Twenty Years of Attacks on the RSA Cryptosystem

Dan Boneh
dabo@cs.stanford.edu

1 Introduction

The RSA cryptosystem, invented by Ron Rivest, Adi Shamir, and Len Adleman [21], was first publicized in the August 1977 issue of *Scientific American*. The cryptosystem is most commonly used for providing privacy and ensuring authenticity of digital data. These days RSA is deployed in many commercial systems. It is used by web servers and browsers to secure web traffic, it is used to ensure privacy and authenticity of Email, it is used to secure remote login sessions, and it is at the heart of electronic credit-card payment systems. In short, RSA is frequently used in applications where security of digital data is a concern.

Since its initial publication, the RSA system has been analyzed for vulnerability by many researchers. Although twenty years of research have led to a number of fascinating attacks, none of them is devastating. They mostly illustrate the dangers of improper use of RSA. Indeed, securely implementing RSA is a nontrivial task. Our goal is to survey some of these attacks and describe the underlying mathematical tools they use. Throughout the survey we follow standard naming conventions and use Alice and Bob to denote two generic parties wishing to communicate with each other. We use Marvin to denote a malicious attacker wishing to eavesdrop or tamper with the communication between Alice and Bob.

We begin by describing a simplified version of RSA encryption. Let $N = pq$ be the product of two large primes of the same size ($n/2$ bits each). A typical size for N is $n = 1024$ bits, i.e. 309 decimal digits. Each of the factors is 512 bits. Let e, d be two integers satisfying $ed = 1 \bmod \varphi(N)$ where $\varphi(N) = (p-1)(q-1)$ is the order of the multiplicative group $\mathbb{Z}_N^*$. We call N the *RSA modulus*, e the *encryption exponent*, and d the *decryption exponent*. The pair $\langle N, e\rangle$ is the *public key*. As its name suggests, it is public and is used to encrypt messages. The pair $\langle N, d\rangle$ is called the *secret key* or *private key* and is known only to the recipient of encrypted messages. The secret key enables decryption of ciphertexts.

A *message* is an integer $M \in \mathbb{Z}_N^*$. To encrypt M, one computes $C = M^e \bmod N$. To decrypt the ciphertext, the legitimate receiver computes $C^d \bmod N$. Indeed,

$$C^d = M^{ed} = M \pmod{N},$$

where the last equality follows by Euler's theorem[1]. One defines the RSA function as $x \longmapsto x^e \bmod N$. If d is given, the function can be easily inverted using the above equality. We refer to d as a *trapdoor* enabling one to invert the function. In this survey we study the difficulty of inverting

[1]Our description slightly oversimplifies RSA encryption. In practice, messages are padded prior to encryption using some randomness [1]. For instance, a simple (but insufficient) padding algorithm may pad a plaintext M by appending a few random bits to one of the ends prior to encryption. Adding randomness to the encryption process is necessary for proper security.

the RSA function *without* the trapdoor. We refer to this as *breaking RSA*. More precisely, given the triple $\langle N, e, C\rangle$, we ask how *hard* is it to compute the e^{th} root of C modulo $N = pq$ when the factorization of N is unknown. Since $\mathbb{Z}_N^*$ is a finite set, one may enumerate all elements of $\mathbb{Z}_N^*$ until the correct M is found. Unfortunately, this results in an algorithm with running time of order N, namely *exponential* in the size of its input, which is of the order $\log_2 N$. We are interested mostly in algorithms with a substantially lower running time, namely on the order of n^c where $n = \log_2 N$ and c is some small constant (less than 5, say). Such algorithms often perform well in practice on the inputs in question. Throughout the paper we refer to such algorithms as *efficient*.

In this survey we mainly study the RSA *function* as opposed to the RSA *cryptosystem*. Loosely speaking, the difficulty of inverting the RSA function on random inputs implies that given $\langle N, e, C\rangle$ an attacker cannot recover the plaintext M. However, a cryptosystem must resist more subtle attacks. If $\langle N, e, C\rangle$ is given, it should be intractable to recover *any* information about M. This is known as *semantic security*[2]. We do not discuss these subtle attacks, but point out that RSA as described above is not semantically secure: given $\langle N, e, C\rangle$, one can easily deduce some information about the plaintext M (for instance, the Jacobi symbol of M over N can be easily deduced from C). RSA can be made semantically secure by adding randomness to the encryption process [1].

The RSA function $x \longmapsto x^e \bmod N$ is an example of a *trapdoor one-way function*. It can be easily computed, but (as far as we know) cannot be efficiently inverted without the trapdoor d except in special circumstances. Trapdoor one-way functions can be used for *digital signatures* [19]. Digital signatures provide authenticity and nonrepudiation of electronic legal documents. For instance, they are used for signing digital checks or electronic purchase orders. To sign a message $M \in \mathbb{Z}_N^*$ using RSA, Alice applies her private key $\langle N, d\rangle$ to M and obtains a signature $S = M^d \bmod N$. Given $\langle M, S\rangle$, anyone can verify Alice's signature on M by checking that $S^e = M \bmod N$. Since only Alice can generate S, one may suspect that an adversary cannot forge Alice's signature. Unfortunately, things are not so simple; extra measures are needed for proper security. Digital signatures are an important application of RSA. Some of the attacks we survey specifically target RSA digital signatures.

An RSA key pair is generated by picking two random $\frac{n}{2}$-bit primes and multiplying them to obtain N. Then, for a given encryption exponent $e < \varphi(N)$, one computes $d = e^{-1} \bmod \varphi(N)$ using the extended Euclidean algorithm. Since the set of primes is sufficiently dense, a random $\frac{n}{2}$-bit prime can be quickly generated by repeatedly picking random $\frac{n}{2}$-bit integers and testing each one for primality using a probabilistic primality test [19].

1.1 Factoring Large Integers

The first attack on an RSA public key $\langle N, e\rangle$ to consider is factoring the modulus N. Given the factorization of N, an attacker can easily construct $\varphi(N)$, from which the decryption exponent $d = e^{-1} \bmod \varphi(N)$ can be found. We refer to factoring the modulus as a *brute-force attack* on RSA. Although factoring algorithms have been steadily improving, the current state of the art is still far from posing a threat to the security of RSA when RSA is used properly. Factoring large integers is one of the most beautiful problems of computational mathematics [18, 20], but it is not the topic of this article. For completeness we note that the current fastest factoring algorithm is the General Number Field Sieve. Its running time on n-bit integers is $\exp\left((c + o(1))n^{1/3}\log^{2/3} n\right)$ for some $c < 2$. Attacks on RSA that take longer than this time bound are not interesting. These

[2] A source that explains semantic security and gives examples of semantically secure ciphers is [11].

include attacks such as exhaustive search for M and some older attacks published right after the initial publication of RSA.

Our objective is to survey attacks on RSA that decrypt messages without directly factoring the RSA modulus N. Nevertheless, it is worth noting that some sparse sets of RSA moduli, $N = pq$, can be easily factored. For instance, if $p-1$ is a product of prime factors less than B, then N can be factored in time less than B^3. Some implementations explicitly reject primes p for which $p-1$ is a product of small primes.

As noted above, if an efficient factoring algorithm exists, then RSA is insecure. The converse is a long standing open problem: must one factor N in order to efficiently compute e^{th} roots modulo N? Is breaking RSA as hard as factoring? We state the concrete open problem below.

Open Problem 1 *Given integers N and e satisfying $\gcd(e, \varphi(N)) = 1$, define the function $f_{e,N} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ by $f_{e,N}(x) = x^{1/e} \bmod N$. Is there a polynomial-time algorithm $\mathcal{A}$ that computes the factorization of N given N and access to an "oracle" $f_{e,N}(x)$ for some e?*

An *oracle* for $f(x)$ evaluates the function on any input x in unit time. Recently Boneh and Venkatesan [6] provided evidence that for small e the answer to the above problem may be "no". In other words, for small e there may not exist a polynomial-time reduction from factoring to breaking RSA. They do so by showing that in a certain model, a positive answer to the problem for small e yields an efficient factoring algorithm. We note that a positive answer to Open Problem 1 gives rise to a "chosen ciphertext attack"[3] on RSA. Therefore, a negative answer may be welcome.

Next we show that exposing the private key d and factoring N are equivalent. Hence there is no point in hiding the factorization of N from any party who knows d.

Fact 1 *Let $\langle N, e\rangle$ be an RSA public key. Given the private key d, one can efficiently factor the modulus $N = pq$. Conversely, given the factorization of N, one can efficiently recover d.*

Proof A factorization of N yields $\varphi(N)$. Since e is known, one can recover d. This proves the converse statement. We now show that given d one can factor N. Given d, compute $k = de - 1$. By definition of d and e we know that k is a multiple of $\varphi(N)$. Since $\varphi(N)$ is even, $k = 2^t r$ with r odd and $t \geq 1$. We have $g^k = 1$ for every $g \in Z_N^*$, and therefore $g^{k/2}$ is a square root of unity modulo N. By the Chinese Remainder Theorem, 1 has four square roots modulo $N = pq$. Two of these square roots are ± 1. The other two are $\pm x$ where x satisfies $x = 1 \bmod p$ and $x = -1 \bmod q$. Using either one of these last two square roots, the factorization of N is revealed by computing $\gcd(x-1, N)$. A straightforward argument shows that if g is chosen at random from $\mathbb{Z}_N^*$ then with probability at least $1/2$ (over the choice of g) one of the elements in the sequence $g^{k/2}, g^{k/4}, \ldots, g^{k/2^t} \bmod N$ is a square root of unity that reveals the factorization of N. All elements in the sequence can be efficiently computed in time $O(n^3)$ where $n = \log_2 N$. $\square$

2 Elementary Attacks

We begin by describing some old elementary attacks. These attacks illustrate blatant misuse of RSA. Although many such attacks exist, we give only two examples.

[3] In this context, "chosen ciphertext attack" refers to an attacker Marvin, who is given a public key $\langle N, e\rangle$ and access to a black box that decrypts messages of his choice. Marvin succeeds in mounting the chosen ciphertext attack if by using the black box he can recover the private key $\langle N, d\rangle$.

2.1 Common Modulus

To avoid generating a different modulus $N = pq$ for each user one may wish to fix N once and for all. The same N is used by all users. A trusted central authority could provide user i with a unique pair e_i, d_i from which user i forms a public key $\langle N, e_i \rangle$ and a secret key $\langle N, d_i \rangle$.

At first glance this may seem to work: a ciphertext $C = M^{e_a} \bmod N$ intended for Alice cannot be decrypted by Bob since Bob does not possess d_a. However, this is incorrect and the resulting system is insecure. By Fact 1 Bob can use his own exponents e_b, d_b to factor the modulus N. Once N is factored Bob can recover Alice's private key d_a from her public key e_a. This observation, due to Simmons, shows that an RSA modulus should *never* be used by more than one entity.

2.2 Blinding

Let $\langle N, d \rangle$ be Bob's private key and $\langle N, e \rangle$ be his corresponding public key. Suppose an adversary Marvin wants Bob's signature on a message $M \in \mathbb{Z}_N^*$. Being no fool, Bob refuses to sign M. Marvin can try the following: he picks a random $r \in \mathbb{Z}_N^*$ and sets $M' = r^e M \bmod N$. He then asks Bob to sign the random message M'. Bob may be willing to provide his signature S' on the innocent-looking M'. But recall that $S' = (M')^d \bmod N$. Marvin now simply computes $S = S'/r \bmod N$ and obtains Bob's signature S on the original M. Indeed,

$$S^e = (S')^e / r^e = (M')^{ed} / r^e \equiv M'/r^e = M \pmod{N}$$

This technique, called *blinding*, enables Marvin to obtain a valid signature on a message of his choice by asking Bob to sign a random "blinded" message. Bob has no information as to what message he is actually signing. Since most signature schemes apply a "one-way hash" to the message M prior to signing [19], the attack is not a serious concern. Although we presented blinding as an attack, it is actually a useful property of RSA needed for implementing anonymous digital cash (cash that can be used to purchase goods, but does not reveal the identity of the person making the purchase).

3 Low Private Exponent

To reduce decryption time (or signature-generation time), one may wish to use a small value of d rather than a random d. Since modular exponentiation takes time linear in $\log_2 d$, a small d can improve performance by at least a factor of 10 (for a 1024 bit modulus). Unfortunately, a clever attack due to M. Wiener [22] shows that a small d results in a total break of the cryptosystem.

Theorem 2 (M. Wiener) *Let $N = pq$ with $q < p < 2q$. Let $d < \frac{1}{3}N^{1/4}$. Given $\langle N, e \rangle$ with $ed = 1 \bmod \varphi(N)$, Marvin can efficiently recover d.*

Proof The proof is based on approximations using continued fractions. Since $ed = 1 \bmod \varphi(N)$, there exists a k such that $ed - k\varphi(N) = 1$. Therefore,

$$\left| \frac{e}{\varphi(N)} - \frac{k}{d} \right| = \frac{1}{d\varphi(N)}.$$

Hence, $\frac{k}{d}$ is an approximation of $\frac{e}{\varphi(N)}$. Although Marvin does not know $\varphi(N)$, he may use N to approximate it. Indeed, since $\varphi(N) = N - p - q + 1$ and $p + q - 1 < 3\sqrt{N}$, we have $|N - \varphi(N)| < 3\sqrt{N}$.

Using N in place of $\varphi(N)$, we obtain:

$$\begin{aligned}\left|\frac{e}{N}-\frac{k}{d}\right| &= \left|\frac{ed-k\varphi(N)-kN+k\varphi(N)}{Nd}\right| \\ &= \left|\frac{1-k(N-\varphi(N))}{Nd}\right| \le \left|\frac{3k\sqrt{N}}{Nd}\right| = \frac{3k}{d\sqrt{N}}.\end{aligned}$$

Now, $k\varphi(N) = ed - 1 < ed$. Since $e < \varphi(N)$, we see that $k < d < \frac{1}{3}N^{1/4}$. Hence we obtain:

$$\left|\frac{e}{N}-\frac{k}{d}\right| \le \frac{1}{dN^{1/4}} < \frac{1}{2d^2}.$$

This is a classic approximation relation. The number of fractions $\frac{k}{d}$ with $d < N$ approximating $\frac{e}{N}$ so closely is bounded by $\log_2 N$. In fact, all such fractions are obtained as convergents of the continued fraction expansion of $\frac{e}{N}$ [12, Th. 177]. All one has to do is compute the $\log N$ convergents of the continued fraction for $\frac{e}{N}$. One of these will equal $\frac{k}{d}$. Since $ed - k\varphi(N) = 1$, we have $\gcd(k, d) = 1$, and hence $\frac{k}{d}$ is a reduced fraction. This is a linear-time algorithm for recovering the secret key d. $\square$

Since typically N is 1024 bits, it follows that d must be at least 256 bits long in order to avoid this attack. This is unfortunate for low-power devices such as "smartcards", where a small d would result in big savings. All is not lost however. Wiener describes a number of techniques that enable fast decryption and are not susceptible to his attack:

Large e: Suppose instead of reducing e modulo $\varphi(N)$, one uses $\langle N, e'\rangle$ for the public key, where $e' = e + t \cdot \varphi(N)$ for some large t. Clearly e' can be used in place of e for message encryption. However, when a large value of e is used, the k in the above proof is no longer small. A simple calculation shows that if $e' > N^{1.5}$ then no matter how small d is, the above attack cannot be mounted. Unfortunately, large values of e result in increased encryption time.

Using CRT: An alternate approach is to use the Chinese Remainder Theorem (CRT). Suppose one chooses d such that both $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$ are small, say 128 bits each. Then fast decryption of a ciphertext C can be carried out as follows: first compute $M_p = C^{d_p} \bmod p$ and $M_q = C^{d_q} \bmod q$. Then use the CRT to compute the unique value $M \in \mathbb{Z}_N$ satisfying $M = M_p \bmod p$ and $M = M_q \bmod q$. The resulting M satisfies $M = C^d \bmod N$ as required. The point is that although d_p and d_q are small, the value of $d \bmod \varphi(N)$ can be large, i.e., on the order of $\varphi(N)$. As a result, the attack of Theorem 2 does not apply. We note that if $\langle N, e\rangle$ is given, there exists an attack enabling an adversary to factor N in time $O\left(\min(\sqrt{d_p}, \sqrt{d_q})\right)$. Hence, d_p and d_q cannot be made too small.

We do not know whether either of these methods is secure. All we know is that Wiener's attack is ineffective against them. Theorem 2 was recently improved by Boneh and Durfee [4], who show that as long as $d < N^{0.292}$, an adversary can efficiently recover d from $\langle N, e\rangle$. These results show that Wiener's bound is not tight. It is likely that the correct bound is $d < N^{0.5}$. At the time of this writing, this is an open problem.

Open Problem 2 *Let $N = pq$ and $d < N^{0.5}$. If Marvin is given $\langle N, e\rangle$ with $ed = 1 \bmod \varphi(N)$ and $e < \varphi(N)$, can he efficiently recover d?*

4 Low Public Exponent

To reduce encryption or signature-verification time, it is customary to use a small public exponent e. The smallest possible value for e is 3, but to defeat certain attacks the value $e = 2^{16} + 1 = 65537$ is recommended. When the value $2^{16} + 1$ is used, signature verification requires 17 multiplications, as opposed to roughly 1000 when a random $e \leq \varphi(N)$ is used. Unlike the attack of the previous section, attacks that apply when a small e is used are far from a total break.

4.1 Coppersmith's Theorem

The most powerful attacks on low public exponent RSA are based on a theorem due to Coppersmith [7]. Coppersmith's theorem has many applications, only some of which will be covered here. The proof uses the LLL lattice basis reduction algorithm [17] as explained below.

Theorem 3 (Coppersmith) *Let N be an integer and $f \in \mathbb{Z}[x]$ be a monic polynomial of degree d. Set $X = N^{\frac{1}{d}-\epsilon}$ for some $\epsilon \geq 0$. Then, given $\langle N, f\rangle$ Marvin can efficiently find all integers $|x_0| < X$ satisfying $f(x_0) = 0 \bmod N$. The running time is dominated by the time it takes to run the LLL algorithm on a lattice of dimension $O(w)$ with $w = \min(1/\epsilon, \log_2 N)$.*

The theorem provides an algorithm for efficiently finding all roots of f modulo N that are less than $X = N^{1/d}$. As X gets smaller, the algorithm's running time decreases. The theorem's strength is its ability to find small roots of polynomials modulo a composite N. When working modulo a prime, there is no reason to use Coppersmith's theorem since other, far better, root-finding algorithms exist.

We sketch the main ideas behind the proof of Coppersmith's theorem. We follow a simplified approach due to Howgrave-Graham [14]. Given a polynomial $h(x) = \sum a_i x^i \in \mathbb{Z}[x]$, define $\|h\|^2 = \sum_i |a_i|^2$. The proof relies on the following observation.

Lemma 4 *Let $h(x) \in \mathbb{Z}[x]$ be a polynomial of degree d and let X be a positive integer. Suppose $\|h(xX)\| < N/\sqrt{d}$. If $|x_0| < X$ satisfies $h(x_0) = 0 \bmod N$, then $h(x_0) = 0$ holds over the integers.*

Proof Observe from the Schwarz inequality that

$$\begin{aligned} |h(x_0)| &= \left|\sum a_i x_0^i\right| = \left|\sum a_i X^i \left(\frac{x_0}{X}\right)^i\right| \leq \sum \left|a_i X^i \left(\frac{x_0}{X}\right)^i\right| \\ &\leq \sum \left|a_i X^i\right| \leq \sqrt{d}\,\|h(xX)\| < N. \end{aligned}$$

Since $h(x_0) = 0 \bmod N$, we conclude that $h(x_0) = 0$. □

The lemma states that if h is a polynomial with low norm, then all small roots of $h \bmod N$ are also roots of h over the integers. The lemma suggests that to find a small root x_0 of $f(x) \bmod N$ we should look for another polynomial $h \in \mathbb{Z}[x]$ with small norm having the same roots as f modulo N. Then x_0 will be a root of h over the integers and can be easily found. To do so, we may search for a polynomial $g \in \mathbb{Z}[x]$ such that $h = gf$ has low norm, i.e., norm less than N. This amounts to searching for an *integer linear combination* of the polynomials $f, xf, x^2 f, \ldots, x^r f$ with low norm. Unfortunately, most often there is no nontrivial linear combination with sufficiently small norm.

Coppersmith found a trick to solve the problem: if $f(x_0) = 0 \bmod N$, then $f(x_0)^k = 0 \bmod N^k$ for any k. More generally, define the following polynomials:

$$g_{u,v}(x) = N^{m-v} x^u f(x)^v$$

for some predefined m. Then x_0 is a root of $g_{u,v}(x)$ modulo N^m for any $u \geq 0$ and $0 \leq v \leq m$. To use Lemma 4 we must find an integer linear combination $h(x)$ of the polynomials $g_{u,v}(x)$ such that $h(xX)$ has norm less than N^m (recall that X is an upper bound on x_0 satisfying $X \leq N^{1/d}$). Thanks to the relaxed upper bound on the norm (N^m rather than N), one can show that for sufficiently large m, there always exists a linear combination $h(x)$ satisfying the required bound. Once $h(x)$ is found, Lemma 4 implies that it has x_0 as a root over the integers. Consequently x_0 can be easily found.

It remains to show how to find $h(x)$ efficiently. To do so, we must first state a few basic facts about lattices in $\mathbb{Z}^w$. We refer to [17] for a concise introduction to the topic. Let $u_1, \ldots, u_w \in \mathbb{Z}^w$ be linearly independent vectors. A (full-rank) lattice L spanned by $\langle u_1, \ldots, u_w \rangle$ is the set of all *integer* linear combinations of $u_1, \ldots, u_w$. The determinant of L is defined as the determinant of the $w \times w$ square matrix whose rows are the vectors $u_1, \ldots, u_w$.

In our case, we view the polynomials $g_{u,v}(xX)$ as vectors and study the lattice L spanned by them. We let $v = 0, \ldots, m$ and $u = 0, \ldots, d-1$, and hence the lattice has dimension $w = d(m+1)$. For example, when f is a quadratic monic polynomial and $m = 3$, the resulting lattice is spanned by the rows of the following matrix:

$$\begin{array}{cc}
 & \begin{array}{cccccccc} 1 & x & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 \end{array} \\
\begin{array}{c} g_{0,0}(xX) \\ g_{1,0}(xX) \\ g_{0,1}(xX) \\ g_{1,1}(xX) \\ g_{0,2}(xX) \\ g_{1,2}(xX) \\ g_{0,3}(xX) \\ g_{1,3}(xX) \end{array} &
\begin{bmatrix}
N^3 & & & & & & & \\
 & XN^3 & & & & & & \\
* & * & X^2N^2 & & & & & \\
 & * & * & X^3N^2 & & & & \\
* & * & * & * & X^4N & & & \\
 & * & * & * & * & X^5N & & \\
* & * & * & * & * & * & X^6 & \\
 & * & * & * & * & * & * & X^7
\end{bmatrix}
\end{array}$$

The entries $*$ correspond to coefficients of the polynomials whose value we ignore. All empty entries are zero. Since the matrix is triangular, its determinant is the product of the elements on the diagonal (which are explicitly given above). Our objective is to find short vectors in this lattice.

A classic result of Hermite states that any lattice L of dimension w contains a nonzero point $v \in L$ whose L_2 norm satisfies $\|v\| \leq \gamma_w \det(L)^{1/w}$, where γ_w is a constant depending only on w. Hermite's bound can be used to show that for large enough m our lattice contains vectors of norm less than N^m, as required. The question is whether we can efficiently construct a short vector in L whose length is not much larger than the Hermite bound. The LLL algorithm is an efficient algorithm that does precisely that.

Fact 5 (LLL) *Let L be a lattice spanned by $\langle u_1, \ldots, u_w \rangle$. When $\langle u_1, \ldots, u_w \rangle$ are given as input, then the LLL algorithm outputs a point $v \in L$ satisfying*

$$\|v\| \leq 2^{w/4} \det(L)^{1/w}.$$

The running time for LLL is quartic in the length of the input.

The LLL algorithm (named after its inventors L. Lovasz, A. Lenstra, and H. Lenstra Jr.) has many applications in both computational number theory and cryptography. Its discovery in 1982 provided an efficient algorithm for factoring polynomials over the integers and, more generally, over number rings. LLL is frequently used to attack various cryptosystems. For instance, many cryptosystems based on the "knapsack problem" have been broken using LLL.

Using LLL, we can complete the proof of Coppersmith's theorem. To ensure that the vector produced by LLL satisfies the bound of Lemma 4 we need

$$2^{w/4} \det(L)^{1/w} < N^m/\sqrt{w},$$

where $w = d(m+1)$ is the dimension of L. A routine calculation shows that for large enough m the bound is satisfied. Indeed, when $X = N^{\frac{1}{d}-\epsilon}$, it suffices to take $m = O(k/d)$ with $k = \min(\frac{1}{\epsilon}, \log N)$. Consequently, the running time is dominated by running LLL on a lattice of dimension $O(k)$, as required.

A natural question is whether Coppersmith's theorem can be applied to bivariate and multivariate polynomials. If $f(x, y) \in \mathbb{Z}_N[x, y]$ is given for which there exists a root (x_0, y_0) with $|x_0 y_0|$ suitably bounded, can Marvin efficiently find (x_0, y_0)? Although the same technique appears to work for some bivariate polynomials, it is currently an open problem to prove it. As an increasing number of results depend on a bivariate extension of Coppersmith's theorem, a rigorous algorithm will be very useful.

Open Problem 3 *Find general conditions under which Coppersmith's theorem can be generalized to bivariate polynomials.*

4.2 Hastad's Broadcast Attack

As a first application of Coppersmith's theorem, we present an improvement to an old attack due to Hastad [13]. Suppose Bob wishes to send an encrypted message M to a number of parties $P_1, P_2, \ldots, P_k$. Each party has its own RSA key $\langle N_i, e_i \rangle$. We assume M is less than all the N_i's. Naively, to send M, Bob encrypts it using each of the public keys and sends out the i^{th} ciphertext to P_i. An attacker Marvin can eavesdrop on the connection out of Bob's sight and collect the k transmitted ciphertexts.

For simplicity, suppose all public exponents e_i are equal to 3. A simple argument shows that Marvin can recover M if $k \geq 3$. Indeed, Marvin obtains C_1, C_2, C_3, where

$$C_1 = M^3 \bmod N_1, \qquad C_2 = M^3 \bmod N_2, \qquad C_3 = M^3 \bmod N_3.$$

We may assume that $\gcd(N_i, N_j) = 1$ for all $i \neq j$ since otherwise Marvin can factor some of the N_i's. Hence, applying the Chinese Remainder Theorem (CRT) to C_1, C_2, C_3 gives a $C' \in \mathbb{Z}_{N_1 N_2 N_3}$ satisfying $C' = M^3 \bmod N_1 N_2 N_3$. Since M is less than all the N_i's, we have $M^3 < N_1 N_2 N_3$. Then $C' = M^3$ holds over the integers. Thus, Marvin may recover M by computing the real cube root of C'. More generally, if all public exponents are equal to e, Marvin can recover M as soon as $k \geq e$. The attack is feasible only when a small e is used.

Hastad [13] describes a far stronger attack. To motivate Hastad's result, consider a naive defense against the above attack. Rather than broadcasting the encryption of M, Bob could "pad" the message prior to encryption. For instance, if M is m bits long, Bob could send $M_i = i2^m + M$ to party P_i. Since Marvin obtains encryptions of different messages, he cannot mount the attack.

Unfortunately, Hastad showed that this linear padding is insecure. In fact, he proved that applying any fixed polynomial to the message prior to encryption does not prevent the attack.

Suppose that for each of the participants $P_1, \ldots, P_k$, Bob has a fixed public polynomial $f_i \in \mathbb{Z}_{N_i}[x]$. To broadcast a message M, Bob sends the encryption of $f_i(M)$ to party P_i. By eavesdropping, Marvin learns $C_i = f_i(M)^{e_i} \bmod N_i$ for $i = 1, \ldots, k$. Hastad showed that if enough parties are involved, Marvin can recover the plaintext M from all the ciphertexts. The following theorem is a stronger version of Hastad's original result.

Theorem 6 (Hastad) *Let $N_1, \ldots N_k$ be pairwise relatively prime integers and set $N_{min} = \min_i(N_i)$. Let $g_i \in \mathbb{Z}_{N_i}[x]$ be k polynomials of maximum degree d. Suppose there exists a unique $M < N_{min}$ satisfying*

$$g_i(M) = 0 \bmod N_i \qquad \text{for all } i = 1, \ldots, k.$$

Under the assumption that $k > d$, one can efficiently find M given $\langle N_i, g_i \rangle_{i=1}^k$.

Proof Let $\overline{N} = N_1 \cdots N_k$. We may assume that all g_i's are monic. (Indeed if, for some i, the leading coefficient of g_i is not invertible in $\mathbb{Z}_{N_i}^*$, then the factorization of N_i is exposed.) By multiplying each g_i by the appropriate power of x, we may assume they all have degree d. Construct the polynomial

$$g(x) = \sum_{i=1}^{k} T_i g_i(x), \quad \text{where } T_i = \begin{array}{ll} 1 \bmod N_j & \text{if } i = j \\ 0 \bmod N_j & \text{if } i \neq j. \end{array}$$

The T_i's are integers known as the Chinese Remainder Coefficients. Then $g(x)$ must be monic since it is monic modulo all the N_i. Its degree is d. Furthermore, we know that $g(M) = 0 \bmod \overline{N}$. Theorem 6 now follows from Theorem 3 since $M < N_{min} \le \overline{N}^{\,1/k} < \overline{N}^{\,1/d}$. □

The theorem shows that a system of univariate equations modulo relatively prime composites can be efficiently solved, assuming sufficiently many equations are provided. By setting $g_i = f_i^{e_i} - C_i \bmod N_i$, we see that Marvin can recover M from the given ciphertexts whenever the number of parties is at least d, where d is the maximum of $e_i \deg(f_i)$ over all $i = 1, \ldots, k$. In particular, if all e_i's are equal to e and Bob sends out *linearly* related messages, then Marvin can recover the plaintext as soon as $k > e$.

Hastad's original theorem is weaker than the one stated above. Rather than d polynomials, Hastad required $d(d+1)/2$ polynomials. Hastad's proof is similar to the proof of Coppersmith's theorem described in the previous section. However, Hastad does not use powers of g in the lattice and consequently obtains a weaker bound.

To conclude this section we note that to properly defend against the broadcast attack above, one must use a *randomized* pad [1] rather than a fixed one.

4.3 Franklin-Reiter Related Message Attack

Franklin and Reiter [8] found a clever attack when Bob sends Alice related encrypted messages using the same modulus. Let $\langle N, e \rangle$ be Alice's public key. Suppose $M_1, M_2 \in \mathbb{Z}_N^*$ are two distinct messages satisfying $M_1 = f(M_2) \bmod N$ for some publicly known polynomial $f \in \mathbb{Z}_N[x]$. To send M_1 and M_2 to Alice, Bob may naively encrypt the messages and transmit the resulting ciphertexts C_1, C_2. We show that given C_1, C_2, Marvin can easily recover M_1, M_2. Although the attack works for any small e, we state the following lemma for $e = 3$ in order to simplify the proof.

Lemma 7 (FR) *Set $e = 3$ and let $\langle N, e\rangle$ be an RSA public key. Let $M_1 \neq M_2 \in Z_N^*$ satisfy $M_1 = f(M_2) \bmod N$ for some linear polynomial $f = ax + b \in \mathbb{Z}_N[x]$ with $b \neq 0$. Then, given $\langle N, e, C_1, C_2, f\rangle$, Marvin can recover M_1, M_2 in time quadratic in $\log N$.*

Proof To keep this part of the proof general, we state it using an arbitrary e (rather than restricting to $e = 3$). Since $C_1 = M_1^e \bmod N$, we know that M_2 is a root of the polynomial $g_1(x) = f(x)^e - C_1 \in \mathbb{Z}_N[x]$. Similarly, M_2 is a root of $g_2(x) = x^e - C_2 \in \mathbb{Z}_N[x]$. The linear factor $x - M_2$ divides both polynomials. Therefore, Marvin may use the Euclidean algorithm[4] to compute the gcd of g_1 and g_2. If the gcd turns out to be linear, M_2 is found. The gcd can be computed in quadratic time in e and $\log N$.

We show that when $e = 3$ the gcd must be linear. The polynomial $x^3 - C_2$ factors modulo both p and q into a linear factor and an irreducible quadratic factor (recall that $\gcd(e, \varphi(N)) = 1$ and hence $x^3 - C_2$ has only one root in $\mathbb{Z}_N$). Since g_2 cannot divide g_1, the gcd must be linear. For $e > 3$ the gcd is almost always linear. However, for some rare M_1, M_2, and f, it is possible to obtain a nonlinear gcd, in which case the attack fails. $\square$

For $e > 3$ the attack takes time quadratic in e. Consequently, it can be applied only when a small public exponent e is used. For large e the work in computing the gcd is prohibitive. It is an interesting question (though likely to be difficult) to devise such an attack for arbitrary e. In particular, can the gcd of g_1 and g_2 above be found in time polynomial in $\log e$?

4.4 Coppersmith's Short Pad Attack

The Franklin-Reiter attack might seem a bit artificial. After all, why should Bob send Alice the encryption of related messages? Coppersmith strengthened the attack and proved an important result on padding [7].

A naive random padding algorithm might pad a plaintext M by appending a few random bits to one of the ends. The following attack points out the danger of such simplistic padding. Suppose Bob sends a properly-padded encryption of M to Alice. An attacker, Marvin, intercepts the ciphertext and prevents it from reaching its destination. Bob notices that Alice did not respond to his message and decides to resend M to Alice. He randomly pads M and transmits the resulting ciphertext. Marvin now has two ciphertexts corresponding to two encryptions of the same message using two different random pads. The following theorem shows that although he does not know the pads used, Marvin is able to recover the plaintext.

Theorem 8 *Let $\langle N, e\rangle$ be a public RSA key where N is n-bits long. Set $m = \lfloor n/e^2 \rfloor$. Let $M \in \mathbb{Z}_N^*$ be a message of length at most $n - m$ bits. Define $M_1 = 2^m M + r_1$ and $M_2 = 2^m M + r_2$, where r_1 and r_2 are distinct integers with $0 \leq r_1, r_2 < 2^m$. If Marvin is given $\langle N, e\rangle$ and the encryptions C_1, C_2 of M_1, M_2 (but is not given r_1 or r_2), he can efficiently recover M.*

Proof Define $g_1(x, y) = x^e - C_1$ and $g_2(x, y) = (x + y)^e - C_2$. We know that when $y = r_2 - r_1$, these polynomials have M_1 as a common root. In other words, $\Delta = r_2 - r_1$ is a root of the "resultant" $h(y) = \mathrm{res}_x(g_1, g_2) \in \mathbb{Z}_N[y]$. The degree of h is at most e^2. Furthermore, $|\Delta| < 2^m < N^{1/e^2}$. Hence, Δ is a small root of h modulo N, and Marvin can efficiently find it using Coppersmith's theorem (Theorem 3). Once Δ is known, the Franklin-Reiter attack of the previous section can be

[4] Although $\mathbb{Z}_N[x]$ is not a Euclidean ring, the standard Euclidean algorithm can still be applied to polynomials in $\mathbb{Z}_N[x]$. One can show that if the algorithm "breaks" in any way, then the factorization of N is exposed.

used to recover M_2 and consequently M. $\square$

When $e = 3$ the attack can be mounted as long as the pad length is less than $1/9^{th}$ the message length. This is an important result. Note that for the recommended value of $e = 65537$, the attack is useless against standard moduli sizes.

4.5 Partial Key Exposure Attack

Let $\langle N, d \rangle$ be a private RSA key. Suppose by some means Marvin is able to expose a fraction of the bits of d, say a quarter of them. Can he reconstruct the rest of d? Surprisingly, the answer is positive when the corresponding public key is small. Recently Boneh, Durfee, and Frankel [5] showed that as long as $e < \sqrt{N}$, it is possible to reconstruct all of d from just a fraction of its bits. These results illustrate the importance of safeguarding the *entire* private RSA key.

Theorem 9 (BDF) *Let $\langle N, d \rangle$ be a private RSA key in which N is n bits long. Given the $\lceil n/4 \rceil$ least significant bits of d, Marvin can reconstruct all of d in time linear in $e \log_2 e$.*

The proof relies on yet another beautiful theorem due to Coppersmith [7].

Theorem 10 (Coppersmith) *Let $N = pq$ be an n-bit RSA modulus. Then given the $n/4$ least significant bits of p or the $n/4$ most significant bits of p, one can efficiently factor N.*

Theorem 9 readily follows from Theorem 10. In fact, by definition of e and d, there exists an integer k such that

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

Since $d < \varphi(N)$, we must have $0 < k \le e$. Reducing the equation modulo $2^{n/4}$ and setting $q = N/p$, we obtain

$$(ed)p - kp(N - p + 1) + kN = p \pmod{2^{n/4}}.$$

Since Marvin is given the $n/4$ least significant bits of d, he knows the value of $ed \bmod 2^{n/4}$. Consequently, he obtains an equation in k and p. For each of the e possible values of k, Marvin solves the quadratic equation in p and obtains a number of candidate values for $p \bmod 2^{n/4}$. For each of these candidate values, he runs the algorithm of Theorem 10 to attempt to factor N. One can show that the total number of candidate values for $p \bmod 2^{n/4}$ is at most $e \log_2 e$. Hence after at most $e \log_2 e$ attempts, N will be factored.

Theorem 9 is known as a *partial key-exposure* attack. Similar attacks exist for larger values of e as long as $e < \sqrt{N}$. However the techniques are a bit more complex [5]. It is interesting that discrete log-based cryptosystems, such as the ElGamal public key system, do not seem susceptible to partial key exposure. Indeed, if $g^x \bmod p$ and a constant fraction of the bits of x are given, there is no known polynomial-time algorithm to compute the rest of x.

To conclude the section we show that when the encryption exponent e is small, the RSA system leaks *half* the most significant bits of the corresponding private key d. To see this, consider once again the equation $ed - k(N - p - q + 1) = 1$ for an integer $0 < k \le e$. Given k, Marvin may easily compute

$$\widehat{d} = \lfloor (kN + 1)/e \rfloor.$$

Then

$$|\widehat{d} - d| \le k(p + q)/e \le 3k\sqrt{N}/e < 3\sqrt{N}.$$

Hence, $\widehat{d}$ is a good approximation for d. The bound shows that, for most d, half the most significant bits of $\widehat{d}$ are equal to those of d. Since there are only e possible values for k, Marvin can construct a small set of size e such that one of the elements in the set is equal to half the most significant bits of d. The case $e = 3$ is especially interesting. In this case one can show that always $k = 2$ and hence the system completely leaks half the most significant bits of d.

5 Implementation Attacks

We turn our attention to an entirely different class of attacks. Rather than attacking the underlying structure of the RSA function, these attacks focus on the implementation of RSA.

5.1 Timing Attacks

Consider a smartcard that stores a private RSA key. Since the card is tamper resistant, an attacker Marvin may not be able to examine its contents and expose the key. However, a clever attack due to Kocher [16] shows that by precisely measuring the *time* it takes the smartcard to perform an RSA decryption (or signature), Marvin can quickly discover the private decryption exponent d.

We explain how to mount the attack against a simple implementation of RSA using the "repeated squaring algorithm". Let $d = d_n d_{n-1} \ldots d_0$ be the binary representation of d (i.e., $d = \sum_{i=0}^{n} 2^i d_i$ with $d_i \in \{0, 1\}$). The *repeated squaring algorithm* computes $C = M^d \bmod N$, using at most $2n$ modular multiplications. It is based on the observation that $C = \prod_{i=0}^{n} M^{2^i d_i} \bmod N$. The algorithm works as follows:

> Set z equal to M and C equal to 1. For $i = 0, \ldots, n$, do these steps:
> (1) if $d_i = 1$ set C equal to $C \cdot z \bmod N$,
> (2) set z equal to $z^2 \bmod N$.
> At the end, C has the value $M^d \bmod N$.

The variable z runs through the set of values $M^{2^i} \bmod N$ for $i = 0, \ldots, n$. The variable C "collects" the appropriate powers in the set to obtain $M^d \bmod N$.

To mount the attack, Marvin asks the smartcard to generate signatures on a large number of random messages $M_1, \ldots, M_k \in \mathbb{Z}_N^*$ and measures the time T_i it takes the card to generate each of the signatures.

The attack recovers bits of d one at a time beginning with the least significant bit. We know d is odd. Thus $d_0 = 1$. Consider the second iteration. Initially $z = M^2 \bmod N$ and $C = M$. If $d_1 = 1$, the smartcard computes the product $C \cdot z = M \cdot M^2 \bmod N$. Otherwise, it does not. Let t_i be the time it takes the smartcard to compute $M_i \cdot M_i^2 \bmod N$. The t_i's differ from each other since the time to compute $M_i \cdot M_i^2 \bmod N$ depends on the value of M_i (simple modular reduction algorithms take a different amount of time depending on the value being reduced). Marvin measures the t_i's offline (prior to mounting the attack) once he obtains the physical specifications of the card.

Kocher observed that when $d_1 = 1$, the two ensembles $\{t_i\}$ and $\{T_i\}$ are correlated. For instance, if, for some i, t_i is much larger than its expectation, then T_i is also likely to be larger than its expectation. On the other hand, if $d_1 = 0$, the two ensembles $\{t_i\}$ and $\{T_i\}$ behave as independent random variables. By measuring the correlation, Marvin can determine whether d_1 is 0 or 1.

Continuing in this way, he can recover d_2, d_3, and so on. Note that when a low public exponent e is used, the partial key exposure attack of the previous section shows that Kocher's timing attack need only be employed until a quarter of the bits of d are discovered.

There are two ways to defend against the attack. The simplest is to add appropriate delay so that modular exponentiation always takes a fixed amount of time. The second approach, due to Rivest, is based on *blinding*. Prior to decryption of M the smartcard picks a random $r \in \mathbb{Z}_N^*$ and computes $M' = M \cdot r^e \bmod N$. It then applies d to M' and obtains $C' = (M')^d \bmod N$. Finally, the smartcard sets $C = C'/r \bmod N$. With this approach, the smartcard is applying d to a random message M' unknown to Marvin. As a result, Marvin cannot mount the attack.

Kocher recently discovered another attack along these lines called *power cryptanalysis*. Kocher showed that by precisely measuring the smartcard's *power consumption* during signature generation, Marvin can often easily discover the secret key. As it turns out, during a multi-precision multiplication the card's power consumption is higher than normal. By measuring the length of high consumption periods, Marvin can easily determine if in a given iteration the card performs one or two multiplications, thus exposing the bits of d.

5.2 Random Faults

Implementations of RSA decryption and signatures frequently use the Chinese Remainder Theorem to speed up the computation of $M^d \bmod N$. Instead of working modulo N, the signer Bob first computes the signatures modulo p and q and then combines the results using the Chinese Remainder Theorem. More precisely, Bob first computes

$$C_p = M^{d_p} \bmod p \qquad \text{and} \qquad C_q = M^{d_q} \bmod q,$$

where $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$. He then obtains the signature C by setting

$$C = T_1 C_p + T_2 C_q \pmod{N},$$

where

$$T_1 = \left\{ \begin{array}{c} 1 \bmod p \\ 0 \bmod q \end{array} \right\} \qquad \text{and} \qquad T_2 = \left\{ \begin{array}{c} 0 \bmod p \\ 1 \bmod q \end{array} \right\}.$$

The running time of the last CRT step is negligible compared to the two exponentiations. Note that p and q are half the length of N. Since simple implementations of multiplication take quadratic time, multiplication modulo p is four times faster than modulo N. Furthermore, d_p is half the length of d and consequently computing $M^{d_p} \bmod p$ is eight times faster than computing $M^d \bmod N$. Overall signature time is thus reduced by a factor of four. Many implementations use this method to improve performance.

Boneh, DeMillo, and Lipton [3] observed that there is an inherent danger in using the CRT method. Suppose that while generating a signature, a glitch on Bob's computer causes it to miscalculate in a single instruction. Say, while copying a register from one location to another, one of the bits is flipped. (A glitch may be caused by ambient electromagnetic interference or perhaps by a rare hardware bug, like the one found in an early version of the Pentium chip.) Given an invalid signature, an adversary Marvin can easily factor Bob's modulus N.

We present a version of the attack as described by A. K. Lenstra. Suppose a single error occurs while Bob is generating a signature. As a result, exactly one of C_p or C_q will be computed incorrectly. Say C_p is correct, but $\widehat{C}_q$ is not. The resulting signature is $\widehat{C} = T_1 C_p + T_2 \widehat{C}_q$. Once Marvin receives $\widehat{C}$, he knows it is a false signature since $\widehat{C}^e \neq M \bmod N$. However, notice that

$$\widehat{C}^e = M \bmod p \qquad \text{while} \qquad \widehat{C}^e \neq M \bmod q.$$

As a result, $\gcd(N, \widehat{C}^e - M)$ exposes a nontrivial factor of N.

For the attack to work, Marvin must have full knowledge of M. Namely, we are assuming Bob does not use any random padding procedure. Random padding prior to signing defeats the attack. A simpler defense is for Bob to check the generated signature before sending it out to the world. Checking is especially important when using the CRT speedup method. Random faults are hazardous to many cryptographic systems. Many systems, including a non-CRT implementation of RSA, can be attacked using random faults [3]. However, these results are far more theoretical.

5.3 Bleichenbacher's Attack on PKCS 1

Let N be an n-bit RSA modulus and M be an m bit message with $m < n$. Before applying RSA encryption it is natural to pad the message M to n bits by appending random bits to it. An old version of a standard known as Public Key Cryptography Standard 1 (PKCS 1) uses this approach. After padding, the message looks as follows:

02	Random	00	M

The resulting message is n bits long and is directly encrypted using RSA. The initial block containing "02" is 16 bits long and is there to indicate that a random pad has been added to the message.

When a PKCS 1 message is received by Bob's machine, an application (e.g., a web browser) decrypts it, checks the initial block, and strips off the random pad. However, some applications check for the "02" initial block and if it is not present they send back an error message saying "invalid ciphertext". Bleichenbacher [2] showed that this error message can lead to disastrous consequences: using the error message, an attacker Marvin can decrypt ciphertexts of his choice.

Suppose Marvin intercepts a ciphertext C intended for Bob and wants to decrypt it. To mount the attack, Marvin picks a random $r \in \mathbb{Z}_N^*$, computes $C' = rC \bmod N$, and sends C' to Bob's machine. An application running on Bob's machine receives C' and attempts to decrypt it. It either responds with an error message or does not respond at all (if C' happens to be properly formatted). Hence, Marvin learns whether the most significant 16 bits of the decryption of C' are equal to 02. In effect, Marvin has an *oracle* that tests for him whether the 16 most significant bits of the decryption of $rC \bmod N$ are equal to 02, for any r of his choice. Bleichenbacher showed that such an oracle is sufficient for decrypting C.

6 Conclusions

Two decades of research into inverting the RSA function produced some insightful attacks, but no devastating attack has ever been found. The attacks discovered so far mainly illustrate the pitfalls to be avoided when implementing RSA. At the moment it appears that proper implementations can be trusted to provide security in the digital world.

We categorized attacks on RSA into four categories: (1) elementary attacks that exploit blatant misuse of the system, (2) low private exponent attacks serious enough that a low private exponent should never be used, (3) low public exponent attacks, (4) and attacks on the implementation. These last attacks illustrate that a study of the underlying mathematical structure is insufficient. Desmedt and Odlyzko [10], Joye and Quisquater [15] and deJonge and Chaum [9] describe some additional attacks. Throughout the paper we observed that many attacks can be defeated by properly padding the message prior to encryption or signing.

Acknowledgments

I thank Susan Landau for encouraging me to write the survey and Tony Knapp for his help in editing the manuscript. I am also grateful to Mihir Bellare, Igor Shparlinski, and R. Venkatesan for comments on an earlier draft.

References

[1] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1994.

[2] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1998.

[3] D. Boneh, R. DeMillo, and R. Lipton. On the importance of checking cryptographic protocols for faults. In *EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 1997.

[4] D. Boneh and G. Durfee. New results on cryptanalysis of low private exponent RSA. Preprint, 1998.

[5] D. Boneh, G. Durfee, and Y. Frankel. An attack on RSA given a fraction of the private key bits. In *AsiaCrypt '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 25–34. Springer-Verlag, 1998.

[6] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71. Springer-Verlag, 1998.

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

[8] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter. Low-exponent RSA with related messages. In *EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 1–9. Springer-Verlag, 1996.

[9] W. de Jonge and D. Chaum. Attacks on some RSA signatures. In *Crypto '85*, volume 218 of *Lecture Notes in Computer Science*, pages 18–27. Springer-Verlag, 1986.

[10] Y. Desmedt and A. Odlyzko. A chosen text attack on the rsa cryptosystem and some discrete logarithm schemes. In *CRYPTO '85*, Lecture Notes in Computer Science, pages 516–522. Springer-Verlag, 1985.

[11] S. Goldwasser. The search for provably secure cryptosystems. In *Cryptology and computational number theory*, volume 42 of *Proceedings of the 42nd Symposium in Applied Mathematics.* American Mathematical Society, 1990.

[12] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers.* Oxford Clarendon Press, 1975. fourth edition.

[13] J. Hastad. Solving simultaneous modular equations of low degree. *SIAM J. of Computing*, 17:336–341, 1988.

[14] N. Howgrave-Graham. Finding small roots of univariate modular equations revisited. In *Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 131–142. Springer-Verlag, 1997.

[15] M. Joye and J.-J. Quisquater. On the importance of securing your bins: The garbage-man-in-the-middle attack. In *4th ACM Conference on Computer and Communications Security*, pages 135–141. ACM Press, 1997.

[16] P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

[17] L. Lovász. *An Algorithmic Theory of Number, Graphs and Convexity.* SIAM Publications, 1986.

[18] A. K. Lenstra and H. W. Lenstra, Jr. Algorithms in number theory. In *Handbook of Theoretical Computer Science (Volume A: Algorithms and Complexity)*, chapter 12, pages 673–715. Elsevier and MIT Press, 1990.

[19] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC, 1996.

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

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

[22] M. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36:553–558, 1990.