Chapter 9
Factorisation and Discrete Logarithms Using a Factor Base

February 15, 2010

9

The two intractable problems which are at the heart of public key cryptosystems, are the infeasibility of factorising large integers and of solving the discrete logarithm (DL) problem. In Chapter 8 some simple schemes for solving these problems were presented. In this chapter the more powerful (and complex) methods based on factor bases are presented.

9.1

It is shown below that the time (T) taken by these methods can be approximated by $ln\ T = c(ln\ n)^a(ln\ ln\ n)^{1-a}$, this is sometimes written as $ln\ T = L(c, a, n)$, where $a = \frac{1}{2}$ and c is 2, $\sqrt{2}$ or 1 depending on the method. (In the faster and even more complex Number Field Sieve $a = \frac{1}{3}$.)

These methods are known as "sub-exponential". If $a = 1$ we have $T = n^c$; the time depends exponentially on the absolute value of n. If $a = 0$ we have $T = (ln\ n)^c$; the time depends exponentially on the log, i.e. the length of n , i.e. much faster. For intermediate values of a we have a compromise: Faster than exponential but not as fast as ordinary arithmetic.

The existence of these sub-exponential techniques is a weakness in many crypto systems based on modular arithmetic. One of the attractions of using Elliptic Curves (EC) techniques for cryptographic algorithms is that there are no known sub-exponential methods of attack.(See Chapter 10).

9.2 Factorising Large Integers

(We assume that n, the integer to be factorised, is tested in advance to ensure that it is not a prime, e.g. using Rabin's test.)

Most factorising algorithms - and certainly the most important ones which can be applied to very large integers - aim to find integer solutions $x,\ y$ (with $x,\ y < n$) to the equation

$$\mathrm{x}^2 = y^2 \ mod\ n \tag{0.1}$$

Since this can be written $(x+y)(x-y) = 0\ (mod\ n)$ we can then factorise n by finding $HCF((x-y), n)$ using the Euclidean algorithm. ($(x+y) = n$ and $x = y$ are obviously "bad" solutions to equation **??** and are discarded if they arise.) The method used to solve equation **??** involves creating a series of congruences

$$\mathrm{x(i)}^2\ mod\ n = y(i) = \prod(p(j)^{e(i,j)}) < n \tag{0.2}$$

where the product is over $j = 1,\ m$ and $p(j)$ stands for the j^{th} prime number. That is, $y(i)$ is represented in terms of a *Factor Base* of m primes. If this can be done for a good number, M say, of $x(i)$, then the solutions can be "cobbled together" multiplicatively to produce a right-hand side that is all squares, thus

$$\prod(Over\ selection\ of\ x(i)^2) = \prod(over\ j = 1, m)(p(j)^{f(j)})^2\ mod\ n \tag{0.3}$$

for some $f(j)$. Equation **??** has the required form of equation **??**. It is clear that constructing the "selection" is equivalent to raising each $y(i)$ to a power $c(i) = 0\ or\ 1$ and solving

$$\sum(over\ i)(c(i).e(i,j)) = 2f(j) = 0\ mod\ 2$$

for $c(i)$, with $j = 1, m$. For M greater then m this can always be done; but to avoid "bad" solutions it may be necessary to use $M = 2m$, say. Therefore the time taken by the procedure is determined by m, the size of the factor base, which in turn controls the probability of finding the congruences (equation

??). If m is small the probability of finding a relatively random $y(i)$ which can be expressed as in equation **??** is very small. (Remember that the probability of an arbitrary integer, X, *not* having a prime factor greater than its square root is about 0.3; while the probability of its *not* having a prime factor greater then $X^{\frac{1}{9}}$ is about 10^{-9}.) But make m large and the factor base is large and the calculations become very lengthy. Somewhere in between is an optimum value for m.

Differing techniques are used by the various factorising methods based on equation **??** to generate the congruence (see equation **??**). The essence of these differences is the way in which each tries to ensure that $y(i)$, while still relatively random, is confined within some range significantly smaller than $(0, n)$, so that the probability of equation **??** holding is not too small.

9.3 Factorising - Dixon's Random Method

An understanding of the issues raised above can be helped by considering the situation where the $x(i)$ are chosen randomly, so that the $y(i)$ are also random in $(0, n)$. The reasoning which follows is heuristic and has no pretensions to rigour.

The number of integers less than n which have prime factors less than or equal to $p(m)$, N say, is the numbers of solutions $n(j)$ to the inequality

$$\sum_{j=1}^{m}(n(j).ln(p(j))) \leq ln\ n$$

Therefore, replacing $p(j)$, $j = 1\ to\ m$, by the larger $p(m)$, we get $N \geq Number\ of\ integer\ solutions\ to$:

$$\sum(n(j)) \leq (\frac{ln\ n}{ln\ p(m)}) = r, \text{ say}$$

Using ordinary combinatorial arithmetic we get $N \geq \frac{(m+r)!}{m!r!}$, a binomial coefficient, which is greater than $\frac{(m^r)}{r!}$. In other words, the probability $P(n, m)$ that an integer lass than n can be factorised using the first m primes only is given by

$$P(n, m) \geq \frac{(m^r)}{n.r!} = \frac{1}{r!}$$

since $r = \frac{ln\ n}{ln\ m}$, if we take $p(m) = m(ln\ m)$, so that $ln\ (p(m)) = ln\ m$ approximately.

The random $y(i)$ are "sieved" (trial divisions by the first m primes) to test if they can be factorised solely in terms of those primes. Each such sieving takes a time of the order of $(m.(ln\ n))$. A selection is then made from the M equations **??** to form equation **??**. In modulo-2 arithmetic the time taken is

insignificant. In the worst case, when n is the product of only two primes, so that each quadratic residue has four roots of which two are "bad", there is a probability of failure to produce a satisfactory y equal to 0.5. k repeated attempts with different selections reduce this probability to 2^{-k}. Thus we may take $M = Order\ (m)$. With this approximation the central part of the factorising process takes a time, T, allowing for the probable failure of the sieving operation:

$$\mathrm{T}=(\mathrm{m}^2)(\tfrac{ln\ n}{P(n,m)}) \tag{0.4}$$

Putting $r! = r^r$, and making some other gross approximations, e.g. ignoring terms in $(ln\ ln\ n)$ in comparison with those in $(ln\ n)(ln\ ln\ n)$, on taking logarithms, we find

$$\ln \mathrm{T} \leq\ 2x+(ln\ n)\tfrac{ln\ ln\ n-ln\ ln\ m}{x} \tag{0.5}$$

where $x = ln\ m$. We find a minimum for $ln\ T$ with respect to x in two stages. Firstly, we shall ignore the $(ln\ ln\ m)$ term and find we need $ln\ m = ((ln\ n)\frac{(ln\ ln\ n)}{2})^{0.5}$. Then we substitute $\theta(ln\ m)$ into equation **??** and minimise with respect to θ to get $\theta = (\frac{1}{2})^{0.5}$ and

$$\ln \mathrm{m} = ((\ln \mathrm{n})(\ln \ln \mathrm{n})/4)^{0.5} \tag{0.6}$$

Substituting in equation **??** gives

$$\ln \mathrm{T} \leq\ (4(ln\ n)(ln\ ln\ n))^{0.5} \tag{0.7}$$

Equation **??** shows that the randomising process of equation **??** enables us to use successfully a factor base with value of m very much smaller than the number of primes less than n, $\pi(n)$, necessary for guaranteed successful sieving; since $ln\ \pi(n) = ln\ n - ln\ ln\ n$. Obviously if we can find a method of confining the $y(i)$ to a range less than $(0, n)$, for a given value of m we would have a much higher probability of success in satisfying equation **??**. Conversely, we could reduce m significantly and still maintain the probability of success achieved with Dixon's random method.

9.4 Factorising - The method of Continued Fractions (CF)

In this method, $x(i)$ of equation **??** is the integer numerator $A(i)$ of a rational number $\frac{A(i)}{B(i)}$, representing the i^{th} continued fraction approximation to the quadratic irrational $(k.n)^{0.5}$. Successive pairs of such values $A(i),\ B(i)$ are generated by a standard continued fraction algorithm, which in fact can be simplified to omit the calculation of the unwanted $B(i)$. Here k is a convenient integer constant, which when suitably and mysteriously chosen gives a better success rate for factorisation over the factor base.

Because $\frac{A(i)}{B(i)}$ is a good approximation to $(kn)^{0.5}$ it can be shown that $x(i)^2 = A(i)^2 = e.(2(kn)^{0.5}))\ mod\ kn$, where $e < 1$. In the case when $k = 1$ we have $x^2\ mod\ < 2n^{0.5}$. In turn this means that if we repeat the approximate analysis of the last section, we use $r = \frac{ln\ n}{2.ln\ m}$, and equation **??** now becomes

$$\ln \mathrm{T} \leq 2x + (ln\ n)^{\frac{ln\ ln\ n - ln\ ln\ m}{2x}} \tag{0.8}$$

giving for the new equation **??** $ln\ m = ((ln\ n)(ln\ ln\ n)/(8))^{\frac{1}{2}}$ and for the new equation **??** $ln\ T \leq (2(ln\ n)(ln\ ln\ n))^{0.5}$

In short, because the continued fractions produce good approximations to $n^{0.5}$, the range of $x^2(mod\ n)$ is significantly reduced, m can be reduced, and $ln\ T$ is reduced by a factor $2^{0.5}$.

The CF algorithm runs as follows:

If $(kn)^{0.5}$ is approximated by a continued fraction $(q(0),\ q(1),\ \ldots,\ q(m)) = \frac{A(m)}{B(m)}$, with the q's integers, then the usual recurrence relation $A(m) = q(m).A(m-1) + A(m-2)$, enables $A(m)$ to be found. $q(m)$ itself may be found from

$$\begin{aligned} & \mathrm{q(m)} = \lfloor (q(0) + \tfrac{\gamma(m)}{\delta(m)} \rfloor \\ with\quad & \gamma(m) = q(m-1)\delta(m-1) - \gamma(m-1) \\ and\quad & \delta(m) - \delta(m-2) = q(m-1)(\gamma(m-1) - \gamma(m)) \end{aligned}$$

Using these auxiliary quantities $\gamma,\ \delta$ it can be shown that $A(m)^2\ mod\ n = \delta(m+1).((-1)^{m+1})$ so that $A(m)^2\ mod\ n$ may be found directly by iterating the above three equations. The initial values are $q(0) = \lfloor (kn)^{0.5} \rfloor,\ \gamma(0) = 0,\ \gamma(1) = q(0),\ \delta(0) = 1,\ \delta(1) = kn - q(0)^2$.

9.5 Factorising - The Quadratic Sieve (QS)

In Dixon's method the $x(i)$ of equation **??** is chosen randomly giving $y(i)$ of *Order* (n). In CF the $x(i)$ is a CF numerator giving rise to a $y(i)$ of *Order* $(n^{0.5})$. In QS the $x(i)$ are successive values $(n' + t)$ where $n' = \lfloor n^{0.5} \rfloor$, the integral part of $n^{0.5}$, and t is systemically incremented and decremented. This gives rise to $y(i) = ((n' + t)^2 - n)$, so that it has negative as well as positive "small" values, also of *Order* $(n^{0.5})$. We may except QS to take the same as CF.

However, the systematic movement of t in the search for $y(i)$ which factorise completely over the factor base, allows for a very efficient calculating procedure, and QS is in fact faster than CF, particularly for large n.

The procedure uses logarithms. A very large table is built and is initialised with a block of the $log\ (y(i))$ corresponding to successive values of t.

For each prime p in the factor base we do the following:

1. Solve $(n\prime + z)^2 - n = 0\ mod\ p$ for $z1$ and $z2$.
2. $z1$ is an index into the table and points to a $y(i)$ divisible by p, so subtract $log\ p$ from that entry, and also from all entries with index $(z1 + kp)$ for positive and negative k up and down the block; since those entries too are divisible by p.
3. Repeat step two using $z2$.

When all p in the factor base have been tried, those entries in the table which are zero (or nearly) are factorisable over the base. Save the corresponding $x(i) = (n' + t)$ and the prime factors of $y(i)$.

Process the next adjacent block of $y(i)$, initialising the table with new $log\ (y(i))$, and remembering the appropriate starting index for each p, from the last block; etc.

Special cases, such as when $y(i)$ has repeated prime factor, when 2 is a prime factor, and when $y(i)$ factorises over the base except for a residual single large prime, can be accommodated.

In the QS method because of the sieving procedure, which uses deterministic logarithmic subtraction rather than trial division, we may perhaps take the times per values sieved as $(ln\ m)$, rather the $m(ln\ n)$.

Thus $T = m(\frac{ln\ m}{P(n,m)})$ and $ln\ T \leq x + (ln\ n)\frac{ln\ ln\ n - ln\ ln\ m}{2x}$ instead of equation **??**. Carrying out the same minimisation procedure we get

$$\begin{aligned} & \ln m=((\ln n)(\ln \ln n)/4)^{0.5} \\ and \quad & \ln T \leq ((ln\ n)(ln\ ln\ n))^{0.5} \end{aligned}$$

QS is the fastest of the sub-exponential sieving techniques with $ln\ T = c((ln\ n)(ln\ ln\ n))^{0.5}$ because of this systematic approach to the sieving. However the Number Field Sieve (NFS) technique is substantially faster for really large n.

9.6 Finding Discrete Logarithms

The best algorithms for finding 'normal' discrete logarithms also employ a factor base of prime numbers, as used for factorisation. (It is the use of this factor base, and the multiplying together of elements expressed in terms of this factor base, which enables sub-exponential algorithms to be found. Conversely, DL-based crypto-systems based on elliptic curves or Lucas series, which do not permit this factor base representation, and for which multiplication of elements does not exist or is not 'closed', can not be attacked with sub-exponential algorithms.)

The 'normal' discrete logarithm problem is: Find x, given $a,\ b$ and p in the equation

$$a^x = b\ mod\ p \tag{0.9}$$

For simplicity we take p to be a prime.

The basic algorithm for solving for x has three stages. In the first two stages the discrete logarithms $N(i)$ (to the base a) of the members $p(i)$ of the factor base are calculated. In the last stage the discrete logarithm of b is found.

Stage 1
We pick random $n(i)\ <\ p-1$ and try to express $a^{n(i)} mod\ p$ in terms of the factor base of the first m primes until we get $M\ \geq\ m$ successes:

$$a^{n(i)}\ mod\ p = \prod_{j=1}^{m} p(j)^{e(i,j)} \qquad i = 1, M \tag{0.10}$$

This is the same procedure as the central part of Dixon's method for factorisation except that we use $a^{n(i)}$ instead of $x(i)^2$. Consequently the time $T1$

taken by stage one can be expressed using equation **??** as:

$$\ln \mathrm{T1} = (4(\ln \mathrm{p})(\ln \ln \mathrm{p}))^{0.5} \tag{0.11}$$

Note that this is the minimum time occurring when $ln\ m = ((ln\ p)(ln\ ln\ p)/(4))^{\frac{1}{2}}$, as per equation **??**.

Stage 2
We suppose $p(j) = a^{N(j)}$ for some $N(j)$. (This is certainly true if a is primitive, if a is not primitive see below). Substituting in equation **??** and taking logarithms gives

$$\mathrm{n(i)}=\textstyle\sum_{j=1}^{m} N(j).e(i,\ j)\ mod\ (p-1) \tag{0.12}$$

These linear equations are then inverted to find the $N(j)$. The inversion is performed modulo each of the prime factors of $(p-1)$, and the results are put together using the Chinese Remainder Theorem. The time taken is $T2$ and is given by

$$\ln \mathrm{T2} = 3(\ln \mathrm{m}) \tag{0.13}$$

No proper minimum exists for $T2$, we want m as small as possible. But if we take Stage 1 minimum value for m, given by equation **??** we find

$$\ln \mathrm{T2} = (2.25(\ln \mathrm{p})(\ln \ln \mathrm{p}))^{0.5} \tag{0.14}$$

Stage 3
We now have from Stage 2 $p(j) = a^{N(j)}$, so we take an arbitrary s and try to express $b.a^s mod\ p$ in terms of the factor base. We try sufficient values for s until we have success:

$$\text{b.a}^s mod\ p = \prod_{j=1}^{m} p(j)^{e'(j)} = Product\ a^{N(j).e'(j)} \tag{0.15}$$

Taking logarithms we have

$$\text{x+s} = \sum_{j=1}^{m} N(j).e'(j)\ mod\ (p-1) \tag{0.16}$$

which gives us x. The time taken for Stage 3 is $T3$ (representation of a single quantity in terms of the factor base instead of m quantities as in equation **??**) which is given by $T3 = m\frac{m(ln\ p)}{P(p,\ m)}$.

Using the Stage 1 minimum value for m we find

$$\text{ln T3} = (2.25(\text{ln p})(\text{ln ln p}))^{0.5} \tag{0.17}$$

which is the same as in $ln\ T2$ (see equation **??**). Thus the (non-minimum) values for $T2$ and $T3$ are less than the minimum value for $T1$ given by equation **??**; so the total time, T, for all three stages is determined by Stage 1 and we may say

$$\text{ln T= } ((4)(\text{ln p})(\text{ln ln p}))^{0.5} \tag{0.18}$$

Equation **??** shows that there is a sub-exponential solution to the basic DL problem, which takes the same time as Dixon's factorisation essentially because it also uses a random approach (the $n(i)$). A very similar but alternative approach to Stage 3 (known as "Index Calculus") raises b to the arbitrary power s, and tries to express this in terms of the factor base:

$$\begin{array}{lll} & \text{b}^s mod\ p & = \prod_{j=1}^{m} p(j)^{e''(j)} = \prod a^{N(j).e''(j)} \\ then & \text{s.(log b)} & = \sum_{j=1}^{m} (N(j).e''(j))\ mod\ (p-1) \end{array}$$

so $(log\ b) = x$ can be found if s has an inverse $s^{-1} mod\ (p-1)$.

If a is *not* primitive we find c primitive $mod\ p$ and use the method to find r in $a = c^r mod\ p$. Then the problem $p_j = a^{N_j} mod\ p$ becomes $p_j = c^{N'_j} mod\ p$ with $N'_j = rN_j$. Solve for N'_j and then find $N_j = r^{-1}N'_j mod\ (p-1)$.

9.7 Coppersmith's Method for DLs

A more efficient method of solving the DL problem has been presented by Coppersmith. The method sieves integers of *Order* ($p^{0.5}$), with a linear logarithmic sieve as in QS, and the net result is that the run-time and the size of the factor base are those of the QS factorisation program, namely:

$$\ln m = ((\ln p)(\ln \ln p)/4\ )^{\ 0.5} \tag{0.19}$$

$$\ln T1 = 2(\ln m) \tag{0.20}$$

This is to be compared with $ln\ T1 = 4(ln\ m)$ which we had before. As a result the matrix inversion of Stage 2 (with $ln\ T2 = 3(ln\ m)$) is now the slowest part of the program. Using techniques, such as those due to Cornelius Lanczos or Wiedemann, for solving linear equations with sparse matrices enables the overall time to be reduced to being proportional to m^2, with m given by equation **??**.

References The best introductory reference is Knuth's "Semi-numerical Algorithm" Volume 2 of "The Art of Computer Programming". The publisher is Addison Wesley. Look in the index. It also contains referrences to other more detailed publications.

For a general introduction to continued fractions see Davenport's book "The Higher Arithmetic". Cambridge University Press.

For the quadratic sieve see "Factoring Large Numbers with a Quadratic Sieve" by Joseph L. Gerver, Mathematics of Computation, Volume 41, Number 163, July 1983, Pages 287-294.

For the discrete logarithm see "Discrete Logarithms in GF(p)" Don Coppersmith et al., Algorithmica, (1986) Vol 1:1-15.

Example Factorising Using Quadratic Sieve (Manual)
Factorise 1769 = n; $\lfloor\sqrt{n}\rfloor = 42$; Factor Base = 2, 3, 5, 7, 11, 13

$$
\begin{array}{lcrcll}
36^2 & = & -473 & = & \text{-}11 \times 43 & NF\ (not\ factorisable) \\
37^2 & = & -400 & = & \text{-}2^4 \times 5^2 & \\
38^2 & = & -325 & = & \text{-}5^2 \times 13 & \\
39^2 & = & -248 & = & \text{-}2^3 \times 31 & NF \\
40^2 & = & -169 & = & \text{-}13^2 & \\
41^2 & = & -88 & = & \text{-}2^3 \times 11 & \\
42^2 & = & -5 & & & \\
43^2 & = & 80 & = & 2^4 \times 5 & \\
44^2 & = & 167 & & & NF \\
45^2 & = & 256 & = & 2^8 & \\
46^2 & = & 347 & & & NF \\
47^2 & = & 440 & = & 2^3 \times 5 \times 11 &
\end{array}
$$

First Try

$$
\begin{aligned}
41^2 \times 42^2 \times 47^2 &= 2^6 \times 5^2 \times 11^2 \ mod\ 1769 \\
or\ (1329)^2 &= (440)^2 \ mod\ 1769 \\
or\ (1769)\times 0 &= 0 \bmod 1769
\end{aligned}
$$

Second Try

$$
\begin{aligned}
37^2 \times 40^2 &= 2^4 \times 5^2 \times 13^2 \\
(1480)^2 &= (260)^2 \\
(1480 - 260)\times(1480 + 260) &= 0 \bmod 1769 \\
1740\times 1220 &= 0 \bmod 1769 \\
2^4 \times 3 \times 5^2 \times 29 \times 61 &= 0 \bmod 1769 \\
1769 &= 29\times 61
\end{aligned}
$$