

# Exposing an RSA Private Key Given a Small Fraction of its Bits

Dan Boneh\*  
dabo@cs.stanford.edu  
Stanford University

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

Yair Frankel  
yfrankel@cs.columbia.edu  
Certco

## Abstract

We show that for low public exponent RSA, given a quarter of the bits of the private key an adversary can recover the entire private key. Similar results (though not as strong) are obtained for larger values of  $e$ . For instance, when  $e$  is a prime in the range  $[N^{1/4}, N^{1/2}]$ , half the bits of the private key suffice to reconstruct the entire private key. Our results point out the danger of partial key exposure in the RSA public key system.

## 1 Introduction

Let  $N = pq$  be an RSA modulus and let  $e, d$  be encryption/decryption exponents, i.e.  $ed = 1 \pmod{\phi(N)}$ . We study the following question: how many bits of  $d$  does an adversary require in order to reconstruct all of  $d$ ? Surprisingly, we show that for low public exponent RSA, given only a quarter of the least significant bits of  $d$ , an adversary can efficiently recover all of  $d$ . We obtain similar results, summarized in the next subsection, for larger values of  $e$  as well. Our results show that RSA, and particularly low public exponent RSA, are vulnerable to *partial key exposure*. We refer to this class of attacks as *partial key exposure attacks*.

To motivate this problem consider a computer system which has an RSA private key stored on it. An adversary may attempt to attack the system in a variety of ways in order to obtain the private key. Some attacks (e.g. a timing attack [4]) are able to reveal some bits of the key, but may fail to reveal the entire key [7]. Our results show that attacks, such as the timing attack on RSA, need only be carried out until a quarter of the least significant bits of  $d$  are exposed. Once these bits are revealed the adversary can efficiently compute all of  $d$ . Another scenario where partial key exposure comes up is in the presence of covert channels. Such channels are often slow or have a bounded capacity. Our results show that as long as a fraction of the private exponent bits can be leaked, the remaining bits can be reconstructed.

It is natural to ask the analogous question in the context of discrete log schemes. For instance, given a fraction of the bits of the private key in the ElGamal public key system [2], can

---

\*Supported by DARPA contract #F30602-97-C-0326.

†Supported by Certicom and an NSF Graduate Research Fellowship.

one efficiently recover the entire key? There is no known method for doing so. Furthermore, the common belief is that no such efficient algorithm exists. This resistance to partial key exposure is an interesting distinction between RSA and discrete log schemes.

We note that Wiener [9] showed that RSA is insecure whenever the private exponent  $d$  is less than  $N^{1/4}$ . In other words, given that the 3/4 most significant bits of  $d$  are zero an adversary can efficiently recover the remaining quarter. This result does not apply to our problem: Wiener’s continued fractions approach does not work when the most significant bits of  $d$  are given to the adversary, but they are non-zero. Instead, we derive our results using powerful tools due to Coppersmith [1].

Let  $N = pq$  be an  $n$ -bit RSA modulus. Throughout the paper we view the private exponent  $d$  as an  $n$ -bit string. When referring to the  $t$  most significant bits of  $d$  we refer to the  $t$  leftmost bits of  $d$  when viewed as an  $n$ -bit string. For instance, it is possible that the  $t$  most significant bits of  $d$  are all zero, for some  $t$ . Similarly, a quarter of the bits of  $d$  always refers to  $n/4$  bits.

## 1.1 Summary of Results

We summarize our results in the following two theorems. The proofs are given in the body of the paper. The first theorem applies to low public exponent RSA. The second applies to larger values of  $e$ . Throughout we assume  $N = pq$  is an RSA modulus with  $\sqrt{N}/2 < q < p < 2\sqrt{N}$ .

**Theorem 1.1** *Let  $N = pq$  be an  $n$ -bit RSA modulus. Let  $1 \leq e, d \leq \phi(N)$  satisfy  $ed \equiv 1 \pmod{\phi(N)}$ . There is an algorithm that given the  $\frac{n}{4}$  least significant bits of  $d$  computes all of  $d$  in time polynomial in  $n$  and  $e$ .*

As we shall see, the running time of the attack algorithm in the above theorem is in fact linear in  $e \log_2 e$ . Consequently, as long as  $e$  is not “too large” the attack can be efficiently mounted. For a very small value of  $e$  such as  $e = 3$  the attack runs in a reasonable amount of time. For larger values, such as  $e = 65537$ , the attack is still feasible, though clearly takes much longer.

**Theorem 1.2** *Let  $N = pq$  be an  $n$ -bit RSA modulus. Let  $1 \leq e, d \leq \phi(N)$  satisfy  $ed \equiv 1 \pmod{\phi(N)}$ .*

1. *Suppose  $e$  is a prime in the range  $[2^t \dots 2^{t+1}]$  with  $\frac{n}{4} \leq t \leq \frac{n}{2}$ . Then given the  $t$  most significant bits of  $d$  there is a polynomial time (in  $n$ ) algorithm to compute all of  $d$ .*
2. *More generally, suppose  $e \in [2^t \dots 2^{t+1}]$  is the product of at most  $r$  distinct primes with  $\frac{n}{4} \leq t \leq \frac{n}{2}$ . Then given the factorization of  $e$  and the  $t$  most significant bits of  $d$  there is an algorithm to compute all of  $d$  in time polynomial in  $n$  and  $2^r$ .*
3. *When the factorization of  $e$  is unknown, we obtain a weaker result. Suppose  $e$  is in the range  $[2^t \dots 2^{t+1}]$  with  $t \in 0 \dots n/2$ . Further, suppose  $d > \epsilon N$  for some  $\epsilon > 0$ . Then there is a polynomial time (in  $n$  and  $\frac{1}{\epsilon}$ ) algorithm that given the  $n - t$  most significant bits of  $d$ , computes all of  $d$ .*

Theorem 1.2 applies to public exponents  $e$  in the range  $2^{n/4} \leq e \leq 2^{n/2}$ . Unlike the previous theorem, Theorem 1.2 makes use of the *most* significant bits of  $d$ . When  $e$  is prime, at most half

the bits of  $d$  are required to mount the attack. Fewer bits are needed when  $e$  is smaller. Indeed, if  $e$  is close to  $N^{1/4}$  only a quarter of the MSB bits of  $d$  are required. The same result holds when  $e$  is not prime, as long as we are given the factorization of  $e$  and  $e$  does not have too many distinct prime factors. The last part of the theorem applies to  $e < N^{1/2}$  when the factorization of  $e$  is not known. To mount the attack, at least half the MSB bits of  $d$  are required. More bits are necessary, the smaller  $e$  is. The attack algorithm works for most  $e$ , but may fail if  $d$  is significantly smaller than  $N$ .

One may refine Theorem 1.2 in many ways. It is possible to obtain other results along these lines for public exponents  $e < N^{1/2}$ . For instance, consider the case when the factorization of  $e$  is unknown. If the adversary is given half the most significant bits of  $d$  and a quarter of the least significant bits then we show the adversary can recover all of  $d$ . When  $e < N^{1/4}$  this is better than the results of Theorem 1.2 part (3). However, we view attacks that require a non-consecutive bits of  $d$  as artificial. We briefly sketch these variants in Section 4.3.

## 1.2 Notation

Throughout the paper we let  $N = pq$  denote an  $n$ -bit RSA modulus. We assume the primes  $p$  and  $q$  are distinct and close to  $\sqrt{N}$ . More precisely, we assume

$$4 < \sqrt{N}/2 < q < p < 2\sqrt{N} \tag{1}$$

We denote the set of such  $n$ -bit RSA moduli by  $\mathbb{Z}_{(2)}(n)$ . Our results also apply to RSA moduli  $N = pq$  where  $p$  is much larger than  $q$ , but we do not give the details here.

Notice that equation (1) implies  $p + q < 3\sqrt{N}$ . For convenience, throughout the paper we set

$$s := p + q.$$

Under the assumption  $p > q$  this implies:

$$p = \frac{1}{2}(s + \sqrt{s^2 - 4N}). \tag{2}$$

Furthermore, it follows by equation (1) that

$$N/2 < N - 4\sqrt{N} < \phi(N) < N. \tag{3}$$

Let  $1 \leq e, d \leq \phi(N)$  be encryption/decryption exponents. Then  $ed \equiv 1 \pmod{\phi(N)}$ . Throughout the paper we denote by  $k$  the unique integer such that:

$$ed - k\phi(N) = ed - k(N - s + 1) = 1. \tag{4}$$

Since  $\phi(N) > d$  we know that  $k < e$ .

## 2 Finding Small Solutions to Bivariate Polynomials

Our results make heavy use of seminal results due to Coppersmith. Using the lattice basis reduction algorithm of Lenstra, Lenstra, and Lovasz [5], Coppersmith [1] shows how to find small solutions  $(x_0, y_0)$  to a bivariate polynomial  $f(x, y)$ , provided appropriate bounds on  $x_0$  and  $y_0$  are known in advance.

**Theorem 2.1 (Coppersmith[1])** *Let  $f(x, y)$  be a polynomial in two variables over  $\mathbb{Z}$ , of maximum degree  $\delta$  in each variable separately, and assume the coefficients of  $f$  are relatively prime as a set. Let  $X, Y$  be bounds on the desired solutions  $x_0, y_0$ . Define  $\tilde{f}(x, y) := f(Xx, Yy)$  and let  $D$  be the absolute value of the largest coefficient of  $\tilde{f}$ . If  $XY < D^{2/(3\delta)}$ , then in time polynomial in  $(\log D, 2^\delta)$ , we can find all integer pairs  $(x_0, y_0)$  with  $p(x_0, y_0) = 0, |x_0| < X, |y_0| < Y$ .*

We make use of an immediate consequence of this theorem, which is a slight generalization of a result in [1].

**Corollary 2.2** *Let  $N = pq$  be an  $n$ -bit RSA modulus. Let  $r \geq 2^{n/4}$  be given and suppose  $p_0 := p \bmod r$  is known. Then it is possible to factor  $N$  in time polynomial in  $n$ .*

**Proof** Without loss of generality, assume  $p$  and  $r$  are relatively prime. From  $p_0 := p \bmod r$  we may find  $q_0 := q \equiv N/p_0 \bmod r$ . We seek a solution  $(x_0, y_0)$  to  $f(x, y) = (rx + p_0)(ry + q_0) - N$ , where  $0 \leq x_0 < X = 2^{n/2+1}/r$  (similarly  $y_0 < Y = 2^{n/2+1}/r$ ). Notice, however, that the greatest common divisor of the coefficients of the polynomial  $f(x, y)$  is  $r$ , so to use Theorem 2.1, we must divide through by  $r$  to get a new polynomial  $g(x, y) = f(x, y)/r$ . Now notice that the largest coefficient of  $\tilde{g}(x, y) = g(Xx, Yy)$  is at least  $2^{n+2}/r$ . So, to use Theorem 2.1 we require

$$XY = r^{-2}2^{n+2} < (2^{n+2}/r)^{2/3},$$

which is satisfied whenever  $r > 2^{(n+2)/4}$ . By doing exhaustive search on the first two bits of  $x_0$  and  $y_0$  this can be reduced to  $r \geq 2^{n/4}$ .  $\square$

## 3 Partial key exposure attack on low exponent RSA

In this section we consider attacks on the RSA cryptosystem with a “small” exponent  $e$ . For our purposes, “small” implies that exhaustive search on all values less than  $e$  is feasible. In particular, since  $k \leq e$  holds, our attack algorithm can try all possible values of  $k$  (recall that  $k$  is the unique integer satisfying  $de - k\phi(N) = 1$ ). We can now prove Theorem 1.1.

**Theorem 3.1** *With the notation as in Section 1.2, given the  $\frac{n}{4}$  least significant bits of  $d$ , we can factor  $N$  in time linear in  $e \log_2 e$  and polynomial in  $n$ .*

**Proof** Suppose we are given the least-significant  $\frac{n}{4}$ -bit block of  $d$ ; that is, we know  $d_0 \equiv d \bmod 2^{n/4}$ . Reducing equation (4) modulo  $2^{n/4}$  and substituting  $s = p + N/p$ , we see that  $p \bmod 2^{n/4}$  is a solution for  $x$  in the equation

$$ed_0 \equiv 1 + k(N - x - N/x + 1) \pmod{2^{n/4}},$$

which leads to the quadratic following equation in  $x$ :

$$kx^2 + (ed_0 - k(N + 1) - 1)x + kN \equiv 0 \pmod{2^{n/4}}. \quad (5)$$

The attack algorithm tries all candidate values for  $k$  in the range  $[0 \dots e]$ . For each candidate value, the algorithm computes every solution for  $x \pmod{2^{n/4}}$  in equation (5) and applies the algorithm of Corollary 2.2 to each solution to try to factor  $N$ .

It remains to show that equation (5) does not have too many solutions for each candidate value  $k$ . Notice that  $k$  divides the coefficients of the quadratic term and the constant term of equation (5). To see that  $k$  divides the coefficient of the linear term, observe  $ed_0 - k(N + 1) - 1 = -k(p + q)$ . Suppose  $k = 2^t m$  for some integer  $t$  and odd integer  $m$ . Then every solution  $x$  to equation (5) satisfies

$$mx^2 - m(p + q)x + mN \equiv 0 \pmod{2^{(n/4)-t}}.$$

Furthermore,  $m$  is odd so  $m^{-1}$  is well-defined, so every such solution satisfies

$$x^2 - (p + q)x + N \equiv 0 \pmod{2^{(n/4)-t}}.$$

This equation has at most two solutions modulo 2, so by Hensel lifting we have that it has at most two solutions modulo  $2^{(n/4)-t}$ . Therefore, equation (5) has at most  $2^{t+1}$  solutions modulo  $2^{n/4}$ . For example, there are at most two solutions for odd candidates  $k$ , four solutions when  $k = 2m$  for odd  $m$ , and so on; since  $k$  is exhaustively searched in the range  $[0 \dots e]$ , this implies that at most  $e \log_2 e$  solutions  $x$  will be tested before the correct value of  $x = p \pmod{2^{n/4}}$  is obtained, exposing the factorization of  $N$ . Hence, the total running time is linear in  $e \log_2 e$  and polynomial in  $n$ .  $\square$

We did not employ the full generality of Corollary 2.2, as mod  $r$  bits could have been used for any  $r \geq 2^{n/4}$ . This will be used in the next section where we consider more sophisticated key exposure attacks.

One may wonder whether a similar partial key exposure attack is possible using the most significant bits of  $d$ . The answer is no. The reason is that low public exponent RSA leaks half the most significant bits of  $d$ . In other words, the adversary may obtain *half* the most significant bits of  $d$  from  $e$  and  $N$  alone. Consequently, revealing the most significant bits of  $d$  does not help the adversary in exposing the rest of  $d$ . This is stated more precisely in the following fact.

**Fact 3.2** *With the notation as in Section 1.2, suppose there exists an algorithm  $\mathcal{A}$  that given the  $n/2$  MSB bits of  $d$  discovers all of  $d$  in time  $t(n)$ . Then there exists an algorithm  $\mathcal{B}$  that breaks RSA in time  $et(n)$ .*

**Proof** Observe that by equation (4), we have  $d = (1 + k(N + 1 - p - q))/e$ . Let  $\tilde{d}$  be

$$\tilde{d} = \left\lfloor \frac{1 + k(N + 1)}{e} \right\rfloor$$

Then

$$0 \leq \tilde{d} - d \leq k(p + q)/e \leq 3k\sqrt{N}/e < 3\sqrt{N}$$

It follows that  $\tilde{d}$  matches  $d$  on the  $n/2$  most significant bits of  $d$ . Hence, once  $k$  is known, the half most significant bits of  $d$  are exposed. With this observation, algorithm  $\mathcal{B}$  can work as follows: try all possible values of  $k$  in the range  $[0 \dots e]$ . For each candidate, compute the value  $\tilde{d}$ . Run algorithm  $\mathcal{A}$  giving it half the most significant bits of  $\tilde{d}$ . Once the correct  $k$  is found the entire private key is exposed.  $\square$

Fact 3.2 explains why for low exponent RSA one cannot mount a partial key recovery attack given the most significant bits of  $d$ . It is natural to ask whether one can expose all of  $d$  given a quarter of the low order bits of  $d$  that are not necessarily the least significant ones. For instance, P. Nguyen[6] observed that if  $p, q$  are randomly chosen  $n/2$ -bit primes, then the attack can be mounted given the  $n/4$  bits in positions  $n/4$  to  $n/2$ . Is it possible to generalize this result to *every* subsequence of  $n/4$  bits from the  $n/2$  low-order bits of  $d$ ? At the moment this is an open question.

**Remark 1.** Fact 3.2 demonstrates that computing the exponentiation associated with the half high-order bits of  $d$  can be performed by an untrusted server. This may be of use in server-aided RSA systems where techniques using the Chinese Remainder Theorem cannot be employed. For instance, in threshold RSA [3], the factorization of the modulus is not known to any party, so Chinese Remainder techniques are of no help in accelerating computation. Fact 3.2 suggests that the half high-order bits of  $d$  can be revealed to the server with no additional compromise in security, allowing accelerated computation by off-loading that portion of the exponentiation operation.

## 4 Partial key exposure attack on medium exponent RSA

We describe several attacks on the RSA system that can be employed when the public key  $e$  is in the range  $2^{n/4}$  to  $2^{n/2}$ . Unlike the previous section, these attacks require the *most* significant bits of  $d$  to be given. We mount the attack by carefully studying equation (4):

$$ed - k(N - s + 1) = 1$$

Recall that  $s = p + q$ .

The key to mounting these attacks is in finding  $k$ . Searching for  $k$  by brute force is infeasible, since  $k$  is an arbitrary element in the range  $[0, e]$ . Fortunately, given sufficiently many MSB's of  $d$ , we may compute  $k$  directly, eliminating it as an unknown from equation (4). Once  $k$  is revealed, we are left with two unknowns,  $d$  and  $s$  which we recover using various methods. The main tool for discovering  $k$  is presented in the following theorem. It shows that as long as  $e < \sqrt{N}$  we can find  $k$  given only  $\log_2 e$  MSB bits of  $d$ . The theorem produces a small constant size interval containing  $k$ . As always, we try all possible values of  $k$  in the interval until our attack algorithm succeeds.

**Theorem 4.1** *With the notation as in Section 1.2, let  $t$  be an integer in the range  $[0 \dots \frac{n}{2}]$ . Suppose  $2^t < e < 2^{t+1}$  and we know the  $t$  most significant bits of  $d$ . Then we can efficiently compute the unique  $k$  satisfying equation (4) up to a constant additive error.*

The proof of Theorem 4.1 relies on the following lemma, which provides general conditions under which  $k$  can be deduced by rounding.

**Lemma 4.2** *Suppose  $d_0$  is given such that the following two conditions hold:*

(i)  $|e(d - d_0)| < c_1 N$ , and

(ii)  $ed_0 < c_2 N^{3/2}$ .

*Then the unique  $k$  satisfying  $ed - k\phi(N) = 1$  is an integer in the range  $[\tilde{k} - \Delta, \tilde{k} + \Delta]$  where  $\tilde{k} = (ed_0 - 1)/N$  and  $\Delta = 2(8c_2 + 2c_1)$ .*

**Proof** Let  $\tilde{k} = (ed_0 - 1)/N$ . Then

$$\left| \tilde{k} - k \right| = \left| (ed_0 - 1) \left( \frac{1}{\phi(N)} - \frac{1}{N} \right) + \frac{e(d - d_0)}{\phi(N)} \right| < c_2 N^{3/2} \left( \frac{N - \phi(N)}{\phi(N)N} \right) + c_1 \frac{N}{\phi(N)}$$

Since  $N - \phi(N) < 4\sqrt{N}$  and  $\phi(N) > N/2$  it follows that

$$\left| \tilde{k} - k \right| < 8c_2 + 2c_1.$$

Consequently,  $k$  is an integer in the range  $[\tilde{k} - \Delta, \tilde{k} + \Delta]$  as required. □

We are now prepared to prove Theorem 4.1.

### Proof of Theorem 4.1

The  $t$  most significant bits of  $d$  enable us to construct an integer  $d_0$  satisfying  $|d - d_0| < 2^{n-t}$ . We use Lemma 4.2 to compute  $k$ . By the restriction on  $e$ , condition (i) is satisfied with  $c_1 = 2$ . Since  $d_0 < N$ , condition (ii) holds with  $c_2 = 2$ . Hence  $k$  is an integer in a known interval of width 40. □

## 4.1 Prime public key

We are now ready to prove part (1) of Theorem 1.2. Theorem 4.1 enables us to find  $k$ . Once  $k$  is found we reduce equation (4) modulo  $e$ . This removes  $d$  from the equation. We can then solve for  $s \pmod{e}$ . Given  $s \pmod{e}$  we are able to factor the modulus.

**Theorem 4.3** *With the notation of Section 1.2, let  $t$  be an integer in the range  $\frac{n}{4} \leq t \leq \frac{n}{2}$ . Suppose  $e$  is a prime in the range  $[2^t \dots 2^{t+1}]$ . Furthermore suppose we are given the  $t$  most significant bits of  $d$ . Then we can factor  $N$  in polynomial time.*

**Proof** The assumptions of the theorem satisfy the conditions of Theorem 4.1. Consequently,  $k$  is known to be an integer in a constant size range. We try all candidate values for  $k$ . For each one we do the following:

1. Compute  $s \equiv N + 1 - k^{-1} \pmod{e}$ . This is well-defined since  $\gcd(e, k) = 1$ .

2. Find  $p \bmod e$  by finding a root  $x_0$  of the quadratic

$$x^2 - sx + N = 0 \pmod{e}$$

This can be done efficiently (in probabilistic polynomial time) since  $e$  is prime. Indeed, if  $s = p + q \bmod e$  then  $x_0 = p \bmod e$ .

3. Use Corollary 2.2 to find  $p$  given  $p \bmod e$ . This is possible since  $e \geq 2^{n/4}$ .

Once the correct value of  $k$  is found (after a constant number of attempts) the factorization of  $N$  is exposed.  $\square$

A surprising consequence of this theorem is that, when  $e$  is prime and is roughly  $\cong 2^{n/4}$ , only the first  $\frac{n}{4}$  MSB's of  $d$  are needed to mount the attack. This attack is as strong as the one on low public exponent RSA. In any case, for prime  $e \in 2^{n/4}..2^{n/2}$  the first  $\frac{n}{2}$  most significant bits of  $d$  always suffice.

The proof shows that it is not necessary for  $e$  to be prime. As long as we can solve the quadratic in step (2) the proof can be made to work. In order to solve the quadratic we must be given the factorization of  $e$ . Unfortunately, modulo a composite, the quadratic may have many roots. We must try them all. If  $e$  has  $r$  distinct prime factors, there are  $2^r$  solutions to consider. As a result, we must also bound the number of prime factors of  $e$ . We obtain part (2) of Theorem 1.2.

**Corollary 4.4** *As in Theorem 4.3 suppose  $e$  is an integer in the range  $[2^t \dots 2^{t+1}]$ . If  $e$  has at most  $r$  distinct prime factors, and its factorization is known, then given the  $t$  most significant bits of  $d$  we can factor  $N$  in polynomial time.*

We point out that when  $e$  is close to  $2^{n/2}$  the same attack can be mounted even if the factorization of  $e$  is unknown. In other words, for all  $e$  sufficiently close to  $2^{n/2}$ , half the MSB's of  $d$  are sufficient to reconstruct all of  $d$ . Indeed, the range  $1..2^{n/2+2}/e$  can be searched exhaustively to find  $s/e$ . Given the value of  $s/e$  we can obtain  $s$  (since  $s \bmod e$  is already known.) Since  $s$  is now known in the integers we can directly find  $p$  using equation (2).

## 4.2 Public key with unknown factorization

We now turn to proving part (3) of Theorem 1.2. We consider the case when  $e$  is in the range  $[2^t \dots 2^{t+1}]$  with  $0 \leq t \leq \frac{n}{2}$ . The factorization of  $e$  is unknown. The following result establishes that we can still find all of  $d$ , given some of its MSB's. Our attack works as long as  $k$  is not significantly smaller than  $e$ . At the end of the section we note that the attack heuristically works for almost all  $e$  in the range  $[2^t, 2^{t+1}]$ .

**Theorem 4.5** *With the notation as in Section 1.2, let  $t$  be an integer in the range  $[0 \dots n/2]$ . Suppose  $e$  is in the range  $[2^t \dots 2^{t+1}]$ . Further suppose  $k > \epsilon \cdot e$  for some  $\epsilon > 0$ . Then there is an algorithm that given the  $n - t$  most significant bits of  $d$  finds all of  $d$ . The algorithm runs in time  $O(n^3/\epsilon)$ .*

**Proof** Given the  $n-t$  most significant bits of  $d$  we can construct a  $d_0$  such that  $0 \leq d-d_0 < 2^t$ . Since  $e < 2^{n/2}$  we can use  $d_0$  and Theorem 4.1 to limit  $k$  to a constant size interval. For each of the candidate  $k$  we do the following:

1. Compute  $d_1 = e^{-1} \bmod k$ . This is possible since  $e$  and  $k$  are relatively prime. Since  $ed - k\phi(n) = 1$  we know that  $d_1 = d \bmod k$ .
2. By assumption  $k > \epsilon 2^t$ . Note that at this point we know  $d \bmod k$  as well as the  $n-t$  MSB's of  $d$ . We determine the rest of the bits by an exhaustive search. More precisely, write

$$d = kd_2 + d_1$$

Then  $d_2 = d_0/k + (d-d_0)/k - d_1/k$ . The only unknown term in this sum is  $v = (d-d_0)/k$ . Since  $k > \epsilon 2^t$  we know that  $v = (d-d_0)/k < 1/\epsilon$ . To find  $v$  we try all possible candidates in the range  $[0, \frac{1}{\epsilon}]$ . For each candidate we compute the candidate value of  $d$  and test it out.

3. Once the correct values of  $v$  and  $k$  are found  $d$  is exposed. Testing each candidate  $d$  takes  $O(n^3)$  time and there are  $O(1/\epsilon)$  candidates to try out.

□

Theorem 4.5 works without having to know the factorization of  $e$ . Unfortunately, the results are not as strong as in the previous section. When  $e$  is close to  $N^{1/4}$  Theorem 4.5 implies that 3/4 of the bits of  $d$  are needed to reconstruct  $d$ . This is much worse than the corresponding bound achieved in the previous section, where only 1/4 the bits were required. When  $e$  is close to  $N^{1/2}$  the theorem produces results similar to the previous section.

Theorem 4.5 can only be applied when  $k > \epsilon \cdot e$ . Intuitively,  $k$  behaves roughly as a random integer in the range  $[1, e]$ . As such, we should have  $k > e/10$  for about 90% of the  $e \in [2^t, 2^{t+1}]$ . Hence, heuristically the attack works efficiently for almost all  $e$ .

### 4.3 More Results

What if the factorization of  $e$  is unknown and  $e$  was not randomly chosen? Although it may be computationally infeasible, it is possible for  $e, d$  to be specifically chosen as factors of  $1+k\phi(N)$  for very small  $k$ , violating the conditions of Theorem 4.5. We stress that this is particularly unlikely, as not only would the rather large value of  $1+k\phi(N)$  would need to be factored into  $ed$ , but a factor  $e$  in the range  $[2^{n/4} \dots 2^{n/2}]$  would need to be obtained, and one that itself cannot easily be factored (making it vulnerable to Corollary 4.4). However, under these circumstances, the above attacks would not apply. We conclude with the following general result which holds for all  $e < 2^{n/2}$ . Unfortunately, the result requires non-consecutive bits of  $d$ .

**Theorem 4.6** *With the notation as in Section 1.2, let  $t$  be an integer in  $[1, \frac{n}{2}]$  and  $e$  in  $[2^t \dots 2^{t+1}]$ . Suppose we are given the  $t$  most significant bits of  $d$  and the  $\frac{n}{4}$  least significant bits of  $d$ . Then in polynomial time we can factor  $N$ .*

**Proof Sketch** Using Theorem 4.1 we may compute a constant size interval  $I$  containing  $k$ . Observe that the proof of Theorem 3.1 applies for all  $e$ , as long as  $k$  and the  $n/4$  least significant bits of  $d$  are known. To recover  $d$ , run the algorithm of Theorem 3.1 on all candidate values of  $k$  in  $I$ .  $\square$

In fact, Theorem 3.1 can be viewed as a special case of Theorem 4.6 in which exhaustive search is performed on the requisite  $O(\log n)$  MSB bits of  $d$ .

## 5 Conclusions

We study RSA's vulnerability to partial key exposure. We showed that for low exponent RSA, a quarter of the least significant bits of  $d$  are sufficient for efficiently reconstructing all of  $d$ . We obtain similar results for larger values of  $e$  as long as  $e < \sqrt{N}$ . For instance, when  $e$  is close to  $\sqrt{N}$ , half the most significant bits of  $d$  suffice.

The results presented demonstrate the danger of leaking a fraction of the bits of  $d$ . We note that discrete log schemes (e.g. DSS, ElGamal) do not seem vulnerable to partial key exposure. A fraction of the bits of the private key in a discrete log based system does not seem to enable the adversary to efficiently break the system.

There are a number of related open problems:

- Does there exist a polynomial time algorithm which enables one to break the RSA system for values of  $e$  substantially larger than  $\sqrt{N}$ , given only a subset of the bits of  $d$ ?
- Our strongest result with respect to the fewest bits of  $d$  required to break the system is for an  $e$  in the range  $N^{1/4} \dots N^{1/2}$  with known factorization. For an  $e$  with unknown factorization and in the same range, does there exist a polynomial time algorithm which provides as strong a result?

## References

- [1] D. Coppersmith, "Finding a small root of a univariate modular equation", Proc. of Eurocrypt '96, pp. 155–165.
- [2] T. ElGamal, "A public key cryptosystem and a signature scheme based on the discrete logarithm", IEEE Transactions on Information Theory, 31(4):469–472, 1985.
- [3] Y. Frankel, "A practical protocol for large group oriented networks", Proc. of Eurocrypt '89, pp. 56–61.
- [4] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems", Proc. of Crypto 96, pp. 104–113.

- [5] A. K. Lenstra, H. W. Lenstra, L. Lovász, “Factoring Polynomials with Rational Coefficients”, *Mathematische Annalen*, vol. 261, no. 4, 1982, pp. 515-534.
- [6] P. Nguyen, private communications.
- [7] J. F. Dhem, F. Koeune, P. A. Leroux, P. Mestré, J. J. Quisquater, and J. L. Willems. “A practical implementation of the timing attack”, In *Proc. of CARDIS 98 – Third smart card research and advanced application conference*, UCL, Louvain-la-Neuve, Belgium, Sep. 14-16, 1998.
- [8] R. L. Rivest, A. Shamir, and L. Adleman. “A method for obtaining digital signatures and public-key cryptosystems”, *Communications of the ACM* 21(2):120-126, Feb. 1978.
- [9] M. Wiener, “Cryptanalysis of short RSA secret exponents”, *IEEE Transactions on Info. Th.*, Vol. 36, No. 3, 1990, pp. 553–558.