Factoring and RSA Codes using DERIVE

Johann Wiesenbauer,
Technical Univ. of Vienna,
j.wiesenbauer@tuwien.ac.at

Much to the surprise of many mathematicians who thought like G.H. Hardy that the "very remoteness of number theory from ordinary human activities" should keep it "gentle and clean", things have changed dramatically in the last decades and today findings and algorithms from number theory play an important role in computer science, coding theory and cryptography. On the other hand, some of these applications have also stimulated the research in certain areas of number theory a lot. A nice example of this sort is the interplay between RSA cryptosystems and the theory of factoring large numbers which we are going to study in the following using the current version 4.09 of DERIVE for Windows. (As for computation times mentioned in the following, they were obtained on a Pentium 166 PC with 32 MB.)

The Classical RSA Cryptosystem and What's Wrong with it

RSA is a so-called public-key cryptosystem that was invented in 1977 by R.Rivest, A.Shamir and L.Adleman (cf. [1]). They proposed the following procedure: Take two large primes p and q (we will say more on how to select p and q properly a little bit later) and find their product n = pq. Then choose a natural number e less than n and relatively prime to (p-1)(q-1) and find a natural number d < (p-1)(q-1) that is a solution of

$$ex \equiv 1 \bmod (p-1)(q-1) \quad (*)$$

The public key is the pair (n,e), the private key is d. The factors p and q of n must be kept secret or even destroyed. Encrypting a message m, which we assume to be represented as a natural number < n, is done by means of

$$c = m^e \bmod n$$

whereas decrypting c uses the formula

$$m = c^d \bmod n$$

In the workshop a famous example that was posed as a problem by the inventors of RSA Codes themselves back in 1977 using the simple encoding scheme A=01, B=02,...,Z=26, space=00 to encode the message

THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE

is studied in detail.

In particular, by trying out several values for e we will see that the value for d we get by solving the conguence (*) is usually too large. Fortunately, it is very easy to compute the smallest possible value for d. All we have to do is to solve the following congruence

$$ex \equiv 1 \bmod \operatorname{lcm}(p-1, q-1) \quad (**)$$

where we have replaced the modulus in (*) by a smaller one (cf. [2] for the very elementary proof).

How to Find Large Primes p and q

Obviously everbody who is capable of factoring the modulus n can also compute the secret key. Therefore n should be sufficiently large. At present, the most common bit length of n is 512, which corresponds to about 155 decimal digits, but this may no longer be safe in a few years. For this reason already now larger n with 768, 1024 or even 2048 bits are in use. As its prime factors p and q have about half as many digits as n itself, this poses a problem that usually frightens laymen a lot, namely how to find such large primes.

We will see that using DERIVE this is actually a very easy problem if we put up with the fact that the provided numbers p and q are only probable primes! To be more precise, they are thought to be prime by DERIVE because they have no prime factor $< 2^{10}$ and have passed 6 so-called Rabin-Miller tests.

As for Rabin-Miller tests, assuming that N is an odd natural number and a an integer with 0 < a < N, we not only demand as in an ordinary Fermat test that Fermat's Little Theorem for N w.r.t. to the base a is fulfilled, namely

$$a^{N-1} \equiv 1 \bmod N,$$

but also that either all numbers of the following sequence

$$a^{(N-1)/2}, a^{(N-1)/4}, \ldots, a^{(N-1)/2^s}$$

are congruent to 1 mod N or one of them is –1 mod N, where s is the biggest number s.t. 2^s divides N - 1. (The additional condition comes from the trivial fact that for a prime N the congruence $x^2 \equiv 1$ mod N has the solutions $x \equiv \pm 1 \bmod N$ only.) Setting $t := (N-1)/2^s$, we can also say that N passes the Rabin-Miller test with respect to the base a if and only if

$$a^t \equiv \pm 1 \bmod N \text{ or } a^{t2^r} \equiv -1 \bmod N \text{ for all r with } 0 < r < s$$

A very fast DERIVE-implementation of the Rabin-Miller test could look like this:

```
RABIN_MILLER(n, a) := IF((ITERATE(IF(k_ = 2 OR a_ = -1, [a_, k_],
  [MODS(a_^2, n), k_/2]), [a_, k_], ITERATE([- ABS(MODS(a^o_, n)), (n - 1)/o_],
  o_, ITERATE(IF(MOD(n_, 2) = 1, n_, n_/2), n_, n - 1), 1)))SUB1 = -1, true, false)
```

In particular, we will see that in the interval [1,10000] the number of composites that pass the Fermat test for the base a=2 drops from 22 to 5 when replacing the Fermat test by the Rabin-Miller test, what is usually put in the following way: There are 22 pseudoprimes, but only 5 strong pseudoprimes w.r.t. 2 below 10000. Moreover, we will see that there are also composite numbers like 561 (this is the smallest one, but there are even infinitely many of them!) which pass the Fermat test for every base a with 0<a<N and (a,N)=1. Again, those numbers, which are called Carmichael numbers, do not exist if the Fermat test is replaced by the Rabin-Miller test.

More precisely, Rabin showed that any composite odd number N > 9 passes the Rabin-Miller test for at most φ(n)/4 of all bases a with 0 < a < N, where φ denotes Euler's φ-function which is given in NUMBER.MTH. This boundary can actually be reached for numbers of the form p(2p-1), where p is an odd prime, and for certain Carmichael numbers with exactly three prime factors. As a matter of fact, by accumulating numbers of this kind, which represent the "worst case" for the Rabin-Miller test in

some sense, the author was able to find an actual example for a composite number that is a prime for DERIVE (at least in DfW 4.09, as this may change in future versions!):

```
106219·(2·106219 − 1) = 22564845703
PRIME(22564845703) = true
PRIME(22564845703, 7) = false
NEXT_PRIME(22564845702) = 22564845703
NEXT_PRIME(22564845702, 7) = 22564845803
FACTOR(22564845703) = 22564845703
```

One can also see that after increasing the number of Rabin-Miller tests by one the number in question is recognized as composite by PRIME and NEXT_PRIME. (Unfortunately, there is no way to make FACTOR work.)

We will be also dealing with the question: What happens if p or q (or even both) are not primes, in particular, does the RSA-cryptosystem still work?

RSA Attacks Based on the Factorization of the Modulus n

Although in general, factoring large numbers is a very tough problem - and the security of RSA codes relies on the fact that in principle this will not change in the foreseeable future - one should take some precautions as regards the selection of p and q, otherwise it could be surprisingly simple to achieve the factorization n = p q.

For instance, there was a remarkable case in Austria, when a criminal tried to fool the local authorities by announcing a new series of letter bombs in a writing that was decrypted by means of RSA . The modulus he used had 243 digits which is quite a lot. Even so, as can be seen in the following, it is possible to factor this number within fractions of a second using DERIVE and an ancient factorization method by Fermat.

```
FERMAT(n) := (ITERATE(IF(NUMBER(SQRT(b_)), [a_, b_, [a_ - SQRT(b_),
   a_ + SQRT(b_)]], [a_ + 1, b_ + 2·a_ + 1, c_]), [a_, b_, c_],
   [- FLOOR(- SQRT(n)), FLOOR(- SQRT(n))^2 - n, [ ]]))SUB3

n :=
630548215070129547156718332495889632234434145411971275888376987603
260225252787926135276738944105689100036295535868141424386536403649
578707699128189491432138631900590774729214990015369102760964884776
344849717811484309528915040117952098061886881

v:= =FERMAT(n)=
[251107191269013549761909333958671246802408057112768448862509598241
562051889494061847352957883875611351675294302430759487 99,
251107191269013549761909333958671246802408057112768448862509598241
562051889494061847352957883875611351675294351184297803 19]

vSUB2 - vSUB1 = 4875353831520
```

As the above compuatations show, his mistake was - and this is was we can learn from this example - to choose p and q too close together.

Another thing one should take of is that both p − 1 and q − 1 contain a large prime factor. By heuristic arguments one can easily see that this precondition is usually

fulfilled if the primes p and q are chosen at random, but if this is not the case Pollard's highly efficient (p − 1)-method comes into play. The basic idea is that if we can find an integer e that is divisible by p − 1 then for any integer a coprime to N

$$a^e = (a^{p-1})^{e/(p-1)} \equiv 1 \bmod p$$

implies $p \mid (a^e - 1, N)$. If p − 1 has relatively small prime factors only, one could find such an e by taking e.g. the l.c.m. of all prime powers up to a certain boundary s. If s is not smaller than the biggest prime factor of p − 1 this will work! In the following fast DERIVE-implementation we use numbers coprime to 210 instead of primes.

```
PMINUS1(n, a, s) := GCD((ITERATE(IF(GCD(a_ - 1, n) > 1 OR IF(s < b_SUB2,
true, false, false), [a_, b_], [MOD(a_^PRODUCT(b_), n), b_ + [0, 210, 210, 210, 210,
210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210,
210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210,
210, 210, 210, 210, 210, 210, 210, 210, 210, 210]]), [a_, b_], [MOD(IF(a, 3, a,
3)^160030080000, n), [210, 1, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61,
67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 121, 127, 131, 137, 139, 143, 149,
151, 157, 163, 167, 169, 173, 179, 181, 187, 191, 193, 197, 199, 209]]))SUB1 - 1, n)
```

As an example we compute

```
PMINUS1(2^67 - 1) = 193707721

FACTOR(193707721 - 1) = 2^3·3^3·5·67·2677
```

in 0.2s, which is faster than by means of the built-in FACTOR! (Note that the default values 3 and ∞ are taken for a and s, respectively, unless they are otherwise specified.)

In a similar way the (p+1)-method works, whenever N has a prime factor such that p + 1 splits up into relatively small prime factors and $(a^2 - 4 / p) = -1$. If the value of the latter Jacobi symbol happens to be 1 - of course, one does not know its value in advance, since p is not known - the outcome will be the same as with the (p − 1)-method, but taking a much longer time for it. By studying the code below, one will notice that the Lucas sequence (V_n) with $V_0 = 2, V_1 = a, V_n = aV_{n-1} - V_{n-2}$ $(n \geq 2)$ is used. In particular, NUMBER.MTH should be loaded.

```
PPLUS1(n, a, s) := GCD((ITERATE(IF(GCD(a_ - 2, n) > 1 OR IF(s < b_SUB2, true,
false, false), [a_, b_], [V_MOD(PRODUCT(SELECT(c_ < 100 OR PRIME(c_), c_,
b_)), a_, 1, n), b_ + [0, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210,
210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210,
210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210,
210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210, 210,
210, 210]]), [a_, b_], [V_MOD(160030080000, IF(a, 3, a, 3), 1, n), [210, 1, 11, 13,
17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103,
107, 109, 113, 121, 127, 131, 137, 139, 143, 149, 151, 157, 163, 167, 169, 173, 179,
181, 187, 191, 193, 197, 199, 209]]))SUB1 - 2, n)
```

Using this method and a=5, the prime factor 225974065503889 of $10^{102} + 1$ was found.

The (p-1)-method and the (p+1)-method are particularly important in connection with RSA Codes, because one could take measures against them by choosing p and q in an appropriate way. There are several other factorization methods like e.g. the Elliptic

Curve Method (ECM) by Lenstra or the Number Field Sieve (NFS), which are extremely powerful, but where there is no protection in advance. Again, some examples will be given in the workshop.

References

[1] R.L.Rivest,A.Shamir and L.Adleman, A Method For Obtaining Digital Signatures and Public-Key Cryptosystems, Comm.ACM 21 (1978), 120-126.

[2] J.Wiesenbauer, Number Theory with DERIVE - Some Suggestions for Classroom Teaching, Proceedings of the 2nd Krems Conference on Mathematics Education (ed. H.Heugl,B.Kutzler), Chartwell-Bratt,1994