MATHEMATICS OF COMPUTATION
Volume 68, Number 228, Pages 1729–1737
S 0025-5718(99)01133-3
Article electronically published on March 1, 1999

SPEEDING FERMAT'S FACTORING METHOD

JAMES MCKEE

ABSTRACT. A factoring method is presented which, heuristically, splits composite n in $O(n^{1/4+\epsilon})$ steps. There are two ideas: an integer approximation to $\sqrt{(q/p)}$ provides an $O(n^{1/2+\epsilon})$ algorithm in which n is represented as the difference of two rational squares; observing that if a prime m divides a square, then m^2 divides that square, a heuristic speed-up to $O(n^{1/4+\epsilon})$ steps is achieved. The method is well-suited for use with small computers: the storage required is negligible, and one never needs to work with numbers larger than n itself.

1. INTRODUCTION

Let n be an odd, composite integer. The aim is to find a non-trivial factorisation of n. Let

$$b = \lceil \sqrt{n} \rceil \,, \tag{1}$$

where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. Define

$$Q(x,y) = (x+by)^2 - ny^2 \,. \tag{2}$$

To factor n, we seek integers x, y and z such that

$$Q(x,y) = z^2 \,. \tag{3}$$

A solution to (3) gives $(x+by)^2 \equiv z^2 \pmod{n}$. We then compute

$$\gcd(x+by-z, n)$$

in the hope that this will be a non-trivial factor of n (i.e., a factor other than 1 or n).

Taking $y = 1$ and $x \geq 0$ gives Fermat's factoring method. Allowing y greater than 1 gives some improvement, and can be viewed as writing n as the difference of two rational squares, then clearing denominators. The SQUFOF method of Shanks (for a good exposition, see §8.7 of [1]) looks for solutions to (3) with both x and y very large, but z very small ($|z| = O(n^{1/4})$). Of course, one can reduce x and y mod n to keep them below n, but SQUFOF does still better: it never computes x and y at all, and mostly works with numbers of size $O(\sqrt{n})$. A theoretical disadvantage of SQUFOF is that the existence of a suitable solution to (3) (i.e., one which splits n) is not guaranteed. In practice, however, the method works very well for smallish numbers. It is used for subsidiary factorisations in certain other factoring methods, such as the quadratic sieve.

In this paper, another factoring method based on finding solutions to (3) is presented. In contrast to SQUFOF, the solutions that we seek have x and y small,

Received by the editor March 28, 1997.
1991 *Mathematics Subject Classification.* Primary 11Y05; Secondary 11Y16, 68Q25.

and consequently z must be rather larger. Moreover, we can guarantee the existence of a solution which splits n: the only problem is finding it. A heuristic method is given, which is expected to factor n in $O(n^{1/4+\epsilon})$ steps. As with SQUFOF, the computations require much less accuracy than mod n arithmetic: we expect to need $O(n^{1/4})$ reductions of n modulo numbers of size $O(\sqrt{n})$, and nearly all other computations require working to an accuracy of at most half the bits of n. Depending on implementational details, the method could be a candidate for replacing SQUFOF, or for being used as well as SQUFOF, in some applications.

The next section gives the key Lemma, which guarantees the existence of a suitable solution to (3). This immediately gives a factoring algorithm, certain to factor n in $O(n^{1/2+\epsilon})$ operations, requiring only $O(n^{1/4})$ trial divisions. Here, and throughout this paper, ϵ denotes a positive real number, as small as we please, with the implied constant in the $O(\cdot)$ depending on ϵ. In fact, the n^{ϵ} factors will never be worse than some power of $\log n$. Section 3 gives a practical method for searching for a suitable solution to (3), which is not guaranteed to work (suitable solutions exist, but the method may not find one), but heuristically will factor n in $O(n^{1/4+\epsilon})$ operations. After an example of the basic method, and some remarks about implementing it on small computers, Section 6 considers some variants. Section 7 gives an interpretation of the method in terms of special changes of polynomial in the quadratic sieve, before some timings for a selection of random numbers. The paper concludes with a brief discussion of other applications of the idea used to speed the search for solutions to (3).

The original intention had been to produce a true algorithm (i.e., one guaranteed to work), and the lemma of the next section provides one, albeit of no practical value. It was a happy chance that a heuristic method for speeding the search for the known solution to (3) proved to be efficient for factoring smallish numbers, and well-suited for use with small computers.

2. The key lemma

The following lemma shows that, after trial division up to $2n^{1/4}$, a naive search for solutions to (3) can be organised so as to be certain to factor n in $O(n^{1/2+\epsilon})$ operations. We suppose, then, that $n = pq$ with $2n^{1/4} < p < q$. Here we may (but do not need to) suppose that p is prime, but we cannot assume that q is prime. Note that, setting $y = 1$ in (3), to give Fermat's method, one might need to take x of order $n^{3/4}$ to find a solution to (3).

Lemma. *Suppose that $n = pq$ with $2n^{1/4} < p < q$. Then with $Q(x, y)$ defined by* (1) *and* (2)*, there exists a solution to* (3) *with* $\gcd(x+by-z, n)$ *a non-trivial factor of n and*

(i) $$2 \le y \le n^{1/4}, \ y \text{ even};$$

(ii) $$|x|y < 2n^{1/2};$$

(iii) $$0 \le z < 2n^{1/2}.$$

In particular, (ii) implies that such x, y and z can be found in $O(n^{1/2+\epsilon})$ operations.

Proof. Let $r = \lfloor \sqrt{(q/p)} \rfloor$, the largest integer not greater than $\sqrt{(q/p)}$. Then $1 \le r \le n^{1/4}/2$. Set $y = 2r$ (so that (i) holds), $x = r^2p + q - by$, $z = q - r^2p$. Then $Q(x, y) = z^2$, so that x, y and z give a solution to (3).

To prove (iii), note that $\sqrt{(q/p)} - 1 < r \le \sqrt{(q/p)}$, so that

$$0 \le z = q - r^2 p < q - q + 2\sqrt{n} - p < 2\sqrt{n}\,,$$

which is (iii).

To prove (ii), write $b = \sqrt{n} + \delta$, with $0 \le \delta < 1$. Then

$$\begin{aligned}(x+by)^2 &= x^2 + 2xy(\sqrt{n}+\delta) + y^2 n + 2\delta y^2\sqrt{n} + \delta^2 y^2 \\ &= y^2 n + z^2\,.\end{aligned}$$

If $x \ge 0$, then this gives

$$xy \le \frac{z^2}{2(\sqrt{n}+\delta)} < 2\sqrt{n}\,,$$

using (iii). If $x < 0$, then we write $r = \sqrt{(q/p)} - \eta$, with $\eta \ge 0$, and note that $x = \eta^2 p + 2\eta\delta - 2\sqrt{n}\delta/p$, so that $|x| < 2\sqrt{n}/p < n^{1/4}$, giving $|x|y < \sqrt{n} < 2\sqrt{n}$. Thus (ii) holds whether or not $x \ge 0$.

Finally, (i), (ii) and (iii) imply that

$$\begin{aligned}|x + by \pm z| &< 2n^{1/2}/y + (n^{1/2}+1)y + 2n^{1/2} \\ &\le n^{3/4} + 2n^{1/2} + 3n^{1/4} \\ &< n\,,\end{aligned}$$

at least for $n > 31$, but the condition that $n = pq$ with $2n^{1/4} < p < q$ implies that $n = 30$ or $n \ge 35$, and for $n = 30$ we have $b = 6$, $x = -1$, $y = 2$, $z = 1$, so that $|x + by \pm z| < n$ here too. If $x + by = z$, then (2) and (3) imply that $y = 0$, contradicting (i).

Thus $\gcd(x + by - z, n)$ is a non-trivial factor of n. □

It is interesting to compare this with R.S. Lehman's factoring method, [2]. He also speeds up Fermat's method, by seeking a solution to $an = x^2 - z^2$ with $1 \le a \le n^{1/3}$, where $n = pq$ with $n^{1/3} < p < q$. His algorithm is rigorous, and runs in $O(n^{1/3+\epsilon})$ steps. One can prove the existence of a suitable solution to $an = x^2 - z^2$ by considering good rational approximations to q/p. For our method, we demand only that $2n^{1/4} < p < q$, and then obtain a suitable solution to $y^2 n = x^2 - z^2$ by considering a good integer approximation to $\sqrt{q/p}$, but without a further heuristic speed-up we obtain only an $O(n^{1/2+\epsilon})$ algorithm.

3. The method

After the Lemma of the previous section, we have a factoring algorithm which runs in $O(n^{1/2+\epsilon})$ steps. We now describe a practical method for speeding the search for solutions to (3), to give a factoring method which runs in $O(n^{1/4+\epsilon})$ steps, subject to a heuristic argument.

Suppose that we have a solution to (3) satisfying (i), (ii) and (iii) of the Lemma, where we may suppose that x, y and z have no common prime factor. Suppose further that m divides z. Then $Q(x,y) \equiv 0 \pmod{m^2}$. Let $x_0 \equiv xy^{-1} \pmod{m^2}$, where y^{-1} is the inverse of $y \pmod{m^2}$. Then $Q(x_0, 1) \equiv 0 \pmod{m^2}$. We have $x = x_0 y - \lambda m^2$ for some λ, so that

$$\frac{x_0}{m^2} - \frac{\lambda}{y} = \frac{x}{m^2 y}\,. \tag{4}$$

If $x > 0$ and

$$m^2 > 2xy\,, \tag{5}$$

then (4) implies that

$$0 < \frac{x_0}{m^2} - \frac{\lambda}{y} < \frac{1}{2y^2}\,,$$

so that λ/y is a convergent in the continued fraction expansion of x_0/m^2.

This suggests a means for searching for solutions to (3). We suppose, after trial division, that n has no prime factors below $2n^{1/4}$. From the Lemma, (5) will certainly hold if $m > 2n^{1/4}$. We take several m larger than $2n^{1/4}$, hoping that m divides z, and for each m we proceed as follows:

Step 1. Compute all solutions x_0 to the equation $Q(x_0, 1) \equiv 0 \pmod{m^2}$.

Step 2. For each x_0 found in Step 1, compute those convergents λ/y in the continued fraction expansion of x_0/m^2 for which $\lambda/y < x_0/m^2$ and $y \le n^{1/4}$. Check if $Q(x_0 y - \lambda m^2, y)$ is a square. If it is, then we can factor n and stop, else we keep trying.

In Step 1 we need to compute a square-root of $n \pmod{m^2}$. If m is prime, then this is easy in practice, and for most prime m it is provably easy (polynomial time). The method will factor n when we take m to be the least prime factor of z greater than $2n^{1/4}$ (unless $x \le 0$, which happens rarely, but we could easily modify the search to allow for this). Of course, such a prime factor may not exist, or it may be too large to be of use in practice. We need more solutions to (3) for there to be any hope of success. The following refinement of the Lemma gives us this hope.

Refinement. *With n as in the Lemma, and T any integer greater than* 1*, there exist at least T solutions to* (3) *with*

(i) $$2 \le y \le n^{1/4} + 2(T-1),\ \ y \ \ \textit{even};$$

(ii) $$|x|y < T^4\sqrt{n}\,;$$

(iii) $$|z| < (T^2 - 1)\sqrt{n}\,.$$

Moreover there exist at least $T-1$ solutions to (3) *satisfying all of the above if we restrict to $x > 0$.*

Proof. Let r be as in the proof of the Lemma. For $0 \le t \le T-1$, we get a solution to (3) given by $x = (r+t)^2p + q - by$, $y = 2(r+t)$, $z = q - (r+t)^2 p$. This gives (i) immediately. For $t = 0$, (ii) and (iii) follow from the Lemma, so we may suppose that $t > 0$. Then

$$0 > z = q - (r+t)^2 p \ge q - (\sqrt{(q/p)} + t)^2 p = -2t\sqrt{n} - t^2 p\,,$$

giving

$$|z| < 2t\sqrt{n} + t^2 p < t(t+2)\sqrt{n} \le (T^2-1)\sqrt{n}\,,$$

which is (iii). For (ii), we follow the proof in the Lemma, noting that now x is always positive, giving

$$0 < xy < z^2/2\sqrt{n} < (T^2-1)^2\sqrt{n}/2 < T^4\sqrt{n}\,.$$

□

If T is small compared to n, then again an easy estimate shows that all such solutions to (3) will split n. One could give a further refinement, using rational approximations to $\sqrt{(q/p)}$ (rather than integer approximations) to produce solutions to (3).

The following crude heuristic argument now suggests that the method should run in $O(n^{1/4+\epsilon})$ steps. Take T of order $\log n$. The probability that any one value of z given by the Refinement should be prime to all primes m between $T^2n^{1/4}$ and $2T^2n^{1/4}$ is heuristically of order $1/\log n$, so we expect success with $m \ll n^{1/4}(\log n)^2$. One does not need to restrict m to being prime, but it keeps the bookkeeping simple when taking square-roots mod m^2.

4. A worked example

Consider $n = 84009841$. Here $n^{1/4} \approx 96$. Prime m for which the method successfully factors n are

$$m = 73,\ 179,\ 229,\ 619,\ 641,\ 1031,\ \dots\ .$$

Let us work through the method in detail for $m = 73$ and $m = 179$.

Here we have $b = 9166$, $Q(x,y) = (x+by)^2 - ny^2 = x^2 + 18332xy + 5715y^2$.

For $m = 73$, we compute $\sqrt{n} \equiv \pm 1123 \pmod{73^2}$, from which we find $Q(x_0, 1) \equiv 0 \pmod{73^2}$ for $x_0 \in \{369, 2615\}$. $Q(369,1) = 2628^2 = (36.73)^2$, which splits n:

$$\gcd(369 + 9166 - 2628, n) = 6907.$$

The case $m = 179$ illustrates Step 2 of the method. We compute $\sqrt{n} \equiv \pm 5517 \pmod{179^2}$, from which we find $Q(x_0, 1) \equiv 0 \pmod{179^2}$ for $x_0 \in \{17358, 28392\}$. Neither value of x_0 makes $Q(x_0, 1)$ square. For $x_0 = 28392$, the relevant continued fraction approximations to x_0/m^2 are 0/1, 7/8, 31/35, 70/79. We find that $35x_0 \equiv 449 \pmod{179^2}$, and $Q(449, 35) = 17184^2 = (96.179)^2$, which again splits n.

For this example, I considered all prime $m \geq 3$. The argument of the previous section requires $m > 2n^{1/4}$, but one can be lucky with smaller m, as here with $m = 73$ and $m = 179$.

5. Remarks on implementation

As a practical remark, note that one does not need to perform arithmetic mod m^2 to compute $yx_0 \pmod{m^2}$: we are simply performing the (extended) Euclidean algorithm on x_0 and m^2. The values of $yx_0 \pmod{m^2}$ are given by the relevant remainders.

The method is (relatively) easy to implement on small computers. Pollard informs me that he has programmed it to run on a tiny Psion computer (Psion Series 3A, 256K). This has a 16-bit processor running at 7MHz. His program allows numbers up to 17 digits (stored as $9+4+4$ digits), and tests m between 3 and 32000. For computing a square root (mod m) one can use Shanks' method (Algorithm 1.5.1 of [1]), extending this to a square root (mod m^2) at the expense of computing an inverse mod m. Then one performs Euclid's algorithm on x_0 and m^2 to find relevant values of x and y.

Since $Q(x,y)$ will be divisible by m^2, one does not need to work to great accuracy to test if $Q(x,y)$ is a square: simply compute $\sqrt{(Q(x,y))}/m$ (which will be of order $n^{1/4}$ when m is of order $n^{1/4}$) in real arithmetic, to enough accuracy to guess whether it is an integer. It may be better first to test whether $Q(x,y)$ is a square modulo a few small prime powers. Pollard reports that checking m up to 32000 takes about 12 minutes on his tiny machine ([6]). For 17-digit numbers, m rarely needs to be this large.

For some timings on a more powerful computer, see the end of this paper.

On a small machine, where one wishes to avoid multiple precision arithmetic, m will in effect be bounded above. The method may then fail, if the smallest successful m is too large. One could then try the method on small multiples of n until these also become too large.

The method is easy to parallelise. Different machines can run through m lying in different residue classes mod k, where the number of machines equals $\varphi(k)$ (the totient function).

6. Variants of the method

6.1. **Interpolating the continued fraction approximations.** If the continued fraction approximations to x_0/m^2 are p_0/q_0, p_1/q_1, $\dots$, then we can write

$$p_{r+1} = a_r p_r + p_{r-1}\,,$$
$$q_{r+1} = a_r q_r + q_{r-1}\,,$$

for some a_r. For $1 \le \lambda \le a_r$, the fractions

$$\frac{\lambda p_r + p_{r-1}}{\lambda q_r + q_{r-1}}$$

give rational approximations to x_0/m^2 which are not as good as the continued fraction approximations, but might (if we are lucky) reveal solutions to (3). For example, with $n = 84009841$, as in the example above, we find that primes m for which this variant splits n are

$$m = 73,\ 83,\ 179,\ 229,\ 233,\ 241,\ 569,\ 619,\ 641,\ 661,\ 823,\ 967,\ 1031,\ \dots\,,$$

and we see that there are rather more of them. Of course, it now takes longer to test each m, and in practice this variant generally takes longer than the original method.

With n as above and $m = 83$, we have $x_0 \in \{2738, 6486\}$. For $x_0 = 2738$, the continued fraction denominators that were tried for y in the original method were 1, 3 and 78. Here we interpolate between 3 and 78 in steps of 5, finding $Q(269, 63) = 18260^2 = (220.83)^2$, which splits n.

6.2. **The greedy variant.** Given m and x_0 such that $Q(x_0, 1) \equiv 0 \pmod{m^2}$, we first test if $Q(x_0, 1)$ is a square. If it is, then we are done, else set

$$r_1 = \lceil m^2/x_0 \rceil\,,$$
$$x_1 = x_0 r_1 - m^2\,,$$
$$y_1 = r_1\,,$$

and test if $Q(x_1, y_1)$ is a square. If it is, then we are done, else set

$$r_2 = \lceil m^2/x_1 \rceil\,,$$
$$x_2 = x_1 r_2 - m^2\,,$$
$$y_2 = y_1 r_2\,,$$

and so on. At the jth stage we have

$$\begin{aligned} r_j &= \lceil m^2/x_{j-1} \rceil \,, \\ x_j &= x_{j-1} r_j - m^2 \,, \\ y_j &= y_{j-1} r_j \,. \end{aligned}$$

We proceed until either $Q(x_j, y_j)$ is a square, or y_j exceeds a chosen bound (of order $n^{1/4}$).

With n as before, we find that primes m for which this variant splits n are

$$m = 59,\ 73,\ 83,\ 229,\ 641,\ 809,\ 967,\ 1031,\ \dots\ .$$

We see that for some values of m (e.g., $m = 179$) we miss factorisations revealed by the basic continued fraction approach, but there are other values of m (e.g., $m = 59$) which succeed here but not with either of the previous variants. Note that the greedy variant requires only one Euclidean division to compute each value of y, compared to the two required by the continued fraction approach (since for the latter only alternate denominators are used).

For $m = 59$, we find that $x_0 \in \{1035, 1519\}$. With $x_0 = 1519$, $x_2 = 823$, $y_2 = 12$, and $Q(823, 12) = (229.59)^2$, which splits n. Indeed we find the same solution to (3) as we did with $m = 229$ using the basic method.

For a larger example, take $n = 663621\ 112452\ 523783$. Here one finds that $m = 95971$ splits n (with $y = 112$), finding the factors 700 119223 and 947 868721.

Using PARI-gp on a SUN Sparc 5, I have found the greedy variant to be best. Pollard, however, on a PSION 3A, finds this variant slower than the basic method, observing that $Q(x, y)$ is more often a square mod 64 for the values of x and y produced by this variant. Hence the test for squareness takes longer on average. His observation can be explained by noting that $Q(x_j, y_j)$ is a square mod ny_j^2, and for the greedy variant the y_j tend to have many small factors (y_j is the product of $r_1, \dots, r_j$, and the r_j are usually small). If $x_j \equiv 0 \pmod 8$, then $Q(x_k, y_k)$ is a square mod 64 for all $k \geq j$.

6.3. **Composite moduli.** Any of the variants discussed work in principle with composite values of m, but we may then need to consider more than two possibilities for x_0. For example, $n = 84009841$ (as above) is split by $m = 36$: $x_0 \in \{369, 739, 1179, 1225\}$, and we have seen already that $Q(369, 1) = (36.73)^2$.

7. Interpolating the quadratic sieve

With $Q(x, y)$ defined by (2), we have $Q(x, y) \equiv (x + by)^2 \pmod n$. In the above method, we sought $Q(x, y)$ an exact square. Instead, we can try to factor $Q(x, y)$ over a factor base $\mathcal{B}$, consisting of a few small primes, then use several relations

$$(x_i + by_i)^2 = Q(x_i, y_i) = \prod_{p \in \mathcal{B}} p^{\alpha_{p,i}} \tag{6}$$

to find squares which agree mod n (by Gaussian elimination over the field with two elements), in the hope of factoring n. This idea is the basis of several modern factoring methods.

If we restrict to $y_i = 1$, then we have a simple version of the quadratic sieve (§10.4.1 of [1]). For $p \in \mathcal{B}$ we compute values of x_p (generally two of them) such that $Q(x_p, 1) \equiv 0 \pmod p$. This allows us to sieve over a range of x and find relations as in (6) more quickly. As x grows larger, so does $Q(x, 1)$, and it becomes less likely that $Q(x,1)$ will factor over $\mathcal{B}$. In the multiple polynomial version (§10.4.2 of [1]), $Q(x,1)$ is replaced after a while by another quadratic polynomial (with the same discriminant), to keep the numbers represented by the polynomial small. The cost of such a change is that the x_p have to be recomputed.

Moving from $Q(x,1)$ to $Q(x,y)$, for fixed y, gives a relatively cheap change of polynomial, in that the new x_p are computed simply by multiplying the old x_p by $y \pmod p$. We can view this change of polynomial as a rational interpolation of $Q(x,1)$, taking $x \in \mathbb{Q}$ with denominator dividing y, then clearing denominators. One also notes that this process is essentially inverse to the "special q's" idea of Davis and Holdridge, in which $Q(x,1)$ is replaced by the polynomial $Q(qx+x_0,1)/q$, where $Q(x_0,1) \equiv 0 \pmod q$.

Having viewed the process of moving from $Q(x,1)$ to $Q(x,y)$ as giving rational approximations to $\sqrt{n}$, rather than integer ones, we may ask what happens if we look for the best rational approximations, as appear in the continued fraction expansion of $\sqrt{n}$. Not surprisingly, this gives the continued fraction factoring method (§10.1 of [1]).

For the method of this paper, we seek $Q(x,y) = z^2$. By looking for $z \equiv 0 \pmod m$, we are using the special q's idea with $q = m^2$. In allowing $y > 1$, we are interpolating.

8. Timings

The timings shown in Table 1 should be interpreted with caution. I used my own routines for both SQUFOF and the method of this paper, using PARI, via the PARI-gp programming language, on a Sun Sparc 5. These should run much more quickly if the PARI routines were called from a C program. For comparison, I give also the times to split n using the MAPLE V routine `ifactor(n,squfof)`.

My SQUFOF program follows Algorithm 8.7.2 of [1], making no use of the infrastructure of the class group. The values of n were products of two randomly chosen primes. For the method of this paper, I used the greedy variant, with $m > n^{1/4}$, and $y < n^{1/4}/10$. Strictly, I should first eliminate the possibility of factors below

Table 1

n	SQUFOF		Our method	
	MAPLE V	PARI-gp	PARI-gp	
			n	$3n$
$91739369 \times 266981831$	4	10	3	6
$327083137 \times 1245254189$	25	66	19	46
$1640261503 \times 1672679527$	192	520	277	1
$700119223 \times 947868721$	0	60	41	7
$108797839 \times 1832615053$	20	53	8	203
$883283243 \times 1682761103$	179	137	9	3
$1258514107 \times 2023452479$	214	575	61	26
$22178813 \times 669848353$	74	163	19	2
$1018825649 \times 1690938463$	30	80	26	48
$589472101 \times 1180140427$	0	37	176	15
mean	74	170	64	36
median	27	73	23	11

about $n^{1/4}$. On the Sun, using PARI, this can be done in negligible time using Pollard's ρ method (Algorithm 8.5.2 in [1]). On a small computer, trial division may be preferable, in view of the extra complications of performing arithmetic to greater precision. Alternatively, one can start with m as small as 3, thereby incorporating trial division into the method.

As with SQUFOF, there is considerable variation in the time taken to split numbers of similar size using the method of this paper. In Table 1, I give the times (in seconds, to the nearest second) taken to split $3n$, as well as n. There is considerable advantage in trying both: the mean of the minima of times to split n and $3n$ is only 11 seconds, with median 7 seconds. Thus if one tested each m on both n and $3n$ the mean time would be 22 seconds, with median 14 seconds.

9. Applications to other methods

The key idea used to speed the search for solutions to (3) is that given any m, a square is divisible by m^2 if and only if its square root is divisible by m, so that we gain information mod m^2 with probability $1/m$. The obvious generalisation may be applied to any polynomial equation in which one of the variables appears only as a square or higher power. This has been used already in another speed-up of Fermat's method, in §5 of [5]. It can also be applied to Euler's method, or the variants in [3] and [4]. These seek solutions to $an = x^2 + dy^2$ for various a and d.

For example, to solve $n = x^2 + y^2$ (when possible), we can take several prime m of size about $n^{1/4}$ and hope that either m divides x or m divides y. Each such m can be tested in polynomial time (assuming that we can compute square roots mod m^2 in polynomial time, which we can in practice). If we are unlucky, and do not find a solution, then we can try an in place of n, where a is representable by the form $x^2 + y^2$. This gives a heuristic $O(n^{1/4+\epsilon})$ method. Applying this idea to the $O(n^{1/2+\epsilon})$ algorithm in [3] gives another heuristic $O(n^{1/4+\epsilon})$ factoring method.

Acknowledgments

I am grateful to John Pollard for numerous comments on an early draft of this paper, and for remarks based on his practical experience with the method. I am also grateful for helpful comments on a later draft by an anonymous referee. The work for this paper began when I was at Trinity Hall, Cambridge; the first draft was written when I was at the University of Edinburgh.

References

[1] H. Cohen, *A course in computational algebraic number theory*, Graduate Texts in Mathematics 138, Springer, 1993. MR **94i:**11105

[2] R.S. Lehman, Factoring large integers, *Math. Comp.*, **2**8 (1974), 637–646. MR **49:**4949

[3] D.H. Lehmer and Emma Lehmer, A new factorization technique using quadratic forms, *Math. Comp.*, **2**8 (1974), 625–635. MR **49:**7204

[4] J.F. McKee, Turning Euler's factoring method into a factoring algorithm, *Bull. London Math. Soc.*, **2**8 (1996), 351–355. MR **97f:**11010

[5] J.F. McKee and R.G.E. Pinch, Old and new deterministic factoring algorithms, in *Algorithmic Number Theory, Proceedings of the Second International Symposium, ANTS-II* (H. Cohen, ed.), Lecture Notes in Computer Science 1122, Springer, 1996, pp. 217–224. MR **98a:**11183

[6] J.M. Pollard, Personal communication.

Pembroke College, Oxford, OX1 1DW, UK
E-mail address: jfm@maths.ox.ac.uk