Journal of Computational and Applied Mathematics 27 (1989) 267–278
North-Holland

Factoring with the quadratic sieve on large vector computers

Herman TE RIELE, Walter LIOEN and Dik WINTER
Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Received 29 June 1988

Abstract: The results are presented of experiments with the multiple polynomial version of the quadratic sieve factorization method on a CYBER 205 and on a NEC SX-2 vector computer. Various numbers in the 50–92 decimal digits range have been factorized, as a contribution to (i) the Cunningham project, (ii) Brent's table of factors of Mersenne numbers, and (iii) a proof by Brent and G. Cohen of the non-existence of odd perfect numbers below 10^{200}. The factorized 92-decimal digits number is a record for general purpose factorization methods.

Keywords: Quadratic sieve factorization, vector computer.

1. Introduction

About ten years ago Rivest et al. [20] discovered that the difficulty of breaking certain cryptographic codes depends on the difficulty of factoring large numbers. This discovery stimulated the renewed interest in the classical problem of the factorization of integers. In 1974, it was considered very difficult to factor numbers in the 40–50 decimal digit range [10, Fig. 1, p. 185]. Now, fourteen years later, 70–80 digits (and even larger!) numbers are factorized in a routine way by Silverman [8,21], Montgomery [15], Brent [2] and, very recently, Lenstra and Manasse [23]. This demonstrates the huge progress in the past decade, particularly when we take into account the—experimental—fact that if the number of decimal digits of the number to be factorized is increased by three, then the amount of CPU-time needed is roughly doubled. A well-written survey of modern factoring and primality testing methods may be found in [5]. Many of the numbers in that book have already been factorized, many others are still awaiting to be factorized. The tables of "most" and "more wanted" numbers, which are regularly updated, are good (but hard) starting points for those who want to contribute to this book.

Factoring can be done on any computer, from pocket calculator [1] to supercomputer (as described in this paper). Silverman has been very successful in using computers in parallel for factoring [8,11]. Of course, any given computer system puts its own specific requirements on the factoring algorithm to be chosen, and on the tuning of the algorithm parameters involved (cf. [7]).

Many numbers in [5] have been factorized by means of Lenstra's elliptic curve method (ECM), described in [13]. Improvements were proposed by Montgomery [15] and Brent [2], and impressive factorizations with ECM were obtained by these people, by Silverman, and, most recently, by Lenstra and Manasse [23].

In this paper, we shall describe our experiences with the so-called multiple polynomial quadratic sieve factorization method (MPQS) on large vector computers. This method may be considered as complementary to ECM since the computing time of MPQS depends on the size of the number to be factorized, whereas ECM's computing time depends on the size of the smallest prime factor of the number to be factorized. At present, numbers with smallest prime divisor up to 30 decimal digits may be best factorized with help of ECM, whereas numbers with smallest prime divisor greater than 30 digits may be best factorized with help of MPQS, provided that the size of the number to be factorized does not exceed about 90 decimal digits. Of course, we usually do not have such knowledge about the size of the prime divisors we are seeking. Anyway, ECM should always be tried before MPQS, in order to eliminate the smaller prime divisors first.

The current record of factorizing numbers by ECM is (L_n is the nth Lucas number, Cxx means a composite number of xx digits, and Pxx a prime number of xx digits)

$$L_{464}/2207 = P36 \cdot P59$$

found by Montgomery [6] and the current record of MPQS is

$$(6^{131} - 1)/(5 \cdot 263 \cdot 3931 \cdot 6551) = C92 = P34 \cdot P59,$$

found by us and described in the present paper.

Numbers which are composed of two prime divisors of approximately equal size are the hardest to factorize, and are particularly interesting for cryptography: they are suitable to act as keys in so-called RSA public-key cryptosystems [20]. The above records indicate that numbers of about 100 decimal digits can no longer be considered as safe keys (as they were about 10 years ago).

We shall report our experiences with the version of MPQS described in [18], on two single-CPU vector computers: a CDC CYBER 205 and a NEC SX-2. Since these machines belong to the fastest (commercially) available single-CPU vector computers, and since MPQS is the best known general purpose factorization method, our results implicitly present the current state-of-the-art of factoring by general purpose methods.

The largest number we factorized on the CYBER 205 has 82 digits and required about 70 CPU-hours; on the NEC SX-2 we factorized a 92-digit composite number in about 95 hours CPU-time. Earlier, Davis and Holdridge implemented a variant of the quadratic sieve on a CRAY-1 and on a CRAY X-MP [10]. Their record is the C71 $(10^{71} - 1)/9$, which took them about 9.5 hours on the CRAY X-MP.

In Section 2 of this paper we describe the multiple polynomial version of the quadratic sieve factoring algorithm. This algorithm goes back to Kraitchik [12]; Pomerance was the first to describe and analyze it in its modern form [17]. Davis and Holdridge [10] and, independently, Montgomery [18] proposed the use of multiple polynomials in the quadratic sieve algorithm. Section 3 gives a global description of the CDC CYBER 205 and the NEC SX-2, and the values of the algorithm parameters which we used in our implementations on the two machines. Section 4 presents our computational results in the form of tables of the numbers we have factorized so far. These fall into three categories:

(i) numbers of the form $b^n \pm 1$ [5],
(ii) numbers which play a role in primality proofs of factors of Mersenne numbers $2^n - 1$ [3],
(iii) numbers which play a role in the proof of the non-existence of odd perfect numbers below 10^{200} [4].

2. The multiple polynomial quadratic sieve

2.1. The MPQS-algorithm

Let N be the (large) number, which is known to be composite by Fermat's little theorem, and which we want to factorize. The quadratic sieve algorithm belongs to a class of algorithms which have the common aim to find two integers X and Y such that

$$X^2 \equiv Y^2 \pmod{N}. \tag{2.1}$$

If $d := \gcd(X - Y, N)$ satisfies $1 < d < N$, then d is a proper divisor of N. In order to find such an (X, Y)-pair, one may try to find triples (U_i, V_i, W_i), $i = 1, 2, \ldots$, such that

$$U_i^2 \equiv V_i^2 W_i \pmod{N}, \tag{2.2}$$

where W_i is easy to factor, or at least easier than N. If sufficiently many congruences (2.2) have been found, these can be combined, by multiplying together a subset of them, in order to get a relation of the form (2.1).

The version of the quadratic sieve algorithm that we shall employ may be described as follows. Let $U(x) := a^2x + b$, $V := a$ and $W(x) := a^2x^2 + 2bx + c$, $x \in [-M, M)$, where a, b and c satisfy the following relations:

$$a^2 \approx \sqrt{2N}/M, \qquad b^2 - N = a^2c, \qquad |b| < \tfrac{1}{2}a^2, \tag{2.3}$$

and M is some fixed integer. Then we have

$$(U(x))^2 \equiv V^2W(x) \pmod{N}. \tag{2.4}$$

There are many pairs a, b satisfying (2.3). The quadratic polynomial $W(x)$ assumes extreme values in $x = 0, \pm M$, and these are such that $|W(0)| \approx |W(\pm M)| \approx M\sqrt{\frac{1}{2}N}$. If $M \ll N$, then it follows that $|W(x)| \ll N$; consequently, $W(x)$ is easier to factorize than N. Moreover, since $W(x)$ is a quadratic polynomial, it has the property that if $d \mid W(x_0)$ for some integer x_0, then $d \mid W(x_0 + kd)$ for any integer k. This property can be used to factorize those $W(x)$, $x \in [-M, M)$, whose prime factors are all smaller than some properly chosen number B, by the following *sieving* process; we initialize a sieving array $SI(j)$, $j = -M, \ldots, M - 1$, to zero and we add $\log(p)$ to $SI(j)$ for those $j \in [-M, M)$ for which $p^e \mid W(j)$, and we do that for all prime powers $p^e < B$. Since the values of $\log|W(x)|$ tend to stay constant on long subintervals of $[-M, M)$ (typically, we have $\log|W(x)| \approx \log(M/2\sqrt{\frac{1}{2}N}))$, we can now collect those $x \in [-M, M)$ for which $W(x)$ is only composed of prime factors $< B$, by selecting those $j \in [-M, M)$ for which $SI(j)$ is close to $\log(M/2\sqrt{\frac{1}{2}N})$. This last number shall be called the *report-threshold*. If, after the sieving, not sufficiently many factorized $W(x)$-values have been found, a new polynomial $W(x)$ is constructed. In Section 2.2 we give the details of how this can be done in a very efficient way.

The potential prime divisors p of a given quadratic polynomial $W(x)$ may be characterized as follows: if $p \mid W(x)$, then,

$$a^2W(x) = (a^2x + b)^2 - N \equiv 0 \pmod{p}, \tag{2.5}$$

i.e., the equation $t^2 - N \equiv 0 \pmod{p}$ should be solvable; in other words: p should be a quadratic residue of N (if we have found a solution $t = t_0$ of the equation $t^2 - N \equiv 0 \pmod{p}$, then we have $x = a^{-2}(t_0 - b)$ as a solution of (2.5)). Whether or not p is a quadratic residue of N is easily checked by means of the so-called Euler criterion [19, p. 280], and this is independent of the choice of the polynomial $W(x)$.

Hence, before we start sieving, we first find the primes $p < B$, for some suitable B, for which the equation $t^2 \equiv N \pmod p$ is solvable. This set of primes is called the *factor base FB*; it is fixed during the whole factorization process. The number of primes in the factor base will be denoted by L, and the primes in the factor base are indicated by p_j, for $j = 1, 2, \ldots, L$.

If at least $L+2$ completely factorized W-values have been collected, then (X, Y)-pairs satisfying (2.1) may be found as follows. We have integers x_i, $i = 1, 2, \ldots, L+2$, such that

$$W(x_i) = (-1)^{\alpha_{i0}} \prod_{j=1}^{L} p_j^{\alpha_{ij}}, \quad i = 1, 2, \ldots, L+2;$$

now we associate with $W(x_i)$ the vector $\boldsymbol{\alpha}_i$ defined by

$$\boldsymbol{\alpha}_i^{\mathrm{T}} := (\alpha_{i0}, \alpha_{i1}, \ldots, a_{iL}) \pmod 2.$$

Since we have more vectors $\boldsymbol{\alpha}_i$ (at least $L+2$) than components $(L+1)$, there exists at least one subset S of the set $\{1, 2, \ldots, L+2\}$ such that

$$\sum_{i \in S} \boldsymbol{\alpha}_i \equiv 0 \pmod 2,$$

so that

$$\prod_{i \in S} W(x_i) \text{ is a square, } Z^2 \text{, say.}$$

Hence, from (2.4) it follows that

$$\left[\prod_{i \in S} (a^2 x_i + b)\right]^2 \equiv Z^2 \prod_{i \in S} a^2 \pmod N,$$

which is of the form (2.1). The set S is to be found by Gaussian elimination (mod 2) on the binary matrix with columns $\boldsymbol{\alpha}_i$ [16]. This process may yield many different sets S. This is useful, since not every set S yields a $\gcd(X - Y, N)$ between 1 and N, and sometimes the number N is composed of more than two prime factors. In order to completely factorize such a number, we need more than one decomposition of N.

The multiple polynomial quadratic sieve algorithm may now be described as follows. A number of refinements and details are described in Section 2.2.

Algorithm MPQS (To factorize the composite number N)

Step 1. Choose B and M and compute the factor base FB;

Step 2. Generate a new quadratic polynomial $W(x)$;

Step 3. Solve $W(x) \equiv 0 \pmod q$, for all $q = p^e < B$, for all primes $p \in FB$, and save the solutions for each q;

Step 4. Initialize the sieving array $SI[-M, M)$ to zero;

Step 5. Add $\log(p)$ to all the elements $SI(j)$, $j \in [-M, M)$, for which $W(j) \equiv 0 \pmod q$, for all $q = p^e < B$, for all primes $p \in FB$;

Step 6. Select those j for which $SI(j) \approx \log \tfrac{1}{2}M + 0.5 \log \tfrac{1}{2}N)$ and report and save a, b and j (c follows from a and b, by (2.3));

Step 7. If the number of $W(x)$-values collected in Step 6 is $< L+2$, then go to Step 2;
Step 8. Perform Gaussian elimination on the matrix of exponents (mod 2) of $W(x)$;
Step 9. Factor N.

2.2. *Algorithmic refinements and details of the MPQS-algorithm*

(i) *Use of a multiplier.* Sometimes, it is worthwhile to premultiply the number N by a small, fixed, positive square-free integer, with the purpose to bias the factor base towards the smaller primes. The criterion we used to determine this multiplier is described in [18].

(ii) *Small prime variation.* When we sieve with a prime p, the number of sieving steps is $\lfloor 2M/p \rfloor$. This number is largest for small p, and its corresponding $\log(p)$-value does not contribute too much to the total $\log|W(x)|$-value. Therefore, it is advantageous to "forget" (as Pomerance names it) to sieve with the smallest primes. To compensate for this, one has to lower the report-threshold value in order not to miss any fully factorizable W-value. The only price to pay is the generation of some false W-values, i.e., which are not fully factorizable over the factor base.

(iii) *Large prime variation.* By lowering the report-threshold by an amount of $\log(\beta B)$, for some fixed $\beta > 1$, an unfactorized portion of size between B and βB is allowed in the $W(x)$-reports. If we manage to catch two of such W's with the *same* unfactorized part, we can combine these two to yield a completely factorized W-value. The birthday paradox promises that this will not happen too infrequently. Usually, the unfactorized portion is a large prime between B and βB, but this is not essential.

(iv) *Generation of polynomials.* Our choice of generating the quadratic polynomials $W(x)$ is a special case of one of several possible choices described in [18]. First, we prepare a list of r primes $g_1, g_2, \ldots, g_r$, of size $\approx (\sqrt{2N}/M)^{1/4}$, where we assume that M is chosen such that $(\sqrt{2N}/M)^{1/4} > B$. Moreover, the primes g are such that the equation $t^2 \equiv N \pmod{g_i^2}$ is solvable. We denote the two solutions by $\pm b_i$. Now let $a := g_i g_j$, $i \neq j$, be the product of two of the g-primes. Then with the Chinese Remainder Theorem [19, p. 268] we find a number b with $|b| < \frac{1}{2}a^2$ and

$$b \equiv b_i \pmod{g_i^2}, \qquad b \equiv \pm b_j \pmod{g_j^2}. \tag{2.6}$$

Then $b^2 \equiv N \pmod{a^2}$ and we take $c = (b^2 - N)/a^2$. These a, b and c satisfy (2.3). Since we have two choices for b, and $\binom{r}{2}$ choices for a we can form $r(r-1)$ different polynomials from the set $\{g_1, g_2, \ldots, g_r\}$.

For each polynomial we have to solve the congruences $W(x) \equiv 0 \pmod q$ for all $q = p^e < B$, for all primes $p \in FB$. This can be done efficiently for a fixed set of g-primes as follows. Solving $W(x) \equiv 0 \pmod q$ is equivalent to solving $(a^2x + b)^2 \equiv N \pmod q$. Let t_q^j, $j = 1, \ldots, k(q)$ be the, precomputed, solutions of the congruence $t^2 \equiv N \pmod q$ [19, pp. 212 and 287–288], then the numbers x for which $W(x) \equiv 0 \pmod q$ are given by

$$x = a^{-2}\left(t_q^j - b\right) \pmod q \quad \text{for } j = 1, 2, \ldots, k(q). \tag{2.7}$$

So we need the numbers $a^{-2} \pmod q$. Since $a = g_i g_j$, we have $a^{-2} \equiv g_i^{-2} g_j^{-2} \pmod q$; therefore, by precomputing and storing the numbers $g_i^{-2} \pmod q$ during the generation of the g-primes, we avoid the need to compute the inverses $a^{-2} \pmod q$ when we select a new polynomial by changing the g_i and g_j in a. The numbers g_i^{-2} (mod q) are found by solving the linear congruence $g_i^2x \equiv 1$ (mod q) [19, pp. 265–266].

3. Implementation of the MPQS-algorithm on the CYBER 205 and the NEC SX-2

We have implemented the MPQS-algorithm on two vector computers: the CDC CYBER 205 and the NEC SX-2. In Table 1 we list a number of hardware and software characteristics of the particular machines we used. Most of the operations in the MPQS-algorithm can be formulated in terms of large vectors of data and such vectors are processed extremely efficiently by these machines.

The dominant computations in the quadratic sieve are the sieving operations of Step 5, and these can be done in 32-bits floating-point arithmetic. This makes the quadratic sieve one of the most powerful factoring methods. The speed we obtained was about 13 million sieving operations per second (i.e., additions of log(p) to an element of the sieving array $SI[-M, M)$) on the CYBER 205, and about 90 million sieving operations per second on the NEC SX-2. On the NEC SX-2 the selection part (Step 6) also became time-critical, due to the high speed by which the sieving part (Step 5) could be executed. Details of how we have vectorized and optimized the time-critical loops in our FORTRAN programs are given in [14].

Table 1
Some hardware and software characteristics of the CYBER 205 and the NEC SX-2 used in our computations

	CYBER 205	NEC SX-2
Hardware		
Processors	Vector processor Scalar processor	Arithmetic processor Control processor
Vector pipes	1 set	4 sets
Vector registers	no	yes
Clock cycle	20 ns	6 ns
Maximum performance	200 Mflop/s	1300 Mflop/s
Word length		
integer	48 bits	32 bits
real	64 bits	32 bits
double		64 bits
Hardware arithmetic	single(64 bits)	double(64 bits)
Central memory	1 MWords	128 Mbytes
Software		
Operating system	VSOS	SXOS
FORTRAN	FORTRAN 200 with special vector syntax	FORTRAN 77/SX with vector directives
Vectorizing compiler	Yes, but very restricted	Yes, good
Optimizing tools	VAST	Vectorizer Optimizer Analyzer

Part of the MPQS-computations have to be carried out in multi-precision integer arithmetic. For this purpose, we used a package of Winter [24], which is also used in Lenstra and Cohen's primality proving program [9].

The method used to do the Gaussian elimination (mod 2) is described in [16]. The elements of the bit-array are packed in words of 64 bits (on the CYBER 205) or 32 bits (on the NEC SX-2). The elimination operations can then be done very efficiently by XOR-ing with the column vectors of the array. On the NEC SX-2 this XOR-ing proceeds with a speed of 4 words of 32 bits per clock cycle. The total Gaussian elimination step takes less than 0.1% of the total work of the MPQS-algorithm!

4. Results

We have factorized various numbers with our vector computer implementations of the multiple polynomial quadratic sieve. For each number, several preliminary tests were carried out in order to determine optimal values of the parameters B and M. In Table 2 we list the combinations of B and M that we have used for numbers of various sizes, and the corresponding (approximate) size of the factor base.

In the sieving Step 5, we did not sieve with the primes and prime powers < 30 (small prime variation). To compensate for this, the report-threshold was lowered by the value $4\log(2) + 3\log(3) + 2\log(5) + \Sigma_{7 \leqslant p \leqslant 29} \log(p) = 28.476$. This lowering of the report-threshold also has the effect that W-values can be reported which are not fully factorizable over the factor base (large prime variation). Here, this incompletely factorizable part of W can be as large as $\exp(28.476) \approx 2.329 \times 10^{12}$. However, in order to have a reasonable chance to find matches in these "incomplete" W's, we rejected those W's for which the incomplete part exceeded βB, where we took $\beta = 20$ on the CYBER 205 and $\beta = 50$ on the NEC SX-2 (cf. Section 2.2 (iii)). Of the incompletely factorized reported W's we found about 30% yielding at least two coinciding parts, and these 30% generated about 60% of the bit-matrix for the Gaussian elimination. On the NEC SX-2, about 25% of the incomplete W's could be used in this way. This lower percentage was caused by the larger value of β which we used on the NEC SX-2.

Table 2
Combinations of size of the factorized numbers, and B, M and L

Size in decimal digits	M	B	L (approximate)	
48	130000	20000	1000	
54–56	200000	50000	2500	
58–60	200000	60000	3000	
63–65	200000	95000	4600	
67–69	200000	130000	6200	
71–73	200000	140000	6500	
74–77, 82	200000	160000	7400	
77	2500000	300000	13100	on NEC SX-2
87	2500000	450000	18800	on NEC SX-2
92	2500000	600000	24300	on NEC SX-2

Table 3
Cunningham-table [5] factorizations

Number	Prime factorization	CPU-time (hours)
C58 3,288+	P27 * P32	0.2
	P27 = 185901652872784317405136897	
C64 5,149+	P27 * P37	1.2
	P27 = 864203844381482464122519761	
C72 10,108+	P34 * P39	4.3
	P34 = 1726290008991504500177463302688697	
C75 2,542L	P36 * P40	12
	P36 = 104167755499168696693743867494211841	
C77 3,187+	P29 * P49	12
	P29 = 18177792435744585993179560027	
C77 3,300+	P30 * P47	16
	P30 = 438156091706986101113661638401	
C82 7,104+	P29 * P53	70
	P29 = 17712988461899423081645348353	
C87 7,122+	P39 * P49	30 (on NEC SX-2)
	P39 = 369232401898464835382701047039367301441	
C92 6,131−	P34 * P59	95 (on NEC SX-2)
	P34 = 1284827442574221936870974393373403	

We chose the number of g-primes, needed for the generation of W-polynomials, to be fixed on 16, so that we could generate 16×15 new polynomials before having to change these g-primes.

Most of the numbers we have factorized were already attacked, by others, with help of the elliptic curve method, but without success.

So far, we have factorized three (very large) numbers on the NEC SX-2, viz., of 77, 87 and 92 decimal digits. These numbers are explicitly marked in Tables 2, 3 and 6. All other numbers have been factorized on the CYBER 205. Primality of the factors found was proved with the help of Cohen and Lenstra's primality proving program [9].

4.1. Cunningham project

We have factorized several numbers for the Cunningham project [5]. These numbers are denoted here according to the convention used in [5]: e.g., C58 3,288+ means a composite number of 58 decimal digits, which is a cofactor of the number $3^{288}+1$. By 2,542L is meant a cofactor of the so-called Aurifeuillian decomposition of $2^{542}+1$. Table 3 lists the factorizations we have found so far. The C72, C75, C82 and C87 in Table 3 were "more wanted" and the C92 was "most wanted" at the time they were factorized. The C92 was an absolute record, in size, of a number factorized by a general purpose factoring algorithm.

In Table 4 we present, for comparison, a survey of the results obtained by Davis and Holdridge on a CRAY-1S by means of a variant of the quadratic sieve [10].

4.2. Factors of Mersenne numbers

Richard Brent is continuously working on a table of factors of Mersenne numbers, which also provides additional information for a succinct primality proof of the factors given [3]. We have

Table 4
Results of Davis and Holdridge on a CRAY-1S

Number	Prime factorization	CPU-time (in hours)
C53 3,128+	P15 * P19 * P21	6.0
C54 2,212+	P23 * P32	1.0
C55 10,64+	P23 * P33	4.4
C55 5,79−	P15 * P20 * P21	1.0
C58 3,124+	P17 * P41	1.8
C60 2,211−	P20 * P40	22.3
C61 10,67−	P20 * P41	1.2
C67 11,64+	P18 * P49	15.3
C69 2.251−	P21 * P23 * P26	32.3
C71 10,71−	P30 * P41	9.5 (on a CRAY X-MP)

Table 5
Numbers factorized for the table of Mersenne numbers [3]

Number	Prime factorization	CPU-time (in hours)
C63 M503	P22 * P41	0.8
	P22 = 5146314011942914857751	
C64 M709	P27 * P38	0.8
	P27 = 106401034370945865184584169	
C67 M509	P24 * P43	2.2
	P24 = 968224465437705734045581	
C69 M431	P27 * P42	2.9
	P27 = 627565950883854318952353547	
C71 M443	P17 * P55	8.1
	P17 = 26760977129762719	
C73 M389	P20 * P53	12.2
	P20 = 16598743384976073023	

factorized several numbers needed for this table. The results are given in Table 5. By M*n* we mean a number factorized for the Mersenne number $2^n - 1$.

4.3. *Factorizations for proofs of the non-existence of certain odd perfect numbers*

For a proof of the non-existence of odd perfect numbers below 10^{200} certain numbers of the form $\sigma(a^b)$ had to be factorized [4]. Here $\sigma(\cdot)$ denotes the sum of the divisors function. In Table 6 we use the following notation: e.g., C48 2017 $\wedge$ 6 means a cofactor of $\sigma(2017^{16})$ of 48 decimal digits.

5. Conclusions

Our experiments indicate that larger vector computers are very well suited for factoring large integers with help of the quadratic sieve method. Our CYBER 205 program runs about twice as

Table 6
Factorizations for [4]

Number		Prime factorization	CPU-time (in hours)
C48	2017 ∧ 16	P23 ∗ P25 P23 = 72008214963608854098577	0.05
C52	317 ∧ 22	P22 ∗ P30 P22 = 9325995656822900233231	0.09
C54	8170509011431363408568150369 ∧ 2	P19 ∗ P36 P19 = 1019154672897905893	0.10
C55	4591 ∧ 16	P24 ∗ P31 P24 = 955801233000296205155233	0.17
C56	467 ∧ 22	P26 ∗ P30 P26 = 61213091380071615958083811	0.24
C58	613 ∧ 22	P29 ∗ P30 P29 = 25815256247831656853726042407	0.29
C58	1163018639068051 ∧ 4	P22 ∗ P37 P22 = 2039459061440951452061	0.44
C59	4733 ∧ 18	P27 ∗ P32 P27 = 681665903475579942644607417	0.47
C60	17189128703 ∧ 6	P28 ∗ P32 P28 = 3729028273377384104094 21939	0.42
C60	800281 ∧ 10	P29 ∗ P31 P29 = 14630656906581675405799182259	0.36
C64	191 ∧ 30	P19 ∗ P45 P19 = 8510327225640925409	1.1
C65	3823 ∧ 18	P33 ∗ P33 P33 = 153434889660683954432261024327561	1.1
C65	101 ∧ 36	P24 ∗ P42 P24 = 275066876202623893829603	1.2
C66	18041 ∧ 16	P26 ∗ P40 P26 = 93815349051618588433552823	1.7
C69	6073 ∧ 18	P28 ∗ P41 P28 = 4418900065929682250746040969	2.4
C69	612067 ∧ 12	P30 ∗ P39 P30 = 401297819245016618043922143043	5.1

C71	625552508473588471 ∧ 4	P11 ∗ P26 ∗ P35
		P11 = 23851865321
		P26 = 30679951282172311526335631
			5.6

C72	34511 ∧ 16	P24 ∗ P48 P24 = 468717183488261171349569	6.0
C74	20241187 ∧ 10	P29 ∗ P45 P29 = 44191479325025062507848929251	11
C75	2467 ∧ 22	P36 ∗ P40 P36 = 266592662694367677346502147254920281	17

C77	64171 ∧ 16	P23 ∗ P24 ∗ P31
		P23 = 28782215721293361271699
		P24 = 610502384841094120870067
			2.5
			(on NEC SX-2)

fast as the CRAY-1S program of Davis and Holdridge. Our NEC SX-2 program is much faster: about 5–10 times as fast as the CYBER 205 program. Moreover, some tests on the NEC SX-2 with numbers already factorized by Davis and Holdridge indicated that our NEC program is also 5–10 times as fast as their CRAY X-MP program. As a comparison with Bob Silverman's program running on a parallel network of 24 Sun-3 workstations, we mention that, at the time of the writing of this paper, the largest number he had factorized with the MPQS-algorithm was C90 5,160 + [22], and this took him about 15 000 CPU-hours. Each machine therefore took about 625 hours. Not very long ago, a key of 100 decimal digits in the RSA public-key cryptosystem seemed safe. Our results show, that this size should now be lifted at least up to 120–130 decimal digits.

Note added in proof

After we wrote this paper, Arjen Lenstra (Univ. of Chicago and DEC SRC) and Mark Manasse (DEC SRC) established new records for MPQS by first factoring a 93-digit number, and next a 96-digit number. Rather than using one single-CPU computer, they used many Firefly workstations, each of which contains 5 MicroVAX II processors, or 4 CVAX processors. In addition, they received assistance from a few other computers (Sun 3, Sun 4 and VAX 750) from other sites. For the 96-digit number, 95% of the computing work was done on 65 Firefly workstations. This work comprised a total of about 10 CPU years, with an elapsed time of 23 days.

Acknowledgements

The computations have been carried out on the 1-pipe CYBER 205 of SARA (The Academic Computer Centre Amsterdam) and on the NEC SX-2 of the Dutch National Aerospace Laboratory (NLR) at Vollenhove. We are very grateful to the staffs of these two supercomputers for the technical and operational support during this project. We are also grateful to the management of the NLR for the provision of a large amount of free computer time for this project. We are much indebted to the Dutch "Werkgroep Gebruik Supercomputers" (Working Group on the Use of Supercomputers) for the provision of the computer time on the CYBER 205 and for partial financial support for the computer time on the NEC SX-2. Last but not least, we also acknowledge the free computer time which we obtained from CDC Benelux during the initial phase of this project.

References

[1] W.D. Blair, C.B. Lacampagne and J.L. Selfridge, Factoring large numbers on a pocket calculator, *Amer. Math. Monthly* **93** (1986) 802–808.

[2] R.P. Brent, Some integer factorization algorithms using elliptic curves, *Austral. Comput. Sci. Comm.* **8** (1986) 149–163.

[3] R.P. Brent, Table of factors of Mersenne numbers, Draft Report, The Australian National University, 1987.

[4] R.P. Brent, G.L. Cohen and H.J.J. te Riele, An improved technique for lower bounds for odd perfect numbers, Report TR-CS-88-08, The Australian National University, Department of Computer Science, 1988.
[5] J. Brillhart, D.H. Lehmer, J.L. Selfridge, B. Tuckerman and S.S. Wagstaff, Jr., *Factorizations of* $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ *up to High Powers*, American Mathematical Society Contemporary Mathematics Series **22** (AMS, Providence, RI, 2nd ed., 1988).
[6] J. Brillhart, P.L. Montgomery and R.D. Silverman, Tables of Fibonacci and Lucas factorizations, *Math. Comp.* **50** (1988) 251–260.
[7] D.A. Buell, Factoring: Algorithms, computations, and computers, *J. Supercomput.* **1** (1987) 191–216.
[8] Th.R. Caron and R.D. Silverman, Parallel implementation of the quadratic sieve, *J. Supercomput.* **1** (1988) 273–290.
[9] H. Cohen and A.K. Lenstra, Implementation of a new primality test, *Math. Comp.* **48** (1987) 103–121.
[10] J.A. Davis, D.B. Holdridge and G.J. Simmons, Status report on factoring (at the Sandia National Laboratories), in: T. Beth, N. Cot and I. Ingemarsson, Eds., *Advances in Cryptology (Proc. EUROCRYPT 84)* (Springer, Berlin, 1985) 183–215.
[11] C.A. Gantz, R.D. Silverman and S.J. Stuart, A distributed batching system for parallel processing, preprint, 1987.
[12] M. Kraitchik, *Théorie des Nombres, Tome II* (Gauthiers-Villars, Paris, 1926).
[13] H.W. Lenstra, Jr., Factoring integers with elliptic curves, *Ann. of Math. (2)* **126** (1987) 649–673.
[14] W.M. Lioen, H.J.J. te Riele and D.T. Winter, Optimization of the MPQS-factoring algorithm on the Cyber 205 and the NEC SX-2, *Supercomputer* **26** (1988) 42–50.
[15] P.L. Montgomery, Speeding the Pollard and elliptic curve methods of factorization, *Math. Comp.* **48** (1987) 243–264.
[16] D. Parkinson and W. Wunderlich, A compact algorithm for Gaussian elimination over GF(2) implemented on highly parallel computers, *Parallel Comput.* **1** (1984) 65–73.
[17] C. Pomerance, Analysis and comparison of some integer factoring algorithms, in: H.W. Lenstra, Jr. and R. Tijdeman, Eds., *Computational Methods in Number Theory,* Math. Centre Tract **154** (Math. Centre, Amsterdam, 1982) 89–139.
[18] C. Pomerance, J.W. Smith and R. Tuler, A pipeline architecture for factoring large integers with the quadratic sieve algorithm, *SIAM J. Comput.* **17** (1988) 387–403.
[19] H. Riesel, *Prime Numbers and Computer Methods for Factorization* (Birkhäuser, Boston, 1985).
[20] R.L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Comm. ACM* **21** (1978) 120–126.
[21] R.D. Silverman, The multiple polynomial quadratic sieve, *Math. Comp.* **48** (1987) 329–339.
[22] S.S. Wagstaff, Jr., Personal communication, 1988.
[23] S.S. Wagstaff, Jr., Update 2.1 to the Second Edition of [5], 1988.
[24] D.T. Winter, A portable package for multi-precision integer arithmetic, to appear.