Factoring

Factoring

The only known method which breaks the RSA cryptosystem involves factoring the public modulus. To prevent this type of attack, one must be aware of the current state of the art in factoring large numbers, so as to avoid those situations in which a fast factoring algorithm exists. We will now examine some of these factoring techniques. Throughout this section, if not otherwise stated, n will indicate a "large" odd integer that we wish to find a factor of (even integers, it goes without saying, have an obvious factor).

Trial Division (Brute Force)

Testing each of the primes up to the square root of n for divisibility, will certainly produce a factor if n is not prime. The problem with this technique is that it is very slow and for large n with no small prime factors it may not find a factor in any reasonable time. Our day to day experience with this naive approach is misleading, since we usually do not meet any integers of the size needed for a cryptosystem which need to be factored.

Fermat Factoring

Theorem: *Any odd integer n > 1 can be written as the difference of two integer squares.*

Pf: Write $n = m_1 m_2$ with $m_1 \leq m_2$ (in a worst case, $m_1 = 1$ will work). Since n is odd, each of m_1 and m_2 are odd. Let $a = \frac{1}{2}(m_1 + m_2)$ and $b = \frac{1}{2}(m_2 - m_1)$. Note that a and b are integers. Then $m_1 = a-b$ and $m_2 = a+b$, so $n = m_1 m_2 = (a-b)(a+b) = a^2 - b^2$.

Now suppose we wish to find a factor of the odd integer n (> 1). Examine, in turn, the numbers $n, n+1^2, n+2^2, n+3^2, \ldots$ until you find a square (this is guaranteed to exist by the theorem), say $n + b^2 = a^2$, then $n = a^2 - b^2 = (a+b)(a-b)$ and so, factors of n have been located.

Example

Find a factor of n = 152398989.

Looking for a square in the sequence, 152398989, 152398990, 152398993, 152398998, 152399005, 152399014, 152399025, 152399038, ... we have $(12344.998541...)^2$, $(12344.99582...)^2$, $(12344.99870...)^2$, $(12344.99890...)^2$, $(12344.99918...)^2$, $(12344.99955..)^2$, **$(12345)^2$**.

Thus, $n = (12345)^2 - 6^2 = (12351)(12339)$.

Speed Up

The method can be sped up a bit by observing that the last digit of a square must be a 0,1,4,5,6 or 9. However, taking square roots to determine if a number is a square is a slow operation, and this naive approach is therefore not very fast. A better algorithm to search for squares would be to examine the sequence of integers given by

$$\left(\lceil \sqrt{n} \rceil + i\right)^2 - n$$

for a square, where $i = 0,1,2,\ldots$. This algorithm does not take as many square roots and those it does take are of smaller numbers. In the above example, using this algorithm, the factorization would have been found in the first step (but, to be honest, trial division would have found a different factorization in one step as well). Factors which are nearly equal will be found fairly quickly by this procedure, thus in the RSA application one must make sure that the two primes are not too close together.

p-1 Factoring Algorithm

This algorithm is due to Pollard and dates from 1974. Choose a "bound" B. Compute $a \equiv 2^{B!} \bmod n$. If $d = \gcd(a-1, n)$ satisfies $1 < d < n$, then d is a factor of n.

To see why this works, consider a prime p which divides n. If p-1 divides B! (which will be the case if, for instance, p-1 only has small prime divisors) then we have $B! = (p-1)k$ for some integer k. Now, since $a \equiv 2^{B!} \bmod n$, we also have $a \equiv 2^{B!} \bmod p$ since p|n. By Fermat's theorem $2^{p-1} \equiv 1 \bmod p$, so $a \equiv 2^{B!} = (2^{p-1})^k \equiv 1 \bmod p$. Therefore $p \mid (a-1)$ and $p \mid n$, so $p \mid \gcd(a-1,n) = d$. Hence, $1 < d$ and if $d < n$, d will be a proper divisor of n.

Example

Let n = 15770708441. Choose B = 180.

Then a = 11620221425 and we compute d = 135979.

We get the factorization 15770708441 = (135979) (115979).

The reason that factorization worked is that d-1 = 135978 = 2(3)(131)(173) has only small prime factors.

Any B ≥ 173 would have worked for this n.

B

The choice of B is crucial in this algorithm. If B is small, the algorithm will run quickly, but the chance of success is small. On the other hand, if B is large, the algorithm will find a factor, but the runtime will be prohibitively slow (comparable to trial division).

In the RSA application, one must ensure that the primes p and q have the property that p-1 and q-1 have at least one large prime factor to avoid an attack by this method. We shall see a generalization of this method later when we consider elliptic curves.

Factor Base Algorithms

Most of the best modern factoring algorithms are based on a generalization of the idea behind Fermat factorization.

Namely, if we can find a congruence of the form $t^2 \equiv s^2 \mod n$, with $t \neq \pm s \mod n$, then since $n | t^2 - s^2 = (t+s)(t-s)$ while it doesn't divide either $t+s$ or $t-s$, n must have some non-trivial common factor with both $t+s$ and $t-s$. One of these common factors is $a = \gcd(t+s,n)$ and the other is $b = n/a$.

Example

Suppose we want to factor n = 4633.

If we notice that $118^2 \equiv 25 = 5^2 \bmod 4633$,
then a = gcd(118+5, 4633) = gcd(123, 4633) = 41
and we have 4633 = (41)(113).

Factor Bases

The factoring problem is then reduced to finding a congruence of this type. To manufacture such a congruence we use the concept of a factor base. A ***factor base*** is simply a set of small primes which is not too big. If B is a factor base, then a number all of whose prime factors lie in B is said to be ***B-smooth***. To find a congruence of the form $t^2 \equiv s^2 \bmod n$, we first find several numbers b_i so that $(b_i)^2$ reduced mod n is B-smooth for a fixed factor base B. Since |B| is small, there will be many repeated primes in the factorizations of these numbers. The next task would be to find some subset of the $(b_i)^2$'s so that all the primes that appear in the product of these $(b_i)^2$'s appear to an even power (so, the product will be a square mod n).

Example

Factor n = 2043221 using the factor base B = {2,3,5,7,11}. We find, by means to be discussed below, the following B-smooth squares:

$$1439^2 \bmod 2043221 = 27500 = 2^2\, 5^4\, 11$$
$$2878^2 \bmod 2043221 = 110000 = 2^4\, 5^4\, 11$$
$$3197^2 \bmod 2043221 = 4704 = 2^5 3\, 7^2$$
$$3199^2 \bmod 2043221 = 17496 = 2^3\, 3^7$$
$$3253^2 \bmod 2043221 = 365904 = 2^4\, 3^3\, 7\, 11^2$$

Consider the 3rd and 4th numbers; we see that $[(3197)(3199)]^2 \equiv 2^8\, 3^8\, 7^2 \bmod 2043221$. Thus, t = (3197)(3199) mod 2043221 = 11098 and $s = 2^4\, 3^4\, 7 \bmod 2043221 = 9072$. Now gcd(t+s,n) = gcd(11098+9072, 2043221) = 2017 and we have 2043221 = (2017)(1013).

Example

This example also illustrates what can go wrong with the procedure. Had we taken the first two numbers, we would have obtained $[(1439)(2878)]^2 \equiv 2^6\ 5^8\ 11^2 \bmod n$, so $t = (1439)(2878) \bmod n = 55000$ and $s = 2^3\ 5^4\ 11 \bmod n = 55000$, i.e. $t = s$, and this does not lead to a factorization.

Linear Algebra

In the example we found the appropriate subset of b_i's to multiply by inspection, but we can do this systematically and at the same time answer the question of how many b_i's do we need to find? We form a 0-1 matrix where each row corresponds to one of the B-smooth squares, having |B| columns, each column corresponding to one prime in the factor base B. For each row the entry in the jth column is a 1 if the jth prime of B appears to an odd power and 0 otherwise. For the last example this matrix would look like:

0 0 0 0 1	$1439^2 \bmod n = 2^2\, 5^4\, 11$
0 0 0 0 1	$2878^2 \bmod n = 2^4\, 5^4\, 11$
1 1 0 0 0	$3197^2 \bmod n = 2^5 3\, 7^2$
1 1 0 0 0	$3199^2 \bmod n = 2^3\, 3^7$
0 1 0 1 0.	$3253^2 \bmod n = 2^4\, 3^3\, 7\, 11^2$

Linear Algebra

We are seeking sets of rows whose sum mod 2 is the zero row, i.e., we are finding a set of linearly dependent rows of this matrix when it is thought of as a matrix over GF (2). If this can't be done by inspection, we can always use the linear algebra technique of row reduction to find such sets. If there are |B| + 1 rows, then we are guaranteed to find at least one set of linearly dependent rows. This means that we can always find a congruence of the form $t^2 \equiv s^2 \bmod n$, however there is no guarantee that $t \neq \pm s \bmod n$. When this occurs, we can either use another set of linearly dependent rows (often requiring the finding of new B-smooth squares) or change the factor base.

Factor Bases

It should now be clear why we are a little vague in the definition of a factor base. If the factor base is small, we will need to find only a few B-smooth squares to get a linear dependency, however, having a small factor base means that the B-smooth squares are rare and so finding them will be hard. On the other hand, a large factor base means that there are many more B-smooth squares, so they will be easier to find, but we will then need to find many more of them. A good algorithm based on these considerations would therefore be one for which the factor base is not too big and which has an efficient way of finding B-smooth squares.

Quadratic Sieve

One could try randomly selecting the b_i and if n is not too large this will be effective, but for large n it isn't. A more effective procedure would be to select the b_i's to be integers near the square root of kn for different choices of k. The squares of these b_i's will be near kn, so, when reduced mod n they should be small and thus made up of only small primes. Another procedure, due to Pomerance, is to start with a large interval of integers around the square root of n, and then systematically remove integers based on a quadratic relationship with each prime in the factor base. The remaining integers have a high probability of being B-smooth. This method is known as the ***Quadratic Sieve***. A more recent algorithm, known as the ***Number Field Sieve*** finds the B-smooth squares by means of computations in rings of algebraic integers.

RSA Challenges

For factoring RSA moduli, the quadratic sieve has been the most successful algorithm. In April 1994, a 129-digit number known as RSA-129 was factored by Atkins, Graff, Lenstra and Leyland using the quadratic sieve. The numbers RSA-100, RSA-110, ..., RSA-500 were a list of RSA moduli publicized on the Internet (RSA Labs) as "challenge" numbers for factoring algorithms. Each number RSA-d was a d-digit number that is the product of two primes of approximately the same length. The numbers RSA-100, RSA-110, RSA-120, RSA-129, RSA-130, RSA-140, RSA-155 and RSA-160 have all been factored (the last of these on April 1, 2003).

New Challenge Numbers

In 2001, RSA Labs renamed and reissued the "challenge" numbers and assigned specific monetary rewards for their factoring. The new list (available at RSA Labs) uses the number of digits in the binary representation in the name, starting at RSA-576 (worth $10K) and going up to RSA-2048 ($200K).

Number Field Sieve

The number field sieve seems to have great potential since its asymptotic running time is faster than other known algorithms. It is still in the developmental stages, but many researchers feel that it might prove to be faster for numbers having more than about 125-130 digits. In 1990, the number field sieve was used by Lenstra, Lenstra, Manasse and Pollard to factor $2^{512} + 1$. On December 3, 2003 the factoring of RSA-576 (174 digits) was announced by a group at the German Federal Agency for Information Technology Security (BIS). They used a number field sieve to obtain the two 87-digit prime factors. The smallest challenge number is now RSA-640 worth $20K. (Factored on Nov. 2, 2005)

RSA Challenge Numbers

Challenge Number	Prize ($US)	Status	Submission Date	Submitter(s)
RSA-576	$10,000	Factored	Dec. 3, 2003	J. Franke et al.
RSA-640	$20,000	Factored	Nov. 2, 2005	F. Bahr et al.
RSA-704	$30,000	Not Factored		
RSA-768	$50,000	Not Factored		
RSA-896	$75,000	Not Factored		
RSA-1024	$100,000	Not Factored		
RSA-1536	$150,000	Not Factored		
RSA-2048	$200,000	Not Factored		

RSA - 640

RSA-640 Prize: $20,000 Status: Factored

Decimal Digits: 193

3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286782437916272838033415471073108501919548529007337724822783525742386454014691736602477652346609

Digit Sum: 806

Factors:

163473364580925384844313388386509085984178367003309231218111085238933310010450815121211816751157 9

and

190087128166482211312685157393541397547189678996851549366663853908802710380210449895719126146557 1

RSA -2048

RSA-2048 Prize: $200,000 Status: Not Factored

Decimal Digits: 617

25195908475657893494027183240048398571429282126204
03202777713783604366202070759555626401852588078440
69182906412495150821892985591491761845028084891200
72844992687392807287776735971418347270261896375014
97182469116507761337985909570009733045974880842840
17974291006424586918171951187461215151726546322822
16869987549182422433637259085141865462043576798423
38718477444792073993423658482382428119816381501067
48104516603773060562016196762561338441436038339044
14952634432190114657544454178424020924616515723350
77870774981712577246796292638635637328991215483143
81678998850404453640235273819513786365643912120103
9712282212072035 7

Decimal Digit Sum: 2738