Tunable balancing of RSA

S. D. Galbraith, C. Heneghan and J. F. McKee*

Department of Mathematics,
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK.
[Steven.Galbraith,C.Heneghan,James.McKee]@rhul.ac.uk

Abstract. We propose a key generation method for RSA moduli which allows the cost of the public operations (encryption/verifying) and the private operations (decryption/signing) to be balanced according to the application requirements. Our method is a generalisation of using small public exponents and small Chinese remainder (CRT) private exponents. Our results are most relevant in the case where the cost of private operations must be optimised. We give methods for which the cost of private operations is the same as the previous fastest methods, but where the public operations are significantly faster. For example, the fastest known (1024 bit) RSA decryption is using small CRT private exponents and moduli which are a product of three primes. In this case we equal the fastest known decryption time and also make the encryption time around 4 times faster.
The paper gives an analysis of the security of keys generated by our method, and several new attacks. The ingredients of our analysis include several ideas of Coppersmith and a new technique which exploits linearisation. We also present a new birthday attack on low Hamming-weight private exponents.

1 Introduction

In many implementations of the RSA cryptosystem the public exponent e is chosen to be very small. Encryption and signature verification then use very few operations modulo N. On the other hand, decryption and signature generation, even if they are performed using the Chinese remainder theorem (CRT), cost much more than encryption/verifying.

This imbalance can be inconvenient in some situations. One example is when a device with limited computational power (such as a smart card) is required to generate RSA signatures. A related issue for such devices is the space requirement for storing private keys, which we would like to minimise. Another example is when a server is required to handle a large number of decryptions of messages from numerous clients and so its computational burden should be minimised. A related issue in this case is to ensure that such a server is not overly vulnerable

* All three authors were supported in their research by a grant from the MathFIT programme, jointly sponsored by the EPSRC and the LMS.

to denial-of-service attacks. Hence there has been much interest in speeding up the private operations in RSA. Many of the previous solutions have achieved this at the cost of a significant loss of performance of the public operations. Our goal is to have fast private operations without paying such a high price for the public operations.

In some applications it might be desired that the public and private operations can be performed with the same computational effort. For example, this may be the case in protocols where two parties are required to act synchronously without idle time, or in systems where, to ensure fairness (i.e., that no party is at an advantage to others), the computational burden of all parties should be equal.

One early proposal to speed up the private operations was to choose the private exponent d to be small (note that e then becomes large). This was cryptanalysed by Wiener [25] who showed that the scheme is insecure if $d < N^{0.25}$. These results were extended by Boneh and Durfee [1] to $d < N^{0.292}$.

Wiener proposed two countermeasures to the above attacks. The first is to increase the size of the public exponent e, which causes a severe penalty on the public operations. The second variant is to use a private exponent d which is not itself small, but which reduces to small values when performing decryption/signing using the CRT. We call these "small CRT private exponents". There is a birthday attack (see [16]), which means that the CRT private exponents must have at least 160 bits. Apart from this attack, the security of such variants has been analysed by May [14] and the only serious attack known is in the unbalanced case (i.e., where the modulus is a product of primes of differing size).

One drawback of using small CRT private exponents is that the public exponent e is large and then the cost of encryption/verification is very expensive (much worse than original cost of decryption/signing).

In this paper we propose a combination of 'small' public exponents e and 'small' CRT private exponents. Of course, they cannot both be very small. Our approach allows the designer to choose the parameters in a suitable way to get a trade-off between the costs of encryption/verification and decryption/signing.

A proposal to balance public and private exponents in RSA was given by Sun, Yang and Laih [21] (improved in [20], to resist an attack in [8]).

These results do not utilise CRT private exponents and so their solution is not competitive with ours.

Lim and Lee [13] gave methods for generating RSA private exponents with relatively low Hamming weight. Some of their methods do not use the Chinese remainder theorem, and so are slow. Section 5 of [13] does use CRT private exponents, but there seems to be no suggestion that the exponents may be taken to be short. Hence the cost of private operations in [13] is much slower than we achieve.

Other methods which allow faster private operations are multiprime RSA (products of more than two primes, see [4], [2] and [15]) and the Takagi system [22] (which uses moduli of the form $p^k q$). The use of 3-prime moduli with small CRT private exponents is the fastest method (for 1024-bit moduli) for RSA

decryption/signing, but the public operations are slow since the public exponent then has 1024 bits. The Takagi system is particularly appealing as it requires small public exponents (otherwise the Hensel lifting is slow) and so both public and private operations are fast. Note that it is impossible to 'equalise' encryption and decryption times using Takagi.

The Rabin cryptosystem [17] is based on squaring modulo N. The easiest case is when all primes dividing N are congruent to 3 modulo 4, in which case the decryption time is comparable to standard RSA decryption using the CRT, which in the present context is considered to be slow.

Our results are most relevant in the case where the cost of private operations must be optimised. We give methods where the cost of private operations is the same as (or almost as good as) the previous fastest methods, but where the public operations are significantly faster. For example, the fastest known (1024 bit) RSA decryption is using small CRT decryption exponents and moduli which are a product of three primes. In this case we match the fastest known decryption time and also make the encryption time around four times faster. For a comparison with previous systems see the timings given in section 11.

Some applications may require the cost of public and private operations to be equal. Our method gives a solution to this problem which is roughly twice as fast as previously known solutions.

We give a thorough analysis of the security of keys generated by our method, including several new attacks. Some of our attacks use ideas of Coppersmith [5, 6]. Another of our attacks exploits linearisation. We also develop a new birthday attack on RSA private exponents of low Hamming weight.

It is clear that our parameter choices are carefully specified to provide optimum performance and are not suitable for all applications. For example, our schemes would be inappropriate in situations where partial knowledge of private keys is likely to be available.

Independently, a related but less general scheme has been proposed in [19, 26].

2 The key generation method

Let $N = p_1 p_2$. Then CRT private exponents are integers d_1, d_2 such that

$$ed_i \equiv 1 \pmod{(p_i - 1)} \quad \text{for} \quad i = 1, 2.$$

Such an equation may be written as

$$ed_i = 1 + k_i(p_i - 1) = k_i p_i - (k_i - 1). \tag{1}$$

Clearly $ed_i > p_i - 1$ and so e and d_i cannot both be very small. But we would like to have e and d_i so that $\log_2(e) + \log_2(d_i) \approx \log_2(p_i)$ by imposing the condition that k_i be small. The case $k_i = 1$ is uninteresting, but $k_i = 2$ is possible, as are large values of k_i.

The method we propose in this section allows us to choose the sizes of e and the d_i and then construct primes p_i so that equation (1) will be satisfied.

The modulus is therefore generated by a special algorithm which depends on parameters which are likely to be known to an adversary. We will analyse the security of moduli arising from this key generation algorithm in later sections.

Choose parameters n_N, n_p, n_e, n_d, n_k which will be the bit-lengths of N, p_i, e, d_i and the k_i respectively ($i = 1, 2$). For example, one typical case would be

$$n_N = 1024, n_p = 512, n_e = 160, n_d = 354, n_k = 2$$

while another would be

$$n_N = 1024, n_p = 512, n_e = 512, n_d = n_k = 269$$

(we must have $n_e + n_d - n_k = n_p$).

Key generation algorithm:

- Input: n_e, n_d, n_k.
- Choose an odd n_e-bit integer e (one may wish to choose e to have low Hamming weight or to be prime).
- For $i = 1, 2$: choose random n_k-bit integers k_i coprime to e and odd n_d-bit integers d_i satisfying the congruence

$$d_i \equiv e^{-1} \pmod{k_i}. \tag{2}$$

  until $p_i = 1 + (ed_i - 1)/k_i$ is prime.
- Output: (p_1, p_2, d_1, d_2).

Note that the primes p_i are roughly $n_e + n_d - n_k$ bits long. The public key ($N = p_1 p_2, e$) has an n_e-bit value for e and we have n_d-bit CRT private exponents, clearly giving us the opportunity to strike a balance between the costs of encryption/verification and decryption/signing.

Notes:

- Clearly, the key generation algorithm runs in random polynomial time. For fixed e we expect to need to choose roughly $n_p/2$ values for k_1 and d_1 until the corresponding odd number p_1 is prime. Doing the same for p_2 leads to around n_p primality tests in total. Hence, the cost of key generation is comparable to usual RSA systems.
- Due to equation (2) we usually assume $n_d \geq n_k$. It is possible to develop more general key generation methods but they have much worse performance.
- In some settings we may also want to choose the d_i to have low Hamming weight. This is easily done if the k_i are small. Security in this case is discussed in section 6 below.
- An analogous algorithm can be used for moduli of the form $p^k q$, pqr etc.

3 Security

The key generation method produces special moduli, and so the security of the resulting public keys must be analysed. We are concerned with attacks which enable an adversary to obtain either of the private keys d_1 and d_2 of the system (equivalently, the factorisation of the modulus). We consider later the case of moduli which are a product of more two primes.

As is already well known, there is a birthday attack on the individual CRT private exponents (see [16] for some details). Hence, we always require that $n_d \geq 160$.

The security analysis falls naturally into two cases. The first is where the values k_1 and k_2 are known to the attacker (for example, they may take especially small values, such as $k_1 = k_2 = 2$). The security in this case is addressed in section 4. The second case is where the values k_1 and k_2 are private (in which case the entire security may depend on whether or not it is possible to calculate one or both of them). This is studied in section 5.

4 Known k_i

Suppose that the k_i are known (e.g., because they are small and can be guessed, or because they have been obtained from some computation).

The first attack is to note that equation (1) implies that

$$p_i \equiv k_i^{-1}(k_i - 1) \pmod{e}\,.$$

Hence we are in a strong position to apply results of Coppersmith on factoring with partial knowledge of a factor. The original results of Coppersmith [5] were phrased in terms of most or least significant bits of p_1 whereas we have information modulo e. The presentation in section 5 of [6] is more general than [5], and from results stated there it is easy to deduce the following result.

Theorem 1. *Let $N = pq$ where $p, q \approx N^{1/2}$. Suppose $e > N^{1/4}$. Then the factorisation of N can be obtained in polynomial time if $p_0 \equiv p \pmod{e}$ is known.*

This technique will split N for us if $n_e > n_N/4$. This attack can be extended to larger ranges by combining it with exhaustive search, Namely, suppose $p_i \equiv a \pmod{e}$ and so $p_i = a + ex$ for some unknown x. We can write $x = x_0 + 2^m y$ and search over all $0 \leq x_0 < 2^m$, trying the attack for each guess (actually, since e and p_i are odd we only need try those $x_0 \equiv a + 1 \pmod{2}$). This extends the range of the attack to cases where $n_e + m \geq n_N/4$, but multiplies the complexity by 2^m. For security we impose the condition

$$n_e \leq n_N/4 - m\,, \tag{3}$$

where m is a security parameter. We shall take $m = 80$ for 1024-bit moduli. This choice of m is fairly conservative; since the running time of Coppersmith's

method for 1024-bit moduli is much more than one bit operation. One might be tempted to use a smaller value for m, but note that attack variant 3 below is rather more efficient than the above attack, so m cannot be taken to be much smaller than 80. The sample parameters above in the case $k_i = 2$ resist this attack.

There are several alternative attacks: see [10] for details. Briefly, the key ideas are as follows.

1. It is easy to see that d_1 is a small solution to the equation $ex + (k_1 - 1) \equiv 0 \pmod{p_1}$. One can thus apply the standard lattice methods to obtain d_1. The value d_1 can potentially be recovered by such methods if

$$d_1 < N^{1/4}.$$

   This attack is therefore a lattice-based 'small CRT private exponents' attack. We comment that the knowledge of k_1 is essential to make this attack possible; it is an open problem to find a small CRT private exponents attack for standard RSA (though see May [14]).
   The security condition (3) combined with the equation $n_e + n_d = n_p + n_k$ implies that $n_d > n_N/4 + m$ and so the attack is avoided.
   Interestingly, as pointed out by Ellen Jochemsz, this attack can be improved if k_1 is larger. Recall that the equation is

$$ed_1 + (k_1 - 1) = k_1 p_1.$$

   Writing $\mathcal{M} = k_1 p_1$ and $\mathcal{N} = k_1 N$ we have a polynomial $f(x) = ex + (k_1 - 1)$ with a small root modulo $\mathcal{M}$ and where $\mathcal{M} \mid \mathcal{N}$ for a known $\mathcal{N}$. If $k_1 = N^{\epsilon}$ then $\mathcal{N} = N^{1+\epsilon}$ and so $\mathcal{M} = \mathcal{N}^{(1/2+\epsilon)/(1+\epsilon)}$. Standard results (for example, page 28 of [6]) show that x can be recovered if

$$x < \mathcal{N}^{((1/2+\epsilon)/(1+\epsilon))^2} = N^{\frac{1}{4}+\frac{3}{4}\epsilon+\frac{\epsilon^2}{(1+\epsilon)}}.$$

   So, in general, the attack becomes more powerful.
   For our system, condition (3) must always be satisfied. Hence we require $n_e < n_N/4 - m$ and so $n_d > n_N/4 + m + n_k$. For the attack to work we need (ignoring the term $\epsilon^2/(1+\epsilon)$, which is acceptable for the parameters of interest) $n_d < n_N(1/4 + 3\epsilon/4) = n_N/4 + 3n_k/4$. So for our system, as n_k increases, the attack becomes less powerful.
2. The equation

$$e^2 d_1 d_2 = (k_1 p_1 - (k_1 - 1))(k_2 p_2 - (k_2 - 1)) = k_1 k_2 N - \epsilon$$

   (where $\log_2(\epsilon) \approx 2n_k + n_p$) implies that the product $d_1 d_2 \approx k_1 k_2 N/e^2$ if n_e is sufficiently large. For example, if $e^2 > \epsilon$ then given N, e, k_1 and k_2 one immediately obtains $d_1 d_2$ which would be relatively easily factored.
3. If $k_1 = k_2$ then, by the equation used above,

$$p_1 + p_2 \equiv (k_1(k_1 - 1))^{-1}(k_1^2 N + (k_1 - 1)^2) \pmod{e^2}.$$

Once again, if $e^2 > 2\sqrt{N}$ then this determines $p_1 + p_2$ and the factorisation is easily obtained. More generally, we can express the problem as a small discrete logarithm search on (p_1+p_2) (via the fact $a^{p_1+p_2} \equiv a^{N+1} \pmod N$) using knowledge of $(p_1 + p_2) \pmod{e^2}$. Hence the problem can be solved in time $O(N^{1/4}/e)$.

Note that the running time of this attack if $n_e = n_N/4 - m$ is $O(2^m)$ multiplications modulo N, so this attack is faster than the one above which uses Coppersmith's method.

4. If k_1 is known then $d_1 \equiv e^{-1} \pmod{k_1}$ which reduces the uncertainty about d_1. Hence, if k_i is known then we should impose $n_d \geq 160 + \log_2(k_i)$. The case $k_i = 2$ simply states that the d_i are odd, which is nothing new.

Condition (3) guards against all of these attacks.

5 Unknown k_i

We now turn to the case where the k_i are private. We will give methods to find the k_i, from which it is usually immediate to recover the full private key using the methods of the previous section. Indeed for this section we imagine that the parameters are such that knowledge of the k_i would mean that the private key can be obtained using the methods of the previous section.

One can perform an exhaustive search on, say, k_1 and (if the parameters are suitable) apply the methods of section 4 to check each guess. Hence, we impose the condition $n_k \geq m$ for security.

We have investigated the possibility of a direct birthday attack on the individual k_i. Our attempts in this direction have been unsuccessful (although there is a closely related birthday attack on the sum of the k_i which we present in subsection 5.2). Hence, we are led to believe that there is no direct birthday attack in this case. Part of the obstruction seems to be the combination of additive and multiplicative roles played by the k_i. In any case, as we will see in the next subsection, in the two prime case we must take $n_k > 2m$. The 3-prime case will be analysed in section 9.

5.1 Linearisation attack

An obvious attack is to use information available by taking equation (1) modulo e. We have (for $i = 1, 2$)

$$k_i p_i \equiv (k_i - 1) \pmod e .$$

Multiplying these two equations together we see that $(x, y) = (k_1, k_2)$ is a solution to the multivariate equation

$$xy(N-1) + x + y - 1 \equiv 0 \pmod e. \tag{4}$$

For this subsection we suppose that $2n_k \leq n_e$ which implies that (k_1, k_2) is a relatively small solution to equation (4). We linearise by defining $u = xy$ and $v = x + y - 1$ and define $0 < A < e$ by $A = (1 - N) \pmod{e}$. We have

$$uA \equiv v \pmod{e} \tag{5}$$

where u and v have bounded size. Candidate (u, v) pairs can be found in polynomial time using the continued fraction method as long as $uv < e$ (the details are standard, and are given below in a more general setting).

Hence, to resist the attack we are required to impose the restriction

$$n_k \geq n_e/3.$$

We now give details of how the algorithm can be extended to the case when $3n_k$ is only slightly larger than n_e by adding an exhaustive-search. A similar problem was investigated by Wiener [25] and Verheul and van Tilborg [24], who both give extensions to the continued fraction method.

First we recall a famous result on good rational approximations.

Theorem 2. *(see e.g., Theorem 184 of [9]). Suppose that* $\gcd(a, b) = \gcd(c, d) = 1$ *and that*

$$\left|\frac{a}{b} - \frac{c}{d}\right| \leq \frac{1}{2d^2}.$$

Then c/d is one of the convergents of the continued fraction expansion of a/b.

The congruence $Au \equiv v \pmod{e}$ means that there is some integer l such that

$$Au = v + le\,. \tag{6}$$

In our situation, we know that v has about $n_k + 1$ bits, and u has about $2n_k$ bits, but we may as well handle the most general situation. We suppose that there are parameters r and s bounding the sizes of u and v:

$$0 < v \leq r \leq e\,, \qquad 0 < u \leq s \leq e\,.$$

(The analysis where u and/or v is allowed to be negative is similar.)

We also suppose that $0 < A < e$, with $\gcd(A, e) = 1$. We then have $l \geq 0$. Dividing equation (6) by ue and adding $j/2s^2$ to each side gives

$$\frac{A}{e} + \frac{j}{2s^2} = \frac{v}{eu} + \frac{j}{2s^2} + \frac{l}{u}\,.$$

Now we apply theorem 2 to see that l/u is a continued fraction convergent to $A/e + j/2s^2$ if

$$\left|\frac{v}{eu} + \frac{j}{2s^2}\right| \leq \frac{1}{2u^2}\,,$$

and this will certainly hold if j is the nearest integer to $-2vs^2/eu$, since $1/2s^2 \leq 1/2u^2$.

If u lies in the range $s/2 < u \leq s$, then we have $-4rs/e \leq j \leq 0$. To find smaller values of u, we repeat with the ranges

$$s/4 < u \leq s/2,\ s/8 < u \leq s/4,\ \ldots$$

(i.e., substitute $s/2^t$ for s everywhere).

We have established the following theorem.

Theorem 3. *Consider the congruence* $Au \equiv v \pmod{e}$*, where* $0 < A < e$ *and* $\gcd(A, e) = 1$*. All solutions* (u, v) *satisfying*

$$0 < v \leq r \leq e\,, \qquad 0 < u \leq s \leq e$$

have that u *is a multiple of a denominator of a continued fraction convergent for one of*

$$A/e + 4^t j/2s^2\,, \qquad t \leq \log_2 s\,, \qquad j \leq 0\,, \qquad |j| \leq 4rs/e2^t\,.$$

In particular, all solutions with $\gcd(u, v) = 1$ *can be found in time* $O(\lceil rs/e \rceil (\log e)^2)$*. Any solutions with* $\gcd(u, v) > 1$ *can be obtained from those with* $\gcd(u, v) = 1$ *by scaling.*

Naively, the expected number of solutions to $Au \equiv v \pmod{e}$ with $0 < v \leq r$ and $0 < u \leq s$ is rs/e (but this does depend on A and e: for extreme cases, consider $A = 1$, or let e be the product of all primes up to r). Hence Theorem 3 is essentially optimal. A qualitatively similar result was obtained by Dujella [7] by other means.

In our case, $r \approx 2^{n_k+1}$ and $s \approx 2^{2n_k}$. We can restrict to $t = 0$ in the above. Therefore, we obtain the following result.

Theorem 4. *Let* $N = p_1 p_2$ *be produced by the key generation method with parameters* (n_e, n_k)*. If* $n_e + m \geq 3n_k + 2$ *then with the computation of continued fraction approximations to* $O(2^m)$ *rational numbers near* A/e*, we will find the pair* $(u, v) = (k_1 k_2, k_1 + k_2 - 1)$*, up to scaling.*

Since it is improbable that $k_1 k_2$ and $k_1 + k_2 - 1$ will have a large common factor, scaling is not a significant issue. Hence, to keep the k_i secret, we impose the condition

$$3n_k \geq n_e + m\,. \tag{7}$$

For our suggested parameters in the case of k_i private we have $3n_k = n_e + m$, and so the attack is avoided.

Note that the algorithm yields $u = k_1 k_2$ which is already enough for attack variant 2 in Section 4. If the actual values k_1 and k_2 are required then they can be recovered as solutions to the quadratic $t^2 - (v + 1)t + u = 0$. For parameters of interest, the value of e will be sufficiently large that the methods of Section 4 immediately recover the private key.

5.2 Further attacks

As in section 4 we have presented the most successful attack on our scheme. We briefly discuss a number of other approaches to the problem, some of which are reformulations or special cases of the attack described in section 5.1. Full details will appear in [10]. In all cases, the parameter restrictions are either already implied by condition (7) or require n_e to be larger than is of interest. For any lattice-based attack, one needs to allow for a brute-force extension: one could try 2^m lattices if any one of them can be checked quickly.

1. One can perform a birthday attack on the variable v in equation (5). Note that $0 < v < 2^{1+n_k}$.
   The idea of the attack is as follows. Let $M = \lfloor 2^{(1+n_k)/2} \rfloor$. We write $v = v_0 + v_1 M$ with $0 \leq v_i < M$. Then the statement that $u \equiv A^{-1}v \pmod{e}$ is small (as before, we assume $2n_k < n_e$) is equivalent to $A^{-1}v_0 \pmod{e}$ and $-A^{-1}Mv_1 \pmod{e}$ being close.
   The algorithm is then as follows. Compute and store a sorted list of values $A^{-1}v_0 \pmod{e}$ for $0 \leq v_0 < M$. Then compute the values $-A^{-1}Mv_1 \pmod{e}$ for $v_1 = 0, 1, 2, \ldots$ and try to find a close match on the top bits in the list. look for agreement with an entry in the list in the top $n_e - 2n_k$ bits. One expects about $2^{3n_k - n_e}$ candidates for (u, v) pairs to be found, all of which must be checked. The attack requires work $\tilde{O}(\max(2^{n_k/2}, 2^{3n_k - n_e}))$, and storage $\tilde{O}(2^{n_k/2})$.
   For security we require $\max(n_k/2, 3n_k - n_e) > m$. This is implied by condition (7). Indeed we saw that Theorem 3 is essentially best-possible for solving the congruence $Au \equiv v \pmod{e}$.
2. One can attempt to find a small solution to the multivariate equation (4) using lattice-based methods, following the methods of Coppersmith [5, 6] and Boneh and Durfee [1]. The details are standard and one can check that the method yields the same asymptotic security requirement as the linearisation/continued fraction method we have presented. In practice the lattice approach would be more complicated and would not be competitive with the attack presented above.
3. Another lattice attack was suggested to us by an anonymous referee. Multiplying together the key equations $ed_i - 1 = k_i(p_i - 1)$ we observe that the polynomial
   $$f(x, y, z) = 1 - ex - (N + 1)y + yz$$
   has the "small" solution $(x, y, z) = (d_1 + d_2, k_1 k_2, p_1 + p_2)$ modulo e^2. One can then attempt a trivariate lattice attack.
   This attack is thwarted by (7) unless $n_e > 0.65 n_N$, which is well outside the space of interesting parameters.
4. Another lattice-based attack arises from reducing equation (1) modulo p_1. In other words $(x, y) = (d_1, k_1 - 1)$ is a very small root of the polynomial
   $$f(x, y) = ex + y$$
   modulo p_1. Again, this attack is not effective for parameters satisfying condition (7).

5. Another attack is to note that equation (1) implies that the polynomial $f(x,y) = xy - 1$ has the solution $(x,y) = (k_1, p_1 - 1)$ modulo e. For parameters of interest, $n_e < n_p$ and so this attack has no chance of succeeding.

6 Low Hamming weight private CRT exponents

In some situations it is useful to choose the exponents d_i to have low Hamming weight. We now discuss the security requirements in this setting.

Suppose that the CRT private exponents d_i are chosen to have low Hamming weight w. We have $ed_1 \equiv 1 \pmod{p_1 - 1}$ and so

$$g^{ed_1} \equiv g \pmod{p_1}$$

for any g such that $\gcd(g, p_1) = 1$. In other words, our goal is to solve the discrete logarithm problem of g to base g^e modulo p_1 where the discrete logarithm is known to have low Hamming weight.

We mimic a randomised algorithm of Coppersmith (see Algorithm 3 of Stinson [18]) for discrete logarithms modulo a known prime p. If the bit-length of the exponent is m (in our application we will have $m = n_d$) and the Hamming weight is w then this algorithm has complexity $\tilde{O}(\sqrt{w}\binom{m/2}{w/2})$.

The idea is to guess a partition (B, B') of $\{0, 1, \ldots, m-1\}$ (i.e., $B \cap B' = \phi$ and $B \cup B' = \{0, 1, \ldots, m-1\}$) where $\#B_1 \approx \#B_2 \approx m/2$. We interpret (B, B') as a partition of the bit positions of m-bit integers. Then d_1 can be decomposed as a sum $d_1 = x_1 + x_2$ where x_i is a binary number with non-zero bits only in positions corresponding to numbers in B_i. We then hope that each x_i has Hamming weight $\approx w/2$.

If a suitable partition has been guessed then the number of choices for each x_i is approximately equal to the value of the binomial coefficient

$$M = \binom{n_d/2}{w/2}.$$

Hence, one has

$$g^{ex_1 + ex_2 - 1} \equiv 1 \pmod{p_1}.$$

The strategy is to perform a birthday attack on the problem using the time/memory tradeoff. Note that we are not seeking an equality, but a pair

$$u = g^{ex_1 - 1} \pmod{N} \text{ and } v = g^{ex_2} \pmod{N}$$

such that $\gcd(uv - 1, N)$ is non-trivial. Hence we use the FFT approach employed in [16]. This is done by first computing all the terms $u_j \equiv g^{ex_1 - 1} \pmod{N}$ as x_1 runs over integers whose binary representation uses bit positions only from the set B_1 and has Hamming weight approximately $w/2$. One then forms the polynomial

$$G(x) = \prod_{j=1}^{M} (u_j x - 1) \pmod{N}.$$

This polynomial has degree M and requires $O(M(\log N)^2)$ steps to construct. The polynomial $G(x)$ requires $M(\log N)$ storage.

Now, compute $v = g^e \pmod N$. We wish to evaluate $G(v^{x_2}) \pmod N$ for each of the candidate values for x_2 (to obtain a list of M values). This can be performed using an FFT-based algorithm (see Theorem 4 of [23]) in time $O(M(\log M)^2(\log\log M)(\log N)^2)$. We then traverse the list of all values $G(v^{x_2}) \pmod N$ and, for each value, we can compute

$$\gcd(G(v^{x_2}), N)$$

to see if we have split N.

One can see that each of the three stages in the attack has complexity $\tilde{O}(M)$ and so the total complexity of the attack is $\tilde{O}(M)$. As explained in [18], one expects to need to try $\sqrt{w}$ possible partitions until one finds a suitable one. Hence, to ensure that the system resists the attack we require that $\sqrt{w}\binom{n_d/2}{w/2}$ is larger than 2^m.

Note that if the Hamming weight w is not known exactly then one can take some upper bound w for the Hamming weight and then try x_i with weight $\leq w/2$. The number of possible values for each x_i is a sum of binomial coefficients which is dominated by the final term. Hence the complexity of the attack is the same as before.

It is an interesting problem to obtain a low memory version of the attack in [16].

7 Further attacks and security analysis

We have become aware of at least one other lattice attack on our system. We briefly mention this attack.

The attack applies in the case of unknown k_i. This attack is similar to the trivariate attack in subsection 5.2. One observes that $(x, y, z) = (d_1(k_2 - 1) + d_2(k_1 - 1), k_1k_2, k_1 + k_2 - 1)$ is a solution to the linear polynomial $ex + (1 - N)y - z \equiv 0 \pmod{e^2}$. Being linear, the problem is easily solved using LLL. To guard against this attack, one can take $n_d + 4n_k \geq 2n_e + 4m$.

With a system as complicated as this, there is always potential for further attacks. Since the parameters are tunable, a new attack does not necessarily destroy the system. However new attacks might add further constraints onto the parameters. We encourage readers to contact the first author if they have found a new attack on the system.

8 Summary of parameter restrictions in the two prime case

We summarise the restrictions on parameters imposed by the above attacks. For a fixed modulus length n_N which is a product of two primes of the size

$n_p = n_N/2$ we must specify parameters n_e, n_k and n_d. Let m be a security parameter (e.g., $m = 80$) so that we want security against an adversary whose total computational power is at most 2^m operations. The parameters must satisfy:

$$n_e + n_d - n_k = n_p \tag{8}$$

$$n_d \geq 2m \tag{9}$$

$$n_d \geq n_k. \tag{10}$$

The final condition is due to the key generation technique.

Now we must separate the two cases in our security analysis.

If the k_i are small (or are not supposed to contribute to security) then the restrictions on parameters also include

$$n_e \leq n_N/4 - m. \tag{11}$$

If the k_i are meant to stay private and add to the security of the system then we must have

$$n_k \geq (n_e + m)/3, \tag{12}$$

$$n_d + 4n_k \geq 2n_e + 4m. \tag{13}$$

Also, if n_e is large, then the third attack in subsection 5.2 introduces an extra parameter restriction, but this not a concern if $n_e \leq 0.5n_N$.

The sample parameters given above satisfy all of these requirements.

The case of small k_i is good for the application of equalising public and private operations using RSA. The case of large k_i seems to be more suitable for the application of minimising the cost of private operations using RSA.

It is possible to give general families of parameters for large N. For example, in the case where the k_i are private, choosing $n_k = n_d$ leads to the general family of secure parameters

$$n_e = n_p \approx n_N/2,\ n_k = n_d = (2n_p + 4m)/5 \approx 2n_p/5 \approx n_N/5.$$

For these parameters we have private operations about 2.5 times faster than standard RSA (using CRT) and public operations twice as fast as if using large public exponents (which would previously have been the case with such fast decryption).

Similarly, in the case when $n_k = 2$ we obtain the general family

$$n_e = n_p/3 - m \approx n_N/6, \quad n_d = 2n_p/3 + m \approx n_N/3.$$

This gives private operations with about 2/3 the running time of standard RSA, and public operations about 6 times faster than was previously possible in the setting of private operations faster than RSA-CRT.

9 Multiprime and Takagi

The Takagi system [22] requires e to be extremely small, and so our approach cannot be applied. We therefore turn our attention to the multiprime case. The key generation method is easily generalised to the case of products of three (or more) primes. We now consider the security of our keys in the multiprime case. See [3, 11] for related analyses of private exponent attacks on multiprime RSA.

9.1 Known k_i

As in section 4, if k_1 is known then we obtain $p_1 \equiv k_1^{-1}(k_1 - 1) \pmod{e}$ and we are in the situation of trying to factor N when given partial information about one of the factors.

Using the analogous result in Theorem 1 in the more general setting, we have

Theorem 5. *Coppersmith Let $N = pq$ where $p = N^\beta$ and $q = N^{1-\beta}$. Suppose $p_0 \equiv p \pmod{e}$ is known where $e = N^\alpha$. If $\alpha > \beta(1 - \beta)$, then the p can be recovered in polynomial time.*

In the multiprime case, assuming we have

$$N = \prod_{i=1} rp_i$$

and $\beta \approx \frac{1}{r}$ this attack works providing

$$n_e > \frac{(r-1)}{r^2} n_N$$

Therefore in order to thwart this attack in general, we need to ensure that

$$n_e < \frac{(r-1)}{r^2} n_N - m. \tag{14}$$

We apply these result in the multiprime case by taking p to be one of the prime factors of N and by taking $q = N/p$. The algorithm will produce the prime p, which means that factoring N is reduced to the problem of factoring the smaller number q. The process can be iterated using partial knowledge of the prime factors of q, or general purpose factoring methods can be applied.

Other variants of this attack can be considered (for example, considering information from several primes at once). Further analysis will appear in [10].

9.2 Unknown k_i

We generalise the linearisation attack of subsection 5.1. Suppose that $N = p_1 p_2 \dots p_r$. The equations $k_i p_i \equiv (k_i - 1) \pmod{e}$ multiply to give

$$\prod_{i=1}^{r} k_i p_i \equiv \prod_{i=1}^{r} (k_i - 1) \pmod{e}.$$

Performing the linearisation $u = \prod_{i=1}^{r} k_i$ and $v = u - \prod_{i=1}^{r}(k_i - 1)$ as before gives the equation

$$u(1 - N) \equiv v \pmod{e} \tag{15}$$

where $u \approx 2^{rn_k}$ and $v \approx r2^{(r-1)n_k}$. The continued fraction method finds (u, v) in polynomial time if $e > uv = r2^{(2r-1)n_k}$.

Once the values u and v have been computed then one can find the values k_i by factoring u and combining the factors in various combinations. One can then attempt to recover the private key as in section 4.

Hence, to resist this attack (and the extension by Theorem 3) requires $n_k \geq (n_e + m)/(2r - 1)$ (where, say, $m = 80$ for 1024-bit moduli). In practice, this condition is much more easily satisfied than the analogous condition in the two prime case. Note that the lattice attacks seem to be much less effective in the multiprime case, since the resulting polynomials are much more complicated.

Indeed, we propose the following parameters

$$n_N = 1024, n_p = 341, n_e = 261, n_d = 160, n_k = 80.$$

This choice of parameters recovers the fastest known decryption time for RSA (i.e., 160-bit CRT private exponents in the 3-prime case) but with public operation nearly 4 times faster than previously realisable using this method.

For the 2048-bit case we set $m = 128$ and suggest the parameters $(n_N, n_p, n_e, n_d, n_k) = (2048, 683, 582, 256, 156)$. Once again, we match the fastest speed for the private operations (256-bit 3-prime CRT exponents) while the public exponent is reduced from 2048 bits to 582 bits.

As in section 5.2 we can consider a birthday attack on equation (15). As before, the continued fraction approach is essentially optimal and so the birthday attack cannot perform better.

10 Equalising cost of encryption and decryption

We now consider the problem of finding RSA keys such that the cost of public and private operations is approximately equal. This can be already achieved in the usual 2-prime RSA-CRT by taking $n_e \approx n_N/3$. Our methods lead to faster performance for equalises public and private operations.

10.1 General case

We assume that $N = \prod_{i=0}^{r} p_i$, where p_i prime. We assume that, for the range of values we are interested in, Karatsuba/Montgomery multiplication is the most efficient way of computing $a^b \pmod{N}$. The complexity of this operation is therefore $O((\log_2 b)(\log_2 n_N)^{1.58})$ bit operations. We then have that encryption takes approx $O(n_e n_N^{1.58})$, while decryption using CRT takes $O(rn_d(n_p)^{1.58})$.

If we assume that $n_N \approx rn_p$ and equate both sides we have

$$n_e(rn_p)^{1.58} = rn_d(n_p)^{1.58}$$

and, as $n_d = n_p + n_k - n_e$, this simplifes to give the balancing condition of

$$n_e = \frac{n_p + n_k}{1 + r^{0.58}}$$

10.2 2-prime case

In the 1024 bit case, using $k = 2$ we get the following parameter set

$$(n_e, n_d, n_k) = (206, 308, 2)$$

However, this parameter set fails to satisfy condition (11) above for an attacker of power 2^{80}.

If we look at the 2048 bit case, the balanced parameters become

$$(n_e, n_d, n_k) = (411, 615, 2)$$

In this case, condition (11) is true for an attacker of the same power as above (i.e., 80 bits of security).

10.3 3-prime case

The balanced parameters in this case are

$$(n_e, n_d, n_k) = (119, 224, 2)$$

which satisfy easily satisfy condition(14) for $r = 3$.

Our implementation shows that encryption and decryption times for the above parameters are indeed balanced at approx 3.3ms each.

11 Performance comparison

In Figure 1 we present timings from an implementation of these RSA variants. These timings are very approximate and should only be treated as a relative guideline. Note that all entries are using the Chinese remainder theorem for decryption. In the tunable case the triple indicates the values (n_e, n_d, n_k) and we provide timings for the parameters suggested in the analysis.

As we can see from the Tunable(512,269,269) timing, it is possible reduce decryption times by approximately 50%, but with much faster public operations than had been possible with previously known techniques. Note that in the 3-prime Tunable(119,224,2) example, we have been able to successfully balance the timings of encryption and decryption.

12 Acknowledgements

The authors thank J.-S. Coron, A. May and E. Jochemsz for comments (the latter visited Royal Holloway with support from the ECRYPT Network of Excellence in Cryptology). In particular, J.-S. Coron suggested the linearisation attack in Section 5 and A. May pointed out a number of lattice-based attacks. The authors also acknowledge the useful comments provided by several anonymous referees.

Variant	Key Generation	Encryption	Decryption
RSA-CRT ($e = 2^{16} + 1$)	340	0.5	8.2
3-prime ($e = 2^{16} + 1$)	148	0.5	4.4
Takagi ($e = 2^{16} + 1$)	99	0.5	3.2
RSA small CRT private exponents	370	41	2.8
3-prime small CRT private exponents	-	41	2.3
Tunable (176,338,2), wt(d)=38	375	4.4	5.2
Tunable (512,269,269)	399	12.5	4.6
3-prime tunable (261,160,80)	170	6.3	2.3
3-prime tunable (119,224,2)	153	3.1	3.2

Fig. 1. Comparative timings for RSA variants examined in this paper. All timings are in ms. for 1024-bit moduli on a 1.80 GHz Pentium 4 desktop using the GMP library

References

1. D. Boneh and G. Durfee, Cryptanalysis of RSA with private key d less than $N^{0.292}$, in J. Stern (ed.), Eurocrypt '99, Springer LNCS 1592 (1999) 1–11.
2. D. Boneh and H, Shacham, Fast variants of RSA, *CryptoBytes*, **5**, No. 1 (2002) 1–9.
3. M. Ciet, F. Koeune, F. Laguillaumie and J.-J. Quisquater, Short private exponent attacks on fast variants of RSA, Louvain technical report CG-2003/4 (2003).
4. T. Collins, D. Hopkins, S. Langford and M. Sabin, Public key cryptographic apparatus and method. US Patent (1997).
5. D. Coppersmith, Small solutions to polynomial equations and low exponent RSA vulnerabilities, *J. Crypt.*, **10** (1997) 233–260.
6. D. Coppersmith, Finding small solutions to small degree polynomials, in J. H. Silverman (ed.), CaLC 2001, Springer LNCS 2146 (2001) 20–31.
7. A. Dujella, Continued fractions and RSA with small secret exponent, *Tatra Mt. Math. Publ.*, **29** (2004) 101-112.
8. G. Durfee and P. Nguyen, Cryptanalysis of the RSA scheme with short secret exponent from Asiacrypt '99, in T. Okamoto (ed.) Asiacrypt 2000, Springer LNCS 1976 (2000) 14–29.
9. G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers*, 5th ed., Oxford (1979).
10. C. Heneghan, Ph.D. thesis, in preparation.
11. M. J. Hinek, M. K. Low and E. Teske, On some attacks on multi-prime RSA, in K. Nyberg and H. M. Heys (eds.), SAC 2002, Springer LNCS 2595 (2003) 385–404.
12. N. A. Howgrave-Graham, Finding small solutions of univariate modular equations revisited, in M. Darnell (ed.), Cryptography and Coding, Springer LNCS 1355 (1997) 131–142.
13. C. H. Lim and P. J. Lee, Sparse RSA secret keys and their generation, Proc. of 3rd Annual Workshop on Selected Areas in Cryptography (SAC'96), (1996) 117–131.
14. A. May, Cryptanalysis of unbalanced RSA with small CRT-exponent, in M. Yung (ed.) CRYPTO 2002, Springer LNCS 2442 (2002) 242–256.
15. C. A. M. Paixão, An efficient variant of the RSA cryptosystem, preprint (2003).
16. G. Qiao and K.-Y. Lam, RSA signature algorithm for microcontroller implementation, J.-J. Quisquater and B. Schneier (eds.), CARDIS '98, Springer LNCS 1820 (2000) 353–356.
17. M. O. Rabin, Digital signatures and public key functions as intractable as factorisation, Technical report MIT/LCS/TR-212 (1979).

18. D. Stinson, Some baby-step-giant-step algorithms for the low Hamming weight discrete logarithm problem, *Math. Comp.* **71**, No. 237 (2001) 379–391.
19. H.-M. Sun and M.-E Wu, An Approach Towards Rebalanced RSA-CRT with Short Public Exponent, Cryptology ePrint Archive, 2005/053.
20. H.-M. Sun and C.-T. Yang, RSA with balanced short exponents and its application to entity authentication, in S. Vaudenay (ed.), PKC 2005, Springer LNCS 3386 (2005) 199–215.
21. H.-M. Sun, W.-C. Yang and C.-S. Laih, On the design of RSA with short secret exponent, in K. Y. Lam et al (eds.), ASIACRYPT '99, Springer LNCS 1716 (2000) 150–164.
22. T. Takagi, Fast RSA-type cryptosystem modulo $p^k q$, in H. Krawczyk (ed.), CRYPTO '98, Springer LNCS 1462 (1998) 318–326.
23. J. W. M. Turk, Fast arithmetic operations on numbers and polynomials, in H. W. Lenstra Jr. and R. Tijdeman (eds.), Computational methods in number theory, Part 1, Mathematical Centre Tracts 154, Amsterdam (1984).
24. E. R. Verheul and H. C. A. van Tilborg, Cryptanalysis of 'less short' RSA secret exponents, Applicable Algebra in Engineering, Communication and Computing, Vol. 8 (1997) 425–435.
25. M. Wiener, Cryptanalysis of short RSA secret exponents, IEEE Trans. Inf. Th., 36 (1990) 553–558.
26. M.-E. Wu, A Study of RSA with Small CRT-Exponent, Thesis of Master Degree, Department of Applied Mathematics, National Chiao Tung University, Taiwan, June 2004.