THE QUADRATIC SIEVE FACTORING ALGORITHM

by

Carl POMERANCE*

Department of Mathematics

University of Georgia

Athens, Georgia 30602 USA

The quadratic sieve algorithm is currently the method of choice to factor very large composite numbers with no small factors. In the hands of the Sandia National Laboratories team of James Davis and Diane Holdridge, it has held the record for the largest hard number factored since mid-1983. As of this writing, the largest number it has cracked is the 71 digit number $(10^{71}-1)/9$, taking 9.5 hours on the Cray XMP computer at Los Alamos, New Mexico. In this paper I shall give some of the history of this algorithm and also describe some of the improvements that have been suggested for it.

KRAITCHIK'S SCHEME

There is a large class of factoring algorithms that share a common strategy. If N is the number to be factored, then the idea is to multiply congruences $U \equiv V \bmod N$, where $U \not\equiv V$ and complete or partial factorizations (depending on the algorithm) have been obtained for U and V, so as to produce a special congruence $X^2 \equiv Y^2 \bmod N$. Then one stands a good chance that the greatest common factor $(X-Y, N)$, found by Euclid's algorithm, is a non-trivial factor of N. If it is not, then another combination of congruences can be tried. Thus these algorithms have several parts :

(1) Generation of the congruences $U \equiv V \bmod N$,

(2) Determination of the complete or partial factorizations of U and V for some of the congruences,

(3) Determination of a subset of the factored congruences which can be multiplied to produce a special congruence $X^2 \equiv Y^2 \bmod N$,

(4) Computation of $(X-Y, N)$.

* supported in part by a grant from the National Science Foundation.

T. Beth, N. Cot, and I. Ingemarsson (Eds.): Advances in Cryptology - EUROCRYPT '84, LNCS 209, pp. 169-182, 1985.
© Springer-Verlag Berlin Heidelberg 1985

For example, say we try to factor $N = 91$ and we notice that

$$81 \equiv -10, \quad 90 \equiv -1, \quad 75 \equiv -16, \text{ and } 64 \equiv -27.$$

Factoring these numbers completely we have

$$3^4 \equiv -2 \cdot 5, \quad 2 \cdot 3^2 \cdot 5 \equiv -1, \quad 3 \cdot 5^2 \equiv -2^4, \text{ and } 2^6 \equiv -3^3.$$

Multiplying the last two congruences , we have

$$2^6 \cdot 3 \cdot 5^2 \equiv 2^4 \cdot 3^3 ,$$

or cancelling common factors,

$$2^2 \cdot 5^2 \equiv 3^2.$$

This gives $10^2 \equiv 3^2 \bmod 91$ and $7 = (10-3, 91)$. Or we might have multiplied the first two congruences, getting

$$2 \cdot 3^6 \cdot 5 \equiv 2 \cdot 5 \longrightarrow 3^6 \equiv 1,$$

so $27^2 \equiv 1^2 \bmod 91$ and $13 = (27-1, 91)$.

This general scheme for factoring was published by Kraitchik [4] in 1926. The numbers U, V are factored into primes except for squared factors. Since most of the congruences one is likely to generate will not successfully factor in step (2), one's chances are enhanced if one of U, V is arranged to be a square and the other has a large square factor. In [5], pp. 26-27, Kraitchik explains how this should be done. He lets $U = x^2$ where x is carefully chosen so that $V = -N + x^2$ has a large factor y^2. He can force y^2 to appear by choosing x as a solution of the quadratic congruence $x^2 \equiv N \bmod y^2$. However, V/y^2 need not be small and so easily factorable. This method has its problems.

Kraitchik opportunistically used other congruences $U \equiv V \bmod N$ that were suggested by the special form of N in question . These congruences would not be available for a "random" N. In his later work [5], the congruences $U \equiv V \bmod N$ were used to assist in finding X and Y with $X^2 - Y^2 = N$. This is an old factoring strategy that goes back to Fermat. I think Kraitchik preferred this method for two reasons. First, fewer congruences $U \equiv V \bmod N$ with multiplicative information about U and V are used. Second, when X, Y are found with $X^2 - Y^2 = N$, one could be assured of a non-trivial factorization of N, unlike with the other method where step (4) may produce a trivial factorization. Little did Kraitchik know that his largely abandoned method of producing "cycles" (the combination of congruences in step (3)) would be the basis of most modern factoring algorithms !

THE CONTINUED FRACTION ALGORITHM

Instead of finding $U \equiv V \bmod N$ with one of U,V a square and the other divisible by a large square factor, another strategy might be to choose one a square and the other *small* in absolute value. It thus would more likely factor in step (2). In 1931, Lehmer and Powers [6] suggested the use of the continued fraction expansion of $\sqrt{N}$ to generate the congruences $U \equiv V \bmod N$ in Kraitchik's scheme. This is done by a simple recursive procedure that creates pairs Q_n, A_n where

$$(1) \qquad Q_n \equiv A_n^2 \bmod N$$

and $|Q_n| < 2\sqrt{N}$. An old method of Legendre also suggested the use of the continued fraction expansion of $\sqrt{N}$, but his aim was to use the congruences (1) to find information on the quadratic character mod Q_n of prime factors p of N. Then a direct search, such as trial division, could be greatly speeded up because many potential divisors would not have the proper character. In contrast, Lehmer and Powers advocated multiplying several congruences of the form (1) to produce congruent squares.

Morrison and Brillhart [10] were the first to try the continued fraction algorithm on a modern computer. In the implementation they made several major improvements and refinements that would be of use in any of the combination of congruences family of algorithms. First, they used a "factor base", or all of the primes to some point F, to dermine which of the congruences (1) were useful. When a congruence (1) was generated, the number Q_n was subjected to trial division by the primes $p \leq F$. If a complete factorization could be obtained, the congruence was kept for later use -if not, it was discarded.

Step (3) of the algorithm, the actual combination of congruences was effected by a Gaussian elimination in a very large matrix over $\mathbf{Z}/2\mathbf{Z}$. Specifically, if the factor base consists of the primes $p_1,\ldots,p_f$, and if

$$Q_n = (-1)^{a_0} \prod_{i=1}^{f} p_i^{a_i}$$

where the a_i are non-negative integers, then we look at the vector

$$\vec{v}(n) = (a_0,a_1,\ldots,a_f) \bmod 2.$$

If we have enough vectors $\vec{v}(n)$, then Gaussian elimination will produce a linear dependency

$$\vec{v}(n_1) + \ldots + \vec{v}(n_k) = \vec{0},$$

so that $Q_{n_1} \ldots Q_{n_k}$ is a square, say X^2. If we compute $X \bmod N$ and $Y = A_{n_1} \ldots A_{n_k} \bmod N$, then $X^2 \equiv Y^2 \bmod N$ and we are ready for step (4).

Another improvement, called the "early abort strategy" was described in [11]. This improvement extended the useful range of the continued fraction algorithm on an ordinary main frame computer by about 10 digits -from the mid 40's to the mid 50's (see [14], [12]).

A special purpose, low cost processor has been designed by J.W. Smith and S.S. Wagstaff, Jr. and built at the University of Georgia to implement the continued fraction algorithm with the early abort strategy. It is designed to do the trial division step on a Q_n in parallel (several trial divisors can be tried at once) and the device has extended precision, so that this arithmetic done with long integers can be done in single precision. It should be fully operational soon and we await their results. It will probably be somewhat inferior to the results produced by the Sandia team, butthis should be weighed by the fact that the cost of the Smith-Wagstaff device is about three orders of magnitude less than the cost of a Cray X M P.

THE MILLER -WESTERN ALGORITHM

The issue of Mathematics of Computation which contains the Morrison-Brillhart paper is dedicated to D.H. Lehmer and has many interesting articles on computational number theory. In this issue there is an article by J.C.P. Miller [7] on factoring that also uses congruences $U \equiv V \bmod N$. He attributes the idea to A.E. Western. The aim is to find congruences with U and V completely factored. But rather than combine these congruences to produce congruent squares, each congruence is read as a linear relation of indices with respect to some primitive root g of p, where p is a prime factor of N. When enough congruences can be found there is a chance of finding p via created congruences of the form $a^t \equiv 1 \bmod N$. If some $q|t$ can be found with $a^{t/q} \not\equiv 1 \bmod N$, then perhaps $(a^{t/q}-1, N)$ is a non-trivial factor of N.

I see no particular advantage to this method over just combining the factored congruences to produce congruent squares in the Kraitchik scheme. I mention the algorithm here because of the very simple way Miller chooses the congruences $U \equiv V \bmod N$. Namely he just partitions N as $A+B$, letting $U=A$, $V=-B$. There is an interesting unsolved problem of Erdös that says that for each $\varepsilon > 0$ there is an $N_0(\varepsilon)$ such that for each integer $N > N_0(\varepsilon)$ there is a partition of N as $A+B$ where no prime in $A\,B$ exceeds N^{ε}. What we need is an *algorithmic* solution of Erdös's problem that gives many such pairs A,B. Perhaps this problem (and factoring itself) is not so hard !

SCHROEPPEL'S ASYMPTOTIC ANALYSIS

In the late 1970's some important advances on factoring were made by Richard Schroeppel. He never published his results, but they have become known through copies of his letters and through second hand published accounts (e.g. [8], [11]). First, Schroeppel began the systematic study of the asymptotic running time of factorization algorithms in the Kraitchik family. Second, he found an algorithm in the family where step (2) could be accomplished without time consuming trial division.

Schroeppel's asymptotic analysis hinged on the optimal choice of the parameter F, the upper bound for the primes in the factor base. A small choice of F means only few factored congruences are necessary to produce a linear dependency, but such congruences are very hard to find. With a large choice of F the situation is reversed. Somewhere between "large" and "small" is the optimal choice. Schroeppel realized that to study this situation asymptotically one needed to use the function $\psi(x,y)$ -the number of integers up to x divisible by no prime exceeding y. Specifically this was needed with x being the average size of the residues being trial divided and $y = F$. Thus $\psi(x,y)/x$ represents the "probability" that a residue will completely factor over the factor base.

For example, suppose we study the continued fraction algorithm. Then the typical Q_n will be approximately $\sqrt{N}$. Further, if f is the number of primes in the factor base, then we should have $f \approx F/2 \log F$ (only those odd primes p with $(N/p) = 1$ can divide a Q_n). We need to obtain about f completely factored Q_n's. Thus we should expect to have to generate

$$f(\psi(\sqrt{N},F)/\sqrt{N})^{-1} = f\sqrt{N}/\psi(\sqrt{N},F)$$

values of Q_n before enough factored ones are found. More, we need to do about f trial divisions on the average Q_n produced, so the total number of trial division steps needed to factor N with the continued fraction algorithm should be about

$$f^2\sqrt{N}/\psi(\sqrt{N},F).$$

Ignoring other steps in the algorithm, we thus choose F so as to minimize this quantity. Schroeppel assumed that

$$\psi(x,x^{1/u})/x = u^{-(1+o(1))u} \quad \text{for} \quad (\log x)^{\varepsilon} < u < (\log x)^{1-\varepsilon}$$

(a result which was subsequently proved in [1]) and found that the optimal choice of F is $L(N)^{1/\sqrt{8}+o(1)}$ where

$$L(N) = \exp(\sqrt{\log N \log\log N})$$

(natural logs) and that the expected running time is $L(N)^{\sqrt{2}+o(1)}$. Of course, this argument is only *heuristic* -for one, it is assumed without proof that the numbers Q_n factor over the primes to F as frequently as random numbers of the same

approximate size.

SCHROEPPEL'S LINEAR SIEVE

Schroeppel's new algorithm with by-passed trial division is also in Kraitchik's family. Let

$$S(A,B) = (\lfloor\sqrt{N}\rfloor+A)(\lfloor\sqrt{N}\rfloor+B) - N$$
$$T(A,B) = (\lfloor\sqrt{N}\rfloor+A)(\lfloor\sqrt{N}\rfloor+B). \tag{2}$$

If $|A|$, $|B|$ are less than N^{ε}, then $|S(A,B)| \lesssim 2N^{1/2+\varepsilon}$ so that the $S(A,B)$ are relatively small, not much larger than the Q_n's given by (1). More, we evidently have

$$S(A,B) \equiv T(A,B) \bmod N$$

so that we use these as the congruences in Kraitchik's scheme. We attempt to completely factor the $S(A,B)$'s over a factor base, but we do not try to factor the $T(A,B)$'s. Note that (2) already gives a partial factorization of $T(A,B)$. We could thus arrange for a product of $T(A,B)$'s to be a square if each A and each B is used an even number of times in the product. Thus we treat the variables A,B as if they were primes in the Gaussian elimination step.

Thus the Gaussian elimination step is harder and the residues $S(A,B)$ are a bit larger than in the continued fraction algorithm. There is an advantage here, though, and it is that the numbers $S(A,B)$ can be factored *without* trial division. The idea is that for a fixed value A_o for A we can let B run over consecutive integers. These numbers form an arithmetic progression, so that if $p|S(A_o,B_o)$, then $p|S(A_o,B_o+p)$, $p|S(A_o,B_o+2p)$, etc. That is, we know beforehand exactly which values of B have $S(A_o,B)$ divisible by p. No more do we need to waste a trial division step on a number where the trial divisor does not go.

Schroeppel's asymptotic analysis suggested the running time of his algorithm was $L(N)^{1+o(1)}$. However, his analysis neglected the time for the Gaussian elimination. This is not a mistake in the continued fraction algorithm analysis because it really takes less time than the trial division step. But in Schroeppel's algorithm we have given the Gaussian elimination a larger task to accomplish and it can be shown (heuristically) that it takes $L(N)^{3/2+o(1)}$ steps, worse than the running time of the continued fraction algorithm.

THE QUADRATIC SIEVE

In 1981 I suggested taking $A=B$ in Schroeppel's linear sieve algorithm, calling the resulting method the quadratic sieve algorithm. This simple move changes things drastically. Let

$$Q(A) = S(A,A) = (\lfloor\sqrt{N}\rfloor+A)^2 - N. \tag{3}$$

Thus we are back in the game of producing quadratic residues as in the continued fraction algorithm, so the Gaussian elimination step should not be a major difficulty. In addition, we can still sieve as Schroeppel did. If $p|Q(A_o)$, then $p|Q(A_o+p)$, $p|Q(A_o+2p)$, etc. This property of the function $Q(A)$ follows from the fact that it is a polynomial with integer coefficients. Heuristically, the running time for the algorithm is $L(N)^{\sqrt{9/8}+o(1)}$, including the matrix step, an improvement over the continued fraction algorithm. This analysis and a description of the algorithm is found in [11].

The idea in (3) is to choose A with $|A| < N^{\varepsilon}$. Since for small A we have

$$Q(A) \approx 2A\sqrt{N},$$

We thus have $|Q(A)| \lesssim 2N^{1/2+\varepsilon}$, as with Schroeppel. It is amusing to note that the method (3) of choosing quadratic residues mod N is very similar to that of Kraitchik discussed above. There is a difference though. Kraitchik carefully prepared values of x so that x^2-N had a large square factor. In (3) we indiscriminately choose all values of x near $\sqrt{N}$.

The advantage is clear, because now we can use a sieve. For each odd prime p in the factor base (p is in the factor base if $(N/p)=1$) we solve the quadratic congruence

$$(\lfloor\sqrt{N}\rfloor+A)^2 \equiv N \bmod p,$$

labelling the solutions $A_1^{(p)}$, $A_2^{(p)}$ (for $p=2$, special treatment is required). We then compute very crude logs of each of the $Q(A)$ for A in a long interval (these logs are all approximately equal). These logs are stored in an array indexed by the values of A. We then pull out each log that has its index $A \equiv A_1^{(p)}$ or $A_2^{(p)} \bmod p$ and subtract log p from the number in the location. (Again, log p is a low precision log). This is done for each p in the factor base and for some of the higher powers of the smaller primes p. At the end, we scan the array for residual logs that are close to 0. These locations correspond to values of $Q(A)$ that completely factored. The number $Q(A)$ may now be computed and factored by trial division. Of course, very few numbers $Q(A)$ completely factor, so the amount of trial division in the algorithm is negligible. Note that not only does the quadratic sieve algorithm have asymptotically fewer steps than the continued fraction algorithm, but each step is simpler. In the quadratic sieve a typical step is a single precision subtraction , while in the continued fraction algorithm a typical step is a divide with remainder of a single precision integer into a long dividend.

Asymptotically, the algorithm of Schnorr and Lenstra [13] (which is not in the Kraitchik family) should be faster than the quadratic sieve : its heuristic run time is $L(N)^{1+o(1)}$. However it has not yet proved computer practical and the crossover point may be very large. A typical step in the Schnorr -Lenstra algorithm is composition of binary quadratic forms with multi-precision entries and finding a reduced form in the class.

THE DAVIS VARIATION

Davis and Holdridge [2] have written a very clear article on the implementation of the quadratic sieve algorithm and there is no need to duplicate their work here. But I would like to mention an important improvement Davis made on the method. It seems clear that the quadratic sieve algorithm majorizes the continued fraction algorithm in every respect but in the size of the quadratic residues. Namely, in the latter method, each $|Q_n|$ is less than $2\sqrt{N}$ but in the former, the numbers $|Q(A)|$ are about $N^{1/2+\varepsilon}$ (where $\varepsilon > 0$ is small and tends to 0 slowly as $N \to \infty$). Of course, the larger the residue, the less likely it is to factor over the factor base.

The Davis variation is simply to sieve over various arithmetic progressions of A's so that the $Q(A)$'s are guaranteed to have a fixed factor. Specifically, if p is some large prime *not* in the factor base and $p | Q(A_o)$ where $0 < A_o < p$, then p divides every $Q(A_o+Ap)$ as noted before. Let

$$Q_p(A) = Q(A_o+Ap).$$

Then

$$Q_p(A)/p \approx 2A\sqrt{N},$$

so that after the known factor p is divided out of $Q_p(A)$, the cofactor is about the same size as $Q(A)$. Thus instead of having just one polynomial to work with, we have a large family of polynomials -one (in fact, two) for each possible p. For each p used we consider p as a new prime in the factor base. Thus if k factored values of $Q_p(A)$ are found, after eliminating p we have $k-1$ vectors left over the original factor base. However, Davis avoids losing even one vector. He does this by finding a factored $Q_p(A)$ for "free". This magic is accomplished as follows. If in the original polynomial $Q(A)$ a location A_1 is found after sieving where the residual log is not near 0, but less than $2 \log F$, then the cofactor after $Q(A_1)$ is divided by all primes in the factor base is a prime p with $F < p < F^2$. We thus use this p to form $Q_p(A)$ (and we can choose $A_o \equiv A_1 \bmod p$). We start with one factored value before sieving the new polynomial, so any new factored values found are all to the good.

THE MONTGOMERY VARIATION

Independently of Davis, Peter Montgomery [9] has come up with another strategy for fighting the drift to infinity of the quadratic residues $Q(A)$. His method tailor makes polynomials to custom fit not only the number N to be factored, but the length of the interval we sieve over before we change polynomials.

Suppose we sieve over intervals of length $2M$ before we change polynomials. We are looking for polynomials

$$F(x) = ax^2+2bx+c \quad \text{where} \quad N \mid b^2-ac,$$

for then

$$(4) \qquad aF(x) = a^2X^2+2abx+ac = (ax+b)^2 - (b^2-ac)$$
$$\equiv (ax+b)^2 \bmod N.$$

Further, we would like the values of $F(x)$ to be small in absolute value on an interval of length $2M$. It thus seems reasonable to center this interval on the vertex of the parabola $F(x)$ -so we specify the interval as

$$I = (-b/a-M,\ -b/a+M)$$

and choose a,b,c so that

$$-F(-b/a) \approx F(-b/a-M) = F(-b/a+M).$$

To be specific, we choose a,b,c so that

$$(5) \qquad b^2-ac = N\ .$$

Then from (4),

$$-aF(-b/a) = N,\ aF(-b/a-M) = aF(-b/a+M) = a^2M^2-N.$$

Thus we should choose a so that $N \approx a^2M^2-N$, i.e.,

$$(6) \qquad a \approx \sqrt{2N}/M.$$

Montgomery suggests then that we decide first on $2M$, the length of the interval sieved. Next an integer a is chosen satisfying (6) and then integers b and c are found satisfying (5). (For example, we could choose a as a prime satisfying $(N/a) = 1$. Then the quadratic congruence $b^2 \equiv N \bmod a$ is solved for b and c is chosen as $(b^2-N)/a$).

We thus have constructed a quadratic polynomial $F(x)$ so that on the interval I

$$|F(x)| \lesssim \frac{1}{\sqrt{2}} M\sqrt{N}.$$

This is better than the polynomials $Q(A)$ and $Q_p(A)/p$. For them on the interval $(-M,M)$ their absolute values are bounded by $2M\sqrt{N}$. Thus the largest of Montgomery's residues are about $2\sqrt{2}$ times smaller and so somewhat more likely to factor over the factor base.

Here is an idea which should improve Montgomery's basic plan. If $k \geq 1$ values of $F(x)$ are found which factor over the factor base, we only end up with $k-1$ vectors because the factor a must be eliminated from the congruences (4). This could be serious if the expected value of k were much smaller than 1, for then in the rare instances we had $k > 0$, it would be likely that $k = 1$ and nothing would

be gained. To solve this problem, we choose $a = g^2$ where g is a prime with $(N/g) = 1$ and $g \approx \sqrt{\sqrt{2N}/M}$. Then everything is as before, but we do not have to eliminate a from (4) because it is a square. All factored values of $F(x)$ are now to the good.

The quadratic congruence

$$b^2 \equiv N \bmod g^2 \tag{7}$$

can be solved very simply if $g \equiv 3 \bmod 4$ and $(N/g) = 1$. Just take

$$b = N^{(g^2-g+2)/4} \bmod g^2.$$

This involves arithmetic mod g^2. Instead, by first solving (7) mod g by taking $b_1 \equiv N^{(g+1)/4} \bmod g$ and next determine x so that $(b_1+xg)^2 \equiv N \bmod g^2$, all of the arithmetic can be done mod g. (This idea was suggested by Wagstaff -it is an elementary application of Hensel's lemma).

Above we chose a satisfying (6) to minimize the maximum value of $|F(x)|$ on I. Instead, it may be more appropriate to minimize the *average* value of $|F(x)|$. For this we should choose

$$a \approx (1.5127453)\sqrt{N}/M.$$

However, it probably makes very little difference whether we choose a by this scheme or by (6).

In the implementation of Montgomery's variation (which has not yet been done) one should compute how costly it is to produce new polynomials $F(x)$. If it is very costly, a larger value should be chosen for M ; if it is not so costly, a smaller value should be chosen for M. That is, we should sieve over as *short* an interval as possible, where the overhead of producing new polynomials and computing the starting points for each prime used in the sieve says it should not be *too* short.

LARGE PRIME VARIATION

In [11] the large prime variation was suggested for the quadratic sieve. This variation is commonly used with the continued fraction algorithm. As mentioned above, if the residual log after sieving is not close to 0, but less than $2 \log F$, then we have produced a quadratic residue that completely factors over the factor base except for one large prime factor p with $F < p < F^2$. Not only do we receive this information for free, but such residues are simple to process. If the large prime p is never seen again in another factored residue, it is useless for us and this line may be discarded. If it appears k times, we can eliminate it, being left with $k-1$ vectors over the factor base. The "birthday paradox" suggests that the event $k \geq 2$ will not be that uncommon.

If this method is used together with the Davis variation, another method should be used to produce the polynomials $Q_p(A)$. We can instead use (7). Let $g > F$ be a prime with $g \equiv 3 \bmod 4$ and $(N/g) = 1$. If b is the solution of (7), we let $A_o = b - \lfloor \sqrt{N} \rfloor \bmod g^2$. Then we can use the polynomial

$$Q_{g^2}(A) = Q(A_o + g^2 A)$$

in the Davis variation. (We can also use $A_o \equiv -b - \lfloor \sqrt{N} \rfloor \bmod g^2$).
Every value factored over the factor base is useful and we can use the large prime variation on all of the $Q_{g^2}(A)$ for various choices of g^2. Note that there is less overhead with producing the polynomials $Q_{g^2}(A)$ than the $F(x)$ in Montgomery's variation because g can be chosen smaller with Davis.

SMALL MODULI

In trial division it takes just as long to test divisibility by 3 as by 101. But sieving by 3 takes 101/3 times *longer* than 101 since it has more frequent "hits". Thus a considerable percentage of sieving time is spent with the very smallest moduli. This seems a waste since these small moduli contribute the least information. One idea is to skip sieving with them completely. Say we do not sieve with any modulus below 30. Then if 3 is in the factor base, for example, we will not sieve mod 3, mod 9, nor mod 27. But we will sieve mod 81, subtracting $4 \log 3$ (instead of $\log 3$) at hits for this modulus. If P is the product of the highest powers of the moduli skipped and if $P < F$, then we lose nothing by this strategy. Indeed, the maximal error introduced in skipping the small moduli is at most $\log P < \log F$. Thus if the residual log is less than $\log F$ the number has factored completely and every completely factored number will have a residual log less than $\log F$.

If this idea proves good, one might "live dangerously" and let P be somewhat bigger than F. In fact if we let P be around F^2 and use the large prime variation too, the only residues lost will be some of the residues which factored with a large prime. Of course, you may prefer not to lose anything.

USE OF A MULTIPLIER

The factor base for N in the quadratic sieve algorithm consists of those primes $p \leq F$ with $p=2$ or $(N/p) = 1$. If we replace N by λN where λ is a small positive square-free integer (Kraitchik again -see [4], p. 208 and [5], Ch. 2) then the factor base changes. The expected contribution to $\log(x^2 - \lambda N)$ by the power of p in $x^2 - \lambda N$ is

$$E_p = (2 \log p)/(p-1)$$

if x is a random integer and $(\lambda N/p) = 1$. For $p=2$ the expected contribution is

$$E_2 = \begin{cases} \frac{1}{2}\log 2 , & \text{if } \lambda N \equiv 3 \bmod 4 \\ \log 2 , & \text{if } \lambda N \equiv 5 \bmod 8 \\ 2\log 2 , & \text{if } \lambda N \equiv 1 \bmod 8. \end{cases}$$

If $p|\lambda$ the expected contribution E_p is $(\log p)/p$. Thus we wish to choose the value of λ so as to maximize the function

$$F(\lambda,N) = -\frac{1}{2}\log|\lambda| + \sum_{p\leq F} E_p$$

where the sum is over those primes $p \leq F$ with $p=2$, $(\lambda N/p) = 1$, or $p|\lambda$. This function is very similar to one associated with the continued fraction algorithm (see [3], p. 391, Ex. 28 or [12]).

SPECIAL PURPOSE PROCESSORS

J.W. Smith, S.S. Wagstaff, Jr., and I have discussed the feasibility of building a special purpose processor to implement the quadratic sieve algorithm. We are encouraged by the prospects. For a budget of perhaps $25,000 in parts, we believe a "quadratic siever" could be built that would rival a Cray in speed. For ten or twenty times as much money a machine could be built that could factor 100 digit numbers in a month. Perhaps these figures are way off, it is hard to tell unless one tries.

The basic idea of the "quadratic siever" would be to construct a sequence of $16 \times 4K$ units each of which would sieve over an interval of length 4096. The largest moduli (fastest through the sieve) would be started one after the other through the sequence of units. There would never be interference of moduli because we have let the fastest racers start first.

Another idea is to use many unextraordinary computers each using a different batch of polynomials with one central computer which is fed the factored residues.

With all of these ideas we may begin to approach the 100 digit level in factoring. But 150 digit numbers should be about 100,000 times harder and it seems clear that current methodology is insufficient for factoring such huge numbers. However, until someone proves that factoring *must* be hard, there will always be some doubt about the security of R SA. When R SA was introduced 40 digit numbers were considered hard to factor, while now we are doing 70 digit numbers and talking about 100 digit numbers. As always, the future is hard to predict.

ACKNOWLEDGEMENTS

I would like to thank the Département de Mathématiques-Informatique at the U.E.R. des Sciences de Limoges for their hospitality while this paper was written. I would also like to thank H.J.J. te Riele for helping me track down the Kraitchik references and Peter Montgomery for his kind permission to describe his improvement to the quadratic sieve alogorithm.

REFERENCES

[1] E.R. Canfield, P. Erdös and C. Pomerance, On a problem of Oppenheim concerning "Factorisatio Numerorum", J. Number Theory, 17 (1983), 1-28.

[2] J.A. Davis and D.B. Holdridge, Factorization using the quadratic sieve algorithm, Sandia Report Sand 83-1346, Sandia National Laboratories, Albuquerque, New Mexico, 1983.

[3] D.E. Knuth, The Art of Computer Programming, vol. 2, Seminumerical Algorithms, 2nd edition, Addison Wesley, Reading, Mass., 1981.

[4] M. Kraitchik, Théorie des Nombres, Tome II, Gauthier-Villars, Paris, 1926.

[5] M. Kraitchik, Recherches sur la Théorie des Nombres, Tome II, Factorisation, Gauthier-Villars, Paris, 1929.

[6] D.H. Lehmer and R.E. Powers, On factoring large numbers, Bull. Amer. Math. Soc. 37 (1931), 770-776.

[7] J.C.P. Miller, On factorisation with a suggested new approach, Math. Comp. 29 (1975), 155-172.

[8] L. Monier, Algorithmes de factorisation d'entiers, thèse de 3e cycle, Orsay (1980).

[9] P. Montgomery, private communication.

[10] M.A. Morrison and J. Brillhart, A method of factoring and the factorization of F_7, Math. Comp. 29 (1975), 183-205.

[11] C. Pomerance, Analysis and comparison of some integer factoring algorithms, in Computational Methods in Number Theory, H.W. Lenstra, Jr. and R. Tijdeman, eds., Math. Centrum Tract 154 (1982), 89-139.

[12] C. Pomerance and S.S. Wagstaff, Jr., Implementation of the continued fraction algorithm, Cong. Numerantium 37 (1983), 99-118.

[13] C.P. Schnorr and H.W. Lenstra, Jr., A Monte Carlo factoring algorithm with finite storage, preprint.

[14] M.C. Wunderlich, A report on the factorization of 2797 numbers using the continued fraction algorithm, unpublished manuscript.