one ends up with the improved bound $d \leq N^{0.292}$. The latter choice of $\mathcal{S}$ is much harder to analyze. For more details, see [3].

3 The Sun-Yang-Laih RSA Key Generation Schemes

3.1 Scheme (I)

The first scheme corresponds to a simple unbalanced RSA [15] in which the parameters are chosen to foil previously known attacks:

1. Select two random primes $p < q$ such that both p and $N = pq$ are sufficiently large to foil factorization algorithms such as ECM and NFS. The more unbalanced p and q are, the smaller d can be.

2. Randomly select the secret exponent d such that $\log_2 d + \log_2 p > \frac{1}{3}\log_2 N$ and $d > 2^\gamma \sqrt{p}$, where γ is the security parameter (larger than 64).

3. If the public exponent e defined by $ed \equiv 1 \pmod{\phi(N)}$ is not larger than $\phi(N)/2$, one restarts the previous step.

A choice of parameters suggested by the authors is: p is a 256-bit prime, q is a 768-bit prime, d is a 192-bit number. Note that 192 is far below Wiener's bound (256 bits) and Boneh-Durfee's bound (299 bits).

3.2 Scheme (II)

The second scheme selects one of the primes in such a way that one can select e and d to be small at the same time:

1. Fix the bit-length of N.

2. Select a random prime p of $\frac{1}{2}\log_2 N - 112$ bits, and a random k of 112 bits.

3. Select a random d of $\frac{1}{2}\log_2 N + 56$ bits coprime with $k(p-1)$.

4. Compute the two Bézout integers u and v such that $du - k(p-1)v = 1$, $0 < u < k(p-1)$ and $0 < v < d$.

5. Return to Step 3 if $v+1$ is not coprime with d.

6. Select a random h of 56 bits until $q = v + hd + 1$ is prime.

The RSA parameters are p, q, $e = u + hk(p-1)$, d and $N = pq$. Notice that e and d satisfy the equation $ed = 1 + k\phi(N)$. They both have approximate bit-length $\frac{1}{2}\log_2 N + 56$. The primes p and q have approximate bit-length $\frac{1}{2}\log_2 N - 112$ and $\frac{1}{2}\log_2 N + 112$ respectively.

A possible choice of parameters for Scheme (II) might be: p a 400-bit prime, q a 624-bit prime, and e and d are each 568-bit integers.

3.3 Scheme (III)

The third scheme is a mix of the first two schemes, allowing a trade-off between e and d such that $\log_2 e + \log_2 d \approx \log_2 N + \ell_k$ where ℓ_k is a predetermined constant. More precisely, the scheme is a parametrized version of scheme II: p, k, d and h have respective bit-length ℓ_p (less than $\frac{1}{2}\log_2 N$), ℓ_k, ℓ_d, and $\log_2 N - \ell_p - \ell_d$. To resist various attacks, the following is required:

1. $\ell_k \gg \ell_p - \ell_d + 1$.

2. $4\alpha(2\beta + \alpha - 1) \gg 3(1 - \beta - \alpha)^2$, where $\alpha = \frac{\log_2 N - \ell_p}{\log_2 N + \ell_k - \ell_d}$ and $\beta = \frac{\ell_k}{\log_2 N + \ell_k - \ell_d}$.

3. k must withstand an exhaustive search and $\ell_k + \ell_p > \frac{1}{3}\log_2 N$.

   A choice of parameters suggested by the authors is: p is a 256-bit prime, q is a 768-bit prime, e is an 880-bit number, and d is a 256-bit number.

4 The Attack Algorithm

In this section we demonstrate how to launch an attack on Schemes (I) and (III). The approach used here closely follows that taken by Boneh and Durfee [3], but differs in several crucial ways to allow it to work when the factors p and q of the public modulus N are unbalanced. Interestingly, our attack gets better (works for larger and larger d) the more unbalanced the factors of the modulus become.

Recall the RSA equation

$$ed = 1 + k\left(\frac{N+1}{2} - \frac{p+q}{2}\right).$$

We note that the Boneh-Durfee approach treats this as an equation modulo e with two "small" unknowns, k and $s = (p+q)/2$. This approach no longer works if p and q are unbalanced, since a good bound on s can no longer be established. For this reason, the authors of the schemes from Section 3 hoped that these schemes would resist the lattice-based cryptanalysis outlined in Section 2.2.3. However, we will see that a more careful analysis of the RSA equation, namely one that does not treat $p+q$ as a single unknown quantity but instead leaves p and q separately as unknowns, leads to a successful attack against two of these schemes.

Writing $A = N + 1$, the RSA equation implies

$$2 + k(A - p - q) \equiv 0 \pmod{e}.$$

The critical improvement of our attack is to view this as a modular equation with *three* unknowns, k, p, q, with the special property that the product pq of two of them is the known quantity N. We may view this problem as follows: given a polynomial $f(x, y, z) = x(A + y + z) - 2$, find (x_0, y_0, z_0) satisfying:

$$f(x_0, y_0, z_0) \equiv 0 \pmod{e},$$

where

$$|x_0| < X, \quad |y_0| < Y, \quad |z_0| < Z, \quad \text{and} \quad y_0 z_0 = N.$$

Note that the bounds $X \approx ed/N$, $Y \approx p$, and $Z \approx q$ can be estimated to within a power of 2 based on the security parameters chosen for the scheme.

Following Coppersmith's method, our approach is to pick r equations of the form $e^{m-v} x^{u_1} y^{u_2} z^{u_3} \cdot f^v(x, y, z)$ and to search for low-norm integer linear combinations of these polynomials. The basic idea is to start with a handful of equations of the form $y^{a+j} f^m(x, y, z)$ for $j = 0, \ldots, t$ for some integers a and t with $t \geq 0$. Knowing $N = pq$ allows us to replace all occurrences of the monomial yz with the constant N, reducing the number of variables in each of these equations to approximately m^2 instead of the expected $\frac{1}{3}m^3$. We will refer to these as the *primary polynomials.*

Since there are only $t + 1$ of these equations, this will result in a lattice that is less than full rank; we therefore include some additional equations to bring the lattice to full rank in order to compute its determinant. We refer to these as the *helper polynomials.* We have a great deal of choice in picking the helper polynomials; naturally, some choices are better than others, and it is generally a tedious but straightforward optimization problem to choose the primary and helper polynomials that are optimal. The equations we work with are the following. Fix an integer m, and let a and $t > 0$ be integers which we will optimize later. We define

- $g_{k,i,b}(x, y, z) := e^{m-k} x^i y^a z^b f^k(x, y, z)$, for $k = 0..(m - 1)$, $i = 1..(m - k)$, and $b = 0, 1$; and,
- $h_{k,j}(x, y, z) := e^{m-k} y^{a+j} f^k(x, y, z)$, for $k = 0..m$ and $j = 0..t$.

The primary polynomials are $h_{m,j}(x, y, z)$ for $j = 0, \ldots, t$, and the rest are the helper polynomials. Following Coppersmith's technique, we form a lattice L by representing $g_{k,i,b}(xX, yY, zZ)$ and $h_{k,j}(xX, yY, zZ)$ by their coefficients vectors, and use LLL to find low-norm integer linear combinations $h_1(xX, yY, zZ)$ and $h_2(xX, yY, zZ)$. The polynomials $h_1(x, y, z)$ and $h_2(x, y, z)$ have (k, p, q) as a root over the integers; to remove z as an unknown, we use the equality $z = N/y$, obtaining $H_1(x, y)$ and $H_2(x, y)$ which have (k, p) as a solution. Taking the resultant $\mathrm{Res}_x(H_1(x, y), H_2(x, y))$ yields a polynomial $H(y)$ which has p as a root. Using standard root-finding techniques allows us to recover the factor p of N efficiently, completing the attack.

The running time of this algorithm is dominated by the time to run LLL on the lattice L, which has dimension $(m + 1)(m + t + 1)$. So it would be ideal to keep the parameters m and t as low as possible, limiting to a reasonable number the polynomials used to construct L. Surprisingly, the attack is successful even if only a handful of polynomials are used. The example given by the original authors for schemes (I) succumbs easily to this attack with $m = 3$ and $t = 1$; with these parameters, our attack generates 20 polynomials. Scheme (III) can be cryptanalyzed with parameters $m = 2$ and $t = 2$, yielding 15 polynomials. This gives lattices of dimension 20 (see Figure 1) and 15, respectively, which can be reduced *via* the LLL algorithm within a matter of seconds on a desktop computer. We discuss our implementation and the results of our experiments more in Section 5.

4.1 Analysis of the Attack

In order to be sure that LLL returns vectors that are "short enough" to use Lemma 2.2, we must derive sufficiently small bounds on the determinant of the lattice L formed from the polynomials $g_{k,i,b}(xX, yY, zZ)$ and $h_{k,j}(xX, yY, zZ)$. Fortunately, this choice of polynomials makes the computation of the determinant of L fairly straightforward, if somewhat tedious. We provide the details in the appendix.

Representing the Lattice as a Triangular Matrix. In order to compute the volume of the lattice L, we would like to list the polynomials $g_{k,i,b}(xX, yY, zZ)$ and $h_{k,j}(xX, yY, zZ)$ in a way that yields a triangular matrix. There is an ordering on these polynomials that leads to such a representation: we first list the $g_{k,i,b}(xX, yY, zZ)$ indexed outermost by $k = 0, \ldots, m-1$, then $i = 0, \ldots, k$, then innermost by $b = 0, 1$. We then list $h_{k,j}(xX, yY, zZ)$ indexed outermost by $k = 0, \ldots, m$ then $j = 0, \ldots, t$. (See Figure 1 for the case of $m = 2$, $t = 1$, $a = 1$.) Each new polynomial introduces exactly one new monomial $x^{u_1}y^{u_2}$ or $x^{u_1}z^{u_3}$. Note that no monomial involving the product yz appears, since yz can be eliminated[2] using the identity $N = yz$.

	xy	x	x^2y	x^2	x^3y	x^3	x^2y^2	x^2z	x^3y^2	x^3z	x^3y^3	x^3z^2	y	y^2	xy^2	xy^3	x^2y^3	x^2y^4	x^3y^4	x^3y^5
e^3xy	e^3XY																			
e^3xyz		e^3X																		
e^3x^2y			e^3X^2Y																	
e^3x^2yz				e^3X^2																
e^3x^3y					e^3X^3Y															
e^3x^3yz						e^3X^3														
e^2xyf	–		–	–			$e^2X^2Y^2$													
e^2xyzf		–	–	–				e^2X^2Z												
e^2x^2yf			–		–	–			$e^2X^3Y^2$											
e^2x^2yzf				–	–	–				e^2X^3Z										
$exyf^2$	–		–	–	–	–	–		–	–	eX^3Y^3									
$exyzf^2$		–	–	–	–	–		–	–	–		eX^3Z^2								
e^3y													e^3Y							
e^3y^2														e^3Y^2						
e^2yf	–	–											–		e^2XY^2					
e^2y^2f		–												–	–	e^2XY^3				
eyf^2	–	–	–	–			–	–					–		–		eX^2Y^3			
ey^2f^2	–		–	–			–							–	–	–	–	eX^2Y^4		
yf^3	–	–	–	–	–	–	–	–	–	–	–	–	–		–		–		X^3Y^4	
y^2f^3	–		–	–	–	–	–		–	–	–			–	–	–	–	–	–	X^3Y^5

Figure 1: *Example of the lattice formed by the vectors $g_{k,i,b}(xX, yY, zZ)$ and $h_{k,j}(xX, yY, zZ)$ when $m = 2$, $t = 1$, and $a = 1$. The matrix is lower triangular. Entries marked with "–" indicate off-diagonal quantities whose values do not affect the determinant calculation. The polynomials used are listed on the left, and the monomials they introduce are listed across the top. The double line break occurs between the $g_{k,i,b}$ and the $h_{k,j}$, while the single line breaks occur between increments of k. The last single line break separates the helper polynomials (top) from the two primary polynomials (bottom).*

[2] Caution must be taken to ensure the polynomials remain monic in the terms $x^{u_1}y^{u_2}$ and $x^{u_1}z^{u_3}$ of highest degree; if the substitution $yz \mapsto N$ causes a coefficient of such a term to be different from 1, then we multiply the polynomial by $N^{-1} \bmod e^m$ (and reduce mod e^m as appropriate) before continuing.

The determinant of this matrix is simply the product of the entries on the diagonal, which for $m = 3$, $t = 1$, $a = 1$ is

$$\mathrm{vol}(L) = \det(M) = e^{40}X^{40}Y^{34}Z^4. \tag{4}$$

We expect the LLL algorithm to return vectors short enough to use Lemma 2.2 when

$$\mathrm{vol}(L) = e^{40}X^{40}Y^{34}Z^4 < e^{mr} = e^{60}.$$

The example given by the original authors for Scheme (I) is to use p of 256 bits, q of 768 bits, d of 192 bits, and e of 1024 bits. This gives bounds

$$X \approx ed/N \approx e^{3/16}, \;\; Y \approx e^{1/4}, \text{ and } Z \approx e^{3/4};$$

we may then confirm

$$\det(M) = e^{40}X^{40}Y^{34}Z^4 \approx e^{59} < e^{60} = e^{mr},$$

so Lemma 2.2 applies.[3] Therefore, when we run the LLL algorithm on this lattice, we will get two short vectors corresponding to polynomials $h_1(x, y, z)$, $h_2(x, y, z)$; by the bound on the determinant, we know that these polynomials will have norm that is low enough to use Lemma 2.2. Therefore these polynomials will have (k, p, q) as a solution over the integers. To turn these into bivariate equations, we use the equality $z = N/y$ to get $H_1(x, y)$ and $H_2(x, y)$ which have (k, p) as a solution over the integers. We then take the resultant $Res_x(H_1(x, y), H_2(x, y))$ to obtain a univariate polynomial $H(y)$ that has p as a root.

More generally, if we pick optimal values for t and a take m sufficiently large, our attack will be successful for even larger bounds on d. The highest possible bound on d for which our attack can work depends on the parameters chosen for the scheme. Suppose the parameter $d \approx N^\delta$ is used. The table below summarizes the largest possible δ for which our attack can succeed. We point out the choices of parameters that give rise to the schemes of Section 3.

		$\log_N(e)$						
		1.0	0.9	0.86	0.8	0.7	0.6	0.55
$\log_N(p)$	0.5	0.284	0.323	0.339	0.363	0.406	0.451	0.475
	0.4	0.296	0.334	0.350	0.374	0.415	0.460	0.483_{II}
	0.3	0.334	0.369	0.384	0.406	0.446	0.487	0.510
	0.25	0.364_{I}	0.398	0.412_{III}	0.433	0.471	0.511	0.532
	0.2	0.406	0.437	0.450	0.470	0.505	0.542	0.562
	0.1	0.539	0.563	0.573	0.588	0.615	0.644	0.659

Figure 2: *Largest δ (where $d < N^\delta$) for which our attack can succeed, as a function of the system parameters.*

[3] The reader may have noticed that we have suppressed the error term associated with the execution of the LLL algorithm. Interestingly, even if the LLL "fudge factor" is taken into account, this bound is still good enough. We require

$$\mathrm{vol}(L) < 2^{r^2/2}e^{59} = 2^{200}e^{59} < e^{59+\frac{1}{5}} < e^{mr}/(\sqrt{r})^r \approx e^{60-\frac{1}{20}}.$$

Slightly larger parameters m and t are required to rigorously obtain the bound for norm of the second basis vector, although in practice the LLL algorithm works well enough so that the parameters chosen here are sufficient.

For example, with the example for Scheme (I), where $e \approx N$ and $p \approx N^{0.25}$, our attack will be successful not only for the $\delta = 0.188$ suggested, but all the way up to $\delta < 0.364$ (assuming a large enough m is used.) Similarly, our attack works in Scheme (III) up to $d < N^{0.412}$. Notice that our attack comes close to, but cannot quite reach, the $d < N^{0.55}$ required to break Scheme (II).

4.2 Comparison with the Bivariate Approach

Alternatively, one can consider the system of two modular equations with three unknowns as a single bivariate equation by incorporating the equation $N = pq$ into the main trivariate equation. This was independently noticed by Willi Meier [11], who also addressed the problem of breaking Schemes (I) and (III), using a bivariate approach rather than our trivariate approach. One then obtains an equation of the form $f(x, y) = x^2y + Axy + Bx + Cy$ modulo e, where the unknowns are k and the smallest prime among p and q.

However, it turns out that the application of Coppersmith's technique to this particular bivariate equation yields worse bounds than with the trivariate approach previously described. For example, the bivariate approach allows one to break scheme (I) as long as $d < N^{0.135}$ (and perhaps slightly higher, if sublattices are considered as in [3]), but fails for larger d. One can view the bivariate approach a special case of our trivariate approach, in which one degree of freedom for optimization has been removed. One then sees that the bivariate approach constrains the choice of primary and helper polynomials in a suboptimal way, resulting in worse bounds on d.

5 Implementation

We implemented this attack using Victor Shoup's Number Theory Library [16] and the Maple Analytical Computation System [10]. The attack runs very efficiently, and in all instances of Schemes (I) and (III) we tested, it produced algebraically independent polynomials $H_1(x, y)$ and $H_2(x, y)$. These yielded a resultant $H(y) = (y-p)H_0(y)$, where $H_0(y)$ is irreducible, exposing the factor p of N in every instance. This strongly suggests that this "heuristic" assumption needed to complete the multivariate modular version of Coppersmith's technique is extremely reliable, and we conjecture that it always holds for suitably bounded lattices of this form. The running times of our attacks are given below.

Scheme	size of n	size of p	size of e	size of d	m	t	a	lattice rank	running time
I	1024	256	1024	192	3	1	1	20	40 seconds
III	1024	256	880	256	2	2	0	15	9 seconds

These tests were run on a 500MHz Pentium III running Solaris.

6 Conclusions and Open Problems

We showed that unbalanced RSA [15] actually improves the attacks on short secret exponent by allowing larger exponent. This enabled us to break most of the RSA schemes [18] with short secret exponent from Asiacrypt '99. The attack extends the Boneh-Durfee attack [3] by using a "trivariate" version of Coppersmith's lattice-based technique for finding small roots of low-degree modular polynomial equations. Unfortunately, despite experimental evidence, the attack is for now only heuristic, as the Boneh-Durfee attack. It is becoming increasingly important to find sufficient conditions for which Coppersmith's technique on multivariate modular polynomials can be proved.

Our results illustrate once again the fact that one should be very cautious when using RSA with short secret exponent. To date, the best method to enjoy the computational advantage of short secret exponent is the following countermeasure proposed by Wiener [20]. When $N = pq$, the idea is to use a private exponent d such that both $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$ are small. Such a d speeds up RSA signature generation since RSA signatures are often generated modulo p and q separately and then combined using the Chinese Remainder Theorem. Classical attacks do not work since d is likely to be close to $\phi(N)$. It is an open problem whether there is an efficient attack on such secret exponents. The best known attack runs in time $\min(\sqrt{d_p}, \sqrt{d_q})$.

Acknowledgements

Part of this work was done while the second author was visiting Stanford University, whose hospitality is gratefully acknowledged. We would like to thank Dan Boneh for many helpful suggestions, and Hyun-Soo Hong for helpful comments on the proceedings version.

References

[1] D. Bleichenbacher. On the security of the KMOV public key cryptosystem. In *Proc. of Crypto '97*, volume 1294 of *LNCS*, pages 235–248. IACR, Springer-Verlag, 1997.

[2] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2):203–213, 1999.

[3] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. In *Proc. of Eurocrypt '99*, volume 1592 of *LNCS*, pages 1–11. IACR, Springer-Verlag, 1999.

[4] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann. Factorization of 512-bit RSA key using the number field sieve. In *Proc. of Eurocrypt'2000*, volume 1807 of LNCS. IACR, Springer-Verlag, 2000. Factorization announced in August, 1999.

[5] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. of Cryptology*, 10(4):233–260, 1997. Final version of two articles from Eurocrypt '96.

[6] M. Gruber and C. G. Lekkerkerker. *Geometry of Numbers.* North-Holland, 1987.

[7] N. Howgrave-Graham. Finding small roots of univariate modular equations revisited. In *Cryptography and Coding*, volume 1355 of *LNCS*, pages 131–142. Springer-Verlag, 1997.

[8] C. S. Jutla. On finding small solutions of modular multivariate polynomial equations. In *Proc. of Eurocrypt '98*, volume 1403 of *LNCS*, pages 158–170. IACR, Springer-Verlag, 1998.

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

[10] Waterloo Maple. The Maple computational algebra system for algebra, number theory and geometry. Information available at `http://www.maplesoft.com/products/Maple6/maple6info.html`.

[11] W. Meier. Private communication. June, 2000.

[12] P. Q. Nguyen and J. Stern. Lattice reduction in cryptology: An update. In *Algorithmic Number Theory – Proc. of ANTS-IV*, volume 1838 of LNCS. Springer-Verlag, 2000.

[13] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[14] R. Sakai, M. Morii, and M. Kasahara. New key generation algorithm for RSA cryptosystem. *IEICE Trans. Fundamentals*, E77-A(1):89–97, 1994.

[15] A. Shamir. RSA for paranoids. *RSA Laboratories CryptoBytes*, 1(3):1–4, 1995.

[16] V. Shoup. Number Theory C++ Library (NTL) version 3.6. Available at `http://www.shoup.net/ntl/`.

[17] C. L. Siegel. *Lectures on the Geometry of Numbers*. Springer-Verlag, 1989.

[18] H.-M. Sun, W-C. Yang, and C.-S. Laih. On the design of RSA with short secret exponent. In *Proc. of Asiacrypt '99*, volume 1716 of *LNCS*, pages 150–164. IACR, Springer-Verlag, 1999.

[19] E. Verheul and H. van Tilborg. Cryptanalysis of less short RSA secret exponents. *Applicable Algebra in Engineering, Communication and Computing*, 8:425–435, 1997.

[20] M. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Trans. Inform. Theory*, 36(3):553–558, 1990.

A General calculation of the determinant

The general formula for the determinant of the lattice we build in Section 4 is

$$\mathrm{vol}(L) = \det(M) = e^{C_e} X^{C_x} Y^{C_y} Z^{C_z},$$

where

$$\begin{aligned}
C_e &= C_x = \frac{1}{6} m(m+1)(4m+3t+5), \\
C_y &= \begin{cases} \frac{1}{6}(m^3 + 3(a+t+1)m^2 + (3t^2+6at+3a^2+6a+6t+2)m \\ \qquad +(3t^2+6at+3a^2+4a+3t-a^3)) & \text{if } a \geq 0, \\ \frac{1}{6}(m^3 + 3(a+t+1)m^2 + (3t^2+6at+3a^2+6a+6t+2)m \\ \qquad +(3t^2+6at+3a^2+3a+3t)) & \text{if } a < 0, \end{cases} \\
C_z &= \begin{cases} \frac{1}{6}(m^3 - 3(a-1)m^2 + (3a^2-6a+2)m + (3a^2-2a-a^3)) & \text{if } a \geq 0, \\ \frac{1}{6}(m^3 - 3(a-1)m^2 + (3a^2-6a+2)m + (3a^2-3a)) & \text{if } a < 0. \end{cases}
\end{aligned}$$

We need $\det(M) < e^{mr} = e^{m(m+1)(m+t+1)}$. In order to optimize the choice of t and a, we write $t = \tau m$ and $a = \alpha m$, and observe

$$\begin{aligned}
C_e &= C_x = \frac{1}{6}(3\tau + 4)m^3 + o(m^3), \\
C_y &= \begin{cases} \frac{1}{6}(3\tau^2 + 6\alpha\tau + 3\alpha^2 + 3\alpha + 3\tau + 1 - \alpha^3)m^3 + o(m^3) & \text{if } \alpha \geq 0, \\ \frac{1}{6}(3\tau^2 + 6\alpha\tau + 3\alpha^2 + 3\alpha + 3\tau + 1)m^3 + o(m^3) & \text{if } \alpha < 0, \end{cases} \\
C_z &= \begin{cases} \frac{1}{6}(3\alpha^2 - 3\alpha + 1 - \alpha^3)m^3 + o(m^3) & \text{if } \alpha \geq 0, \\ \frac{1}{6}(3\alpha^2 - 3\alpha + 1)m^3 + o(m^3) & \text{if } \alpha < 0. \end{cases}
\end{aligned}$$

Suppose we write $e = N^{\varepsilon}$, $d = N^{\delta}$, and $Y = N^{\beta}$, so $Z = N^{1-\beta}$. Then $X = N^{\varepsilon+\delta-1}$. So the requirement on $\det(M)$ now becomes

$$N^{\varepsilon C_e + (\varepsilon+\delta-1)C_x + \beta C_y + (1-\beta)C_z} < e^{m(m+1)(m+t+1)} = N^{\varepsilon(\tau+1)m^3 + o(m^3)}.$$

The above expression holds (for large enough m) when

$$\varepsilon C_e + (\varepsilon + \delta - 1)C_x + \beta C_y + (1-\beta)C_z - (\tau + 1) < 0. \tag{5}$$

The left-hand-side of this expression achieves its minimum at

$$\begin{aligned}
\tau_0 &= (2\alpha_0\beta - \beta - \delta + 1)/(2\beta), \\
\alpha_0 &= \begin{cases} 1 - \beta - (1 - \beta - \delta + \beta^2)^{(1/2)} & \text{if } \beta < \delta, \\ (\beta - \delta)/(2\beta - 2) & \text{if } \beta \geq \delta. \end{cases}
\end{aligned}$$

Using $\tau = \tau_0$ and $\alpha = \alpha_0$ will give us the minimum value on the left-hand-side of inequality 5, affording us the largest possible X to give an attack on the largest possible $d < N^{\delta}$. The entries in Figure 2 were generated by plugging in τ_0 and α_0 and solving for equality in Equation 5.

It is interesting to note that formulation of the root-finding problem for RSA as a trivariate equation is strictly more powerful than its formulation as the small inverse problem. This is because the small inverse problem is not expected to have a unique solution once $\delta > 0.5$, while our attack works in many cases with $\delta > 0.5$. We note that when $\varepsilon = 1$ and $\beta = 0.5$ – as in standard RSA – our attack gives identical results to simpler Boneh-Durfee attack ($d < N^{0.284}$). Their optimization of using lattices of less than full rank to achieve the $d < N^{0.292}$ bound should also work with our approach, but we have not analyzed how much of an improvement it will provide.