Twin RSA

Arjen K. Lenstra[1,2], Benjamin M.M. de Weger[2]

[1] Lucent Technologies, Bell Laboratories, Room 2T-504
600 Mountain Avenue, P.O.Box 636, Murray Hill, NJ 07974-0636, USA
[2] Technische Universiteit Eindhoven
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
November 9, 2005

Abstract. We introduce *Twin RSA*, pairs of RSA moduli $(n, n+2)$, and formulate several questions related to it. Our main questions are: is Twin RSA secure, and what is it good for?

Keywords: recreational cryptography

1 Introduction

Regular RSA moduli are constructed by multiplying two more or less randomly selected primes of appropriate sizes. As a result, representation of a regular $2N$-bit RSA modulus requires about $2N$ bits. To save on the representation size of RSA moduli, several methods were proposed in [8], some of which were broken in [1]. An often reinvented folklore approach to generate $2N$-bit RSA moduli that can be represented using just N bits, published in [5] along with several simple variants, still seems to be unbroken. This simple method works as follows. For an N-bit number x that is known from the context, repeatedly select an N-bit prime p at random until the integer part q of the quotient $(x+1)2^N/p$ is prime, then the most significant N bits of the RSA modulus $n = pq$ are given by x. Faster variants add some slack to x and replace q by $q+1$ until it is prime, but the principle remains the same. Since x is known from the context—or can for instance be chosen as 2^{N-1}—the N least significant bits suffice to represent the $2N$-bit RSA modulus n. If one is willing to also consider moduli of unbalanced factor sizes, e.g. a product of primes of sizes $\frac{1}{2}N$ and $\frac{3}{2}N$, respectively, then, as was shown in [5], a $2N$-bit modulus can even be represented using $\frac{1}{2}N$ bits. In particular this shows that pairs of RSA moduli can be generated in such a way that the pair can be represented using the space of a single regular or even a half unbalanced RSA modulus.

In this note we present a method that achieves the same 'compression ratio' for pairs of RSA moduli, in a different and esthetically more pleasing way. Our method is implicit in one of the methods described in [6] and thus not new. The reason we present this particular case of the method from [6] is the fact that the possibility of the construction is usually met first with amazement, quickly followed by skepticism about the security, and finally with puzzled resignation that the resulting moduli indeed look hard to break. Thus, we would like to

offer it as a challenge to a wider audience, hoping for either a better security argument than what can be found in [6], or a more effective cryptanalysis.

Another question we want to pose with this note is: are there any applications of Twin RSA that are more interesting than the ones we have been able to offer so far? We realize that it is by no means good marketing policy to present a new cryptographic method without convincing evidence of its practical potential or cryptographic significance. On the other hand, publishing the method despite the fact that we cannot think of a sensible application ourselves, at least has the potential to uncover new possibilities by bringing it to the attention of members of the practical cryptographic community who may never have realized that such remarkable pairs of RSA moduli were possible—or secure.

The remainder of this note is organized as follows. Our method to generate RSA moduli with a fixed prescribed difference is described and discussed in Section 2. A few generalizations are offered in Section 3, and Section 4 concludes this note with two factoring challenges.

2 Twin RSA

Generation of RSA moduli with any prescribed even integer difference d is an easy application of the Chinese Remainder Theorem. The details are described in Algorithm 1 below.

Algorithm 1. Let $d \neq 0$ be a small fixed even integer and let $2N$ be the bitlength of the RSA moduli to be generated.

1. Select two random N-bit primes p and q.
2. Use the Chinese Remainder Theorem to calculate the least positive integer n such that $n \equiv 0 \bmod p$ and $n \equiv -d \bmod q$ and let $r = n/p$ and $s = (n+d)/q$.
3. If n or $n + d$ does not have bitlength $2N$, or if r or s is composite, then return to Step 1.
4. Output the pair of RSA moduli $(n, n + d)$ with factorizations $n = pr$ and $n + d = qs$.

For actual RSA applications of the resulting moduli, co-primality requirements with respect to one's favorite public exponent(s) and $p-1$, $q-1$, $r-1$, and $s-1$ have to be included in the above description.

Twin RSA. We introduce the term *Twin RSA* for the pair of moduli that results from Algorithm 1 when $d = \pm 2$.

Abundance. A single moment of reflection learns that it is most likely the case that Twin RSA moduli are abundant. The Prime Number Theorem combined with the assumption that the factorizations of n and $n+2$ are independent leads to the conjecture that the number of Twin RSA moduli up to x is asymptotically equal to $cx/(\log x)^4$, for some positive constant c. The same argument applies to the general case $(n, n + d)$ for even d.

Runtime of Algorithm 1. Based on the Prime Number Theorem and the runtime of a single probabilistic compositeness test (using standard arithmetic),

one may expect that each execution of Step 1 of Algorithm 1 takes expected runtime $O(N \times N^3 + N \times N^3) = O(N^4)$. Assuming that r and s behave as independent random N-bit numbers, they will simultaneously be prime with probability proportional to $1/N^2$, again based on the Prime Number Theorem. Using standard arithmetic, the computation in Step 3 of Algorithm 1 can be expected to take runtime $O(N^3 + (1/N) \times N^3) = O(N^3)$ (where the factor $1/N$ accounts for the probability that r is not found to be composite, in which case compositeness of s has to be tested as well) and dominates the runtime of the computation in Step 2. Overall, we find that the expected runtime becomes $O(N^2(N^4 + N^3)) = O(N^6)$.

Practical considerations. A practical speed-up can be obtained by generating the sequence of candidate p's in Step 1 of Algorithm 1 using sieving based methods, independently from the similarly generated sequence of candidate q's. Also, upon return to Step 1, one may decide to generate a new p or a new q, but not both.

A more substantial speed-up is obtained by allowing more candidate quotients per prime pair (p, q): in Step 3 of Algorithm 1, add some slack to the lengths and find the smallest positive integer k such that the quotients $(n+kpq)/p$ and $(n + kpq + d)/q$ are both prime. The resulting method can be made to work in expected time $O(N^5)$, but results in RSA moduli pairs that are somewhat less 'elegant' because their factors will have slightly different sizes. The more time one is willing to invest in the generation of RSA moduli pairs with fixed prescribed difference, the closer factor sizes one will be able to obtain.

Independent generation of n and $n + d$? Given d, the moduli n and $n + d$ are generated simultaneously by the single party that executes Algorithm 1. As a result, that same party knows the factorizations of both moduli. We are not aware of a simple variant of our approach where a first party generates p, a second party generates q, and the two parties engage in a straightforward protocol that results in RSA modulus n for the first party and $n + 2$ for the second party, without either party knowing the factorization of the other party's modulus. We pose the challenge of developing such a protocol because it may lead to more interesting applications of Twin RSA.

Security? With variable d, and as argued in [6], it seems impossible to distinguish an RSA moduli pair $(n, n + d)$ generated using our method from a pair of regular RSA moduli that happens to have difference d. This suggests that 'our' pairs $(n, n + d)$ are as secure as 'regular' pairs: being able to do the private RSA operation for either modulus is independent of whether the private RSA operation can be carried out for the other modulus or not.

But what about a fixed choice of d, such as $d = 2$? Most certainly, and intuitively, Twin RSA looks highly suspicious. A more subtle security argument is required in this case, which is one of the challenges we pose with this note. All we can present at this point in support of our belief in the security of Twin RSA—if one is willing to believe in the hardness of factoring to begin with—is the circumstantial evidence that generations of factorers who contributed to

the Cunningham factoring project (cf. [3]), where factorizations of $b^k \pm 1$ are collected for $2 \leq b \leq 12$ up to high powers k, have never been able to profit from the factorization of a certain $b^k \pm 1$ to factor the corresponding $b^k \mp 1$.

We also believe that, even when the two moduli share the same public exponent e (such as in practice often happens, e.g. $e = 65537$), the two corresponding private exponents will be completely different and independent for all practical purposes. So, here we do not see any reason for additional suspicion either.

Applications? As mentioned in the Introduction, one of our reasons to publish this note is that we are curious to know if there are any interesting applications of Twin RSA. Here the 'application' may either wear a white or a black hat, as long as it enables us to do something we were unable to do before. Some rather unconvincing white hat applications use the twin modulus as backup in case the other one is believed to be compromised, or use one for encryption and the other for signature purposes—conveniently avoiding the cost of two different certificates for the two different keys. In situations where the size of two standard X.509 certificates is too costly, e.g. because of memory or bandwidth restrictions such as in the mobile telephony world, Twin RSA can be useful. We can envisage one certificate, in which one Certificate Authority (CA) signature is used to bind the owner's identity information to two RSA key pairs (a Twin RSA pair), which is represented by just one of the two moduli and a common (usually very small) public exponent. This will save almost half of the space needed to represent the public key (and when predetermined bits are used, as explained in the next section, a reduction to 25% even becomes possible). One of the conditions that has to be posed is that there must be a standardized way of interpreting this public key representation, of how to extract both public keys, and of establishing the allowed key usages for both keys. It should be possible to do this with only minor adaptations to existing certificate standards such as X.509. Another condition is that the CA should explicitly guarantee that it has seen proof that the certificate owner is in possession of both private keys.

Are there applications with more cryptographic significance? And are there any applications with 'interesting' cryptanalytic potential?

3 Generalizations

Multiple RSA. With a proper choice of even differences d_i for $0 \leq i < t$ and $d_0 = 0$, our method allows construction of t-tuples of RSA moduli $(n + d_i)_{i=0}^{t-1}$, if one is willing to accept moduli where the size of the largest of the two factors is $t - 1$ times the size of the smallest one. For large modulus sizes this may be acceptable, and possibly even desirable (cf. [7]). Note that, for instance, $d_1 = 2$, $d_2 = 4$ will not work: the set $\{d_i \bmod 3 : i = 0, 1, 2\}$ equals the full residue set $\{0, 1, 2\}$ modulo the prime 3, so that for any integer n there will always be an $i \in \{0, 1, 2\}$ such that $n + d_i$ is divisible by 3. More in general, the set consisting of the d_i's, for $0 \leq i < t$, should not contain a full residue system modulo any prime $\leq t$. The latter condition is obviously necessary, but also sufficient. This can easily be seen as follows: for each prime $q \leq t$ there exists an

$a_q \in \{0, 1, \ldots, q-1\}$ such that $d_j \not\equiv a_q \bmod q$ for all j. We now restrict ourselves to n that are equal to $-a_q \bmod q$ for all $q \leq t$, which can be obtained by Chinese Remaindering. Then q does not divide $n + d_j$ for all $q \leq t$ and all j.

Algorithm 2. Let $t \geq 2$, let $(d_i)_{i=0}^{t-1}$ with $d_0 = 0$ be a set of small even integers that does not contain a full residue system modulo any prime $\leq t$, and let tN be the bitlength of the RSA moduli to be generated.

1. Select t random N-bit primes p_i, $0 \leq i < t$.
2. Use the Chinese Remainder Theorem to calculate the least positive integer n of bitlength at most tN such that $n \equiv -d_i \bmod p_i$ for $0 \leq i < t$, and let $n_i = n + d_i$ and $r_i = n_i/p_i$ for $0 \leq i < t$.
3. If n_i does not have bitlength tN for an $i \in \{0, 1, \ldots, t-1\}$ or if there is an $i \in \{0, 1, \ldots, t-1\}$ such that r_i is composite, then return to Step 1.
4. Output the t-tuple of RSA moduli $(n_i)_{i=0}^{t-1}$ with factorizations $n_i = p_i r_i$.

As before, Algorithm 2 can be seen to have expected runtime $O(N^{t+4})$.

Twin RSA with predetermined bits. Our methods can simply be combined with the methods from [5], again at the cost of severely unbalancing the factor sizes (cf. [6]), in a way similar to the construction of the colliding X.509 certificates that were reported in [2]. For instance, for any N-bit number x and assuming that N is even, a pair of $2N$-bit RSA moduli $(n, n + d)$ can be constructed such that the leading N bits of n are given by x, and such that both n and $n + d$ are the products of a $3N/2$-bit prime and an $N/2$-bit prime. In the interest of space we elaborate.

Algorithm 3. Let $d \neq 0$ be a small fixed even integer, let x be an integer with $2^{N-1} \leq x < 2^N$ for some even positive integer N such that $2N$ is the bitlength of the RSA moduli to be generated.

1. Select two random $N/2$-bit primes p and q.
2. Use the Chinese Remainder Theorem to calculate the least positive integer $b < pq$ such that $b \equiv -x2^N \bmod p$ and $b \equiv -x2^N - d \bmod q$, let $n = x2^N + b$, and let $r = n/p$ and $s = (n + d)/q$.
3. If r or s is composite, then return to Step 1.
4. Output the pair of RSA moduli $(n, n + d)$ with factorizations $n = pr$ and $n + d = qs$. The most significant N bits of n and $n + d$ are the same and are given by x.

Big brother RSA. There is no need to restrict to a 'twin' of the form $n + d$ for an RSA modulus n: the same method can be used to generate a 'big brother' $an + d$ for any integer $a \neq 0$ (that must be coprime to d). Thus, one can generate pairs of Sophie Germain RSA moduli $(n, 2n+1)$ where the big brother follows by pasting an additional 1-bit 'at the end' of n (pasting a 1-bit 'before' n was already covered by taking $d = 2^{2N}$ in the standard case $a = 1$), or 'complementary pairs' where n with negated bits (except the least significant one), i.e., $2^{2N+1} - n$, is again an RSA modulus. Similarly, twins $(n, n + 2)$ with big brother $2n + 1$ can

be generated, or longer sequences of pasted-on bits on either side of n (such as $n, n||1 = 2n+1, n||11 = 2(2n+1)+1, \ldots$ if pasted on 'at the end'), if one is willing to tolerate unbalanced RSA moduli, and complementary pairs can be generated with respect to any radix (i.e., not just radix 2). Note that a 'little brother' $(n+1)/2$ for $(n, n+2)$ is unfortunately impossible because $n \equiv 2 \bmod 3$ so that the little brother is always divisible by 3.

Remarks. None of the approaches mentioned in this or the previous section yields a representation saving that is larger than what can be obtained by using just the methods from [5].

The practical speed-up tricks that applied to Algorithm 1 can, with the obvious changes, be applied to Algorithms 2 and 3 as well. Furthermore, Algorithm 3 allows a similar generalization to t-tuples—Multiple RSA with predetermined bits—as was presented for Algorithm 1 in Algorithm 2. The restriction to even N in Algorithm 3 is not crucial and just for ease of exposition. Different sized predetermined parts can be handled in essentially the same way. Putting the predetermined part in the least significant bits, or spreading it over most and significant bits, is possible too. The underlying idea of all these generalizations is the same: combine any of the methods described in [5] with the Chinese Remainder Theorem.

We gladly leave other variants of our idea, namely *Twin Discrete Log*, or subvariants such as *Twin (Hyper)Elliptic Discrete Log*, for others to pursue. This could take the not so challenging form of twin prime fields $(\mathbf{F}_p, \mathbf{F}_{p+2})$, of equally uninteresting generators $(g, g+1)$ (no need to jump by 2 this time!) of the same full multiplicative group, or, the sole interesting case (cf. [6]), of generators $g, g+1$ of a relative small prime order subgroup of a multiplicative group of a prime field—actually, the range of possibilities is only limited by one's imagination. What it would be good for is an entirely different matter.

4 Factoring challenges

Below the n is given of a Twin RSA pair $(n, n+2)$ consisting of two 1024-bit RSA moduli that each have one 256-bit prime factor, along with the 256-bit factor p of n. Note that the Twin RSA pair $(n, n+2)$ can be represented using just 512 bits. Finding the 256-bit factor of $n+2$ should be borderline possible using the elliptic curve factoring method (ECM), now that a 219-bit (66-digit) factor found using ECM has been announced [4]. Can anyone factor $n+2$ faster, if at all possible using the known factorization of n?

$n =$ 80000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
11B9E917 7E937E9D 6AAB2AB0 28940F89 BEEC962C 286F28A9 DB965A18 688EE789
ACF43457 0E44D41B F837B4EF 9E435CFB 56C2E2F7 00EE8DDD 2A3ECF9F B2EA360D

$p =$ EF0650E4 304D1242 F3DBAF45 80BFB645 77527C60 3C1E3006 BCDE98FB 5F97E507.

Obviously, with the fixed prefix this size cannot be recommended for practical purposes. A more substantial factoring challenge is given by the Twin RSA pair

$(m, m+2)$, for which the 2048-bit m is given below along with its 512-bit factor q. Finding the 512-bit prime factor of $m+2$ using ECM can be expected to be about as hard as factoring $m+2$ using the Number Field Sieve. Can $m+2$ be factored in an easier way, possibly using the known factorization of m?

$m =$ 80000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
396099A3 5F9D2B49 E7BB729E 9542A7B0 A1FAD34B EE884199 E29A5DB4 E49DE1C8
279682F4 2A92FBFF 4F0F891F 65638997 B28D26DA 10B7529A 40CFA534 8BB95BE8
ADF4A21B 7DC562D4 93590D53 6B6124C5 6DB5D693 1004A7B4 C031C401 A4B6E1E8
EA5C8362 E7B2DB3F BFDEF87D 75311FEA 7D9BF1C3 9E3E64DF 9163E468 6D5D2711

$q =$ C1A25EAE FF7A187A 6F793972 199F192B 3D2912DF BD4586CF 4D1CE614 6D75992F
AB3A7E2A 2149CD3C 4FE9F8A4 8DF3B515 74660F13 696BC4BD 4808E475 69E414B9.

As far as we know this last Twin RSA size can be recommended for practical purposes, either with or without a fixed 1024-bit prefix. With the prefix it allows representation of two 2048-bit moduli at the cost of representing 1024 bits.

References

1. D. Coppersmith, *Finding a small root of a bivariate integer equation; factoring with high bits known*, Eurocrypt'96, LNCS 1070, Springer-Verlag 1996, 178–189.
2. Colliding X.509 Certificates, see `http://www.win.tue.nl/~bdeweger/CollidingCertificates/`.
3. Cunningham project, see `http://www.cerias.purdue.edu/homes/ssw/cun/`.
4. B. Dodson, *email announcement of the ECM factorization of M963*, April 6, 2005.
5. A.K. Lenstra, *Generating RSA moduli with a predetermined portion*, Asiacrypt'98, LNCS 1514, Springer-Verlag 1998, 1–10.
6. A.K. Lenstra, B.M.M. de Weger, *On the possibility of constructing meaningful hash collisions for public keys*, ACISP 2005, LNCS 3574, Springer-Verlag 2005, 267–279.
7. A. Shamir, *RSA for paranoids*, RSA Laboratories' Cryptobytes, v. 1, no. 3 (1995) 1–4.
8. S.A. Vanstone, R.J. Zuccherato, *Short RSA keys and their generation*, J. Cryptology, **8** (1995) 101–114.