MATHEMATICS OF COMPUTATION
VOLUME 43 NUMBER 167
JULY 1984 PAGES 289 311

A Monte Carlo Factoring Algorithm With Linear Storage

By C. P. Schnorr* and H. W. Lenstra, Jr

Abstract We present an algorithm which will factor an integer n quite efficiently if the class number $h(-n)$ is free of large prime divisors The running time $T(n)$ (number of composi tions in the class group) satisfies $\text{prob}[T(m) \leq n^{1/2r}] \geq (r-2)$ ( ) for random $m \in [n/2\ n]$ and $r \geq 2$ So far it is unpredictable which numbers will be factored fast Running the algorithm on all discriminants $-ns$ with $s \leq r^r$ and $r = \sqrt{\ln n/\ln\ln n}$ every composite integer n will be factored in $o(\exp\sqrt{\ln n \ln\ln n})$ bit operations The method requires an amount of storage space which is proportional to the length of the input n In our analysis we assume a lower bound on the frequency of class numbers $h(-m)$, $m \leq n$ which are free of large prime divisors

1. Introduction. The problem of factoring an integer n into its prime power divisors is computationally equivalent to determining all ambiguous, reduced posi tive forms $ax^2 + bxy + cy^2$ (notation (a, b, c)), $a, b, c \in Z$ with *discriminant* $b^2 - 4ac = -n$ ($b^2 - 4ac = +n$, resp) In fact these ambiguous forms correspond to the relatively prime factorizations of n, i e to the pairs (n_1, n_2) with $n = n_1 n_2$, $\gcd(n_1, n_2) = 1$

According to Gauss [5] the equivalence classes of forms with fixed discriminant Δ form a group under composition, the *class group* $G(\Delta)$ The order $h(\Delta)$ of this group is the *class number* Multiplication in $G(\Delta)$ can be done efficiently working with representatives of classes The *ambiguous* classes are the classes H with $H^2 = 1$

In case of negative discriminant $\Delta < 0$ there is a unique reduced form in each class, and this form can be efficiently calculated from any other class representative Therefore, factoring n is computationally equivalent to determining representatives of all ambiguous classes in $G(-n)$ The reduced forms of these classes correspond to the relatively prime factorizations $n_1 n_2 = n$ of n

In case of positive discriminants $\Delta > 0$, under a different concept of reduction there are $O(\sqrt{\Delta}\ \ln \Delta)$ reduced forms in each class They form a cycle under the reduction operation Composition of forms yields a group like structure on the principal cycle A reduced form $(a\ b, c)$ is *ambiguous* if its square under composition yields the unit form Again, the ambiguous, reduced forms with discriminant $n > 0$ correspond to the relatively prime factorizations $n_1 n_2 = n$ of n

Several factoring algorithms have been developed on this basis Here we are only concerned with negative discriminants For positive discriminants see the algorithms of Shanks as described in Monier [11] and Wagstaff-Wunderlich [22] For negative

Received March 21 1983
1980 *Mathematics Subject Classification* Primary 10A30 10C07 68C25
*Research supported by Bundesminister fur Forschung und Technologie under grant 08 30108

discriminants Shanks [17] after guessing generators for $G(-n)$ computes the class number $h(-n)$ by exploiting the group structure Then Shanks computes $H(k) = H^{h(-n)/2^k}$ for the smallest k such that $H(k) \neq 1$ with $H \in G(-n)$ chosen arbitrarily Clearly $H(k)$ is ambiguous Under reasonable assumptions it takes $O(n^{1/4})$ steps to factor n in this way This method can be speeded up to an $O(n^{1/5})$-algorithm by approximating $h(-n)$ via the class formula (let $(\ ,\ )$ denote the Kronecker symbol)

$$\lim_{m} \frac{\sqrt{n}}{\pi} \prod_{p<m} \frac{p}{p-(n,p)} = h(-n),$$

which by the generalized Riemann hypothesis has an error term $O(\ln(mn)n^{1/2}m^{-1})$ For this algorithm the amount of storage will be proportional to the running time

In Schnorr [19] a method was proposed to generate ambiguous forms which is similar to the Morrison-Brillhart factoring algorithm We collect equations

$$H_i^2 = \prod_p H_p^{a_{p,i}} \qquad a_{p,i} \in Z,$$

with $H_i \in G(-n)$ chosen at random and $H_p = [(p, b_p, c_p)]$ for small primes p By combining these equations one obtains

$$H = \prod_i \prod_i H_i^{b_i} \qquad F = \prod_p H_p^{\left(\sum_i b_i a_{p,i}\right)/2}$$

such that $H^2 = F^2$ $H \neq F$ Then HF^{-1} is ambiguous Under reasonable assumptions n will be factored with $o(\exp(\ln n \ln\ln n))$ steps and $o(\exp(\ln n \ln\ln n))$ storage

The new algorithm, given the first t primes $p_1 = 2$ $p_2 = 3$ $p_t = n^{1/2r}$ needs only to store a fixed number of forms which takes $O(\log n)$ bits Let $e_i = \max\{r, p_i^r < p_t^2\}$ Then Stage 1 of the new algorithm computes

$$H = H_0^{\prod_i p_i^{e_i}},$$

for an arbitrarily chosen $H_0 \in G(-n)$ Then compute H^{2^k} for the smallest $k \leq \log \sqrt{n}$ such that $H^{2^k} = 1$ Clearly $H^{2^{k-1}}$ is ambiguous n will be factored by Stage 1 if $h(-n)$ divides $2^k \prod^t p_i^{e_i}$ for some k If Stage 1 fails then Stage 2 does a random walk through the group generated by H

Stage 2 will factor n if $\operatorname{ord}(H) < p_t^2$ for some k We if $h(-n)$ divides $2^k \prod^t p_i q$ for some $q \leq p_t^2$ With $p_t = n^{1/2r}$ Stage 1 of the algorithm takes $O(p_t)$ compositions and for random composite $m \in [0, n]$ with probability $\geq r^{-r}$ detects a proper divisor of m Stage 2 also takes $O(p_t) = O(n^{1/2r})$ compositions and with probability $\geq (r-2)^{(r-2)} r^{-r} \geqslant 2$ detects a proper divisor of m Running Stage 1 on the integers ns for $s \leqslant r^r$, $r = \sqrt{\ln n / \ln\ln n}$ every composite integer n will be factored within $o(\exp \sqrt{\ln n \ln\ln n})$ bit operations The latter bound already takes into account the cost of the arithmetic The cost for a composition in $G(-n)$ is proportional to the cost of the extended Euclidean algorithm which given integers $u, v \geq \sqrt{n}$ computes $r, s \in \mathbf{N}$ with $ru + sv = \gcd(u, v)$ Using standard algorithms for multiplication and division this takes $O(\ln n)^2$ bit operations (e.g. binary Boolean operations, see Knuth [7, 4.5.2, exercise 30 and algorithm X])

The particular features of the new factoring algorithm are

(1) it can easily be operated with $O(\log n)$ bit storage

(2) it is Monte Carlo in the sense that every 1000th integer will be factored about 1000 times faster than average time,

(3) the integers which will be factored very fast are randomly distributed, there is no way to predict whether a given m will be factored fast,

(4) the algorithm is of the parallel type, e.g. 1000 processors will factor 1000 times faster.

Properties (2), (3) seem to endanger the RSA-cryptoscheme, see Rivest et al. [15]. In particular no methods are known that generate class numbers with large prime divisors.

Stages 1 and 2 of the algorithm are presented in Sections 2 and 3. The main algorithm which factors arbitrary integers is given in Section 4. Some computational experience with the factoring algorithm is reported in Section 5. In Appendix I we collect basic theorems and algorithms on quadratic forms. Appendix II contains various tables which demonstrate the performance of the algorithm. We exemplify the distribution of class numbers and integers which are free of large prime divisors, the frequency of class numbers divisible by small primes, and the performance of various pseudo-random functions used in Stage 2 of the algorithm.

2 Stage 1 of the Algorithm. Let n be the integer to be factored. $-n$ is the discriminant of some quadratic form if and only if $-n \equiv 1 \bmod 4$ or $-n \equiv 0 \bmod 4$. The purpose of Stages 1, 2 is to find a nontrivial divisor of n, provided $-n$ is a discriminant and $h(-n)$ is a product of small primes. In order to factor general integers n, the main algorithm in Section 4 applies Stages 1, 2 to multiples ns with $-ns \equiv 0, 1 \bmod 4$. If $-n$ is a discriminant, we can easily construct forms (a, b, c) with discriminant $-n$: choose a small odd prime p with $\left(\frac{-n}{p}\right) = 1$, and solve $b^2 \equiv -n \bmod 4p$, which yields $b = -n + 4pc$ for some $c \in \mathbb{Z}$. Hence (p, b, c) has discriminant $-n$.

Throughout Sections 2, 3 we restrict ourselves to the case $-n \equiv 1 \bmod 4$; consult Theorem III, Appendix I for the case $-n \equiv 0 \bmod 4$. Then the unit $1 \in G(-n)$ is represented by the form $(1, 1, (1+n)/4)$. This ambiguous class yields the improper factorization $1 \cdot n = n$. The other ambiguous classes correspond in a 1-1 way to the relatively prime factorizations of n with nontrivial divisors.

Stage 1. Let $n \in \mathbb{N}$, $n \equiv 1 \bmod 4$, be given.
1. For some $t \in \mathbb{N}$ compute the t first primes $p_1 = 2, p_2 = 3, \ldots, p_t$.
2. choose $H_0 \in G(-n)$ arbitrarily.
3. $H = H_0^{[1]} \cdot p_1^{e_1}$ with $e_1 = \max\{e : p_1^e \leq p_1^*\}$, if $H = 1$, then stop (in this case $\operatorname{ord}(H_0)$ is odd and another H_0 must be chosen).
4. $\overline{H} = H, e_* = \lfloor \log \sqrt{n} \rfloor$.
5. for $\nu = 1, 2, \ldots, e_*$ do [$S = H$; $H = H^{p_\nu}$; if $H = 1$ go to 7].
6. go to *Stage* 2.
7. (at this point S is ambiguous and yields some divisor d of n).

Stage 1 by itself is the core of the new factoring algorithm. The improvements resulting from Stage 2 are important for practical applications, but they scarcely influence the asymptotic time bound of the main algorithm.

Fact 1. Suppose $h(-n) \cdot [1^*, p_t]$, and $\operatorname{ord}(H_0)$ is even; then Stage 1 generates an ambiguous class $S \neq 1$.

In case $-n \equiv 1 \bmod 4$ every ambiguous class $S \neq 1$ yields a proper divisor of n In particular, when n has d odd prime divisors, then $2^{d-1} \mid h(-n)$ and there are exactly 2^{d-1} ambiguous classes corresponding to the 2^{d-1} pairs $\{n_1, n_2\}$ with $n_1 n_2 = n$, $n_1 < n_2$, $\gcd(n_1, n_2) = 1$ Moreover, when n is composite and $H_0 \in G(-n)$ is chosen at random, then prob[ord(H_0) even] $\geq 1/2$ Hence Stage 1 has a chance $\geq 1/2$ to find a proper divisor of n, provided $h(-n) \mid \prod_{i=1}^{t} p_i^{e}$ A few repetitions of Stage 1 almost surely generate a proper divisor of n, provided $h(-n) \mid \prod_{i=1}^{t} p_i^{e}$ and n is composite

Fact 2 Suppose $h(-n) \mid \prod_{i=1}^{t} p_i^{e}$ and n is composite If Stage 1 is passed with H_0 chosen independently k times, then with probability $\geq 1 - 2^{-k}$ a proper divisor of n has been found

Next consider the chance that for random $m \leq n$

$$h(-m) \mid \prod_{i=1}^{t} p_i^{e_i}, \qquad e_i = \max\{\nu : p_i^{\nu} \leq p_i^2\}$$

Siegel [21] proved

$$\forall \varepsilon \ \exists n_\varepsilon \ \forall m \geq n_\varepsilon \ h(-m) \in [m^{1/2-\varepsilon}, m^{1/2+\varepsilon}]$$

We will base the analysis of Stage 1 on the following hypothesis

(2 1)
$$\begin{aligned} &\text{For all } n \text{ and } t \\ &\#\left\{m \leq n \ : \ h(-m) \mid \prod_{i=1}^{t} p_i^{e_i}\right\} / (0.5\, n) \\ &\geq \#\left\{\bar{m} \leq \sqrt{n} \ : \ \bar{m} \mid \prod_{i=1}^{t} p_i^{e_i}\right\} / \sqrt{n} \end{aligned}$$

H G Franke has tested this hypothesis experimentally, see Tables 1–3, Appendix II for $n \approx 4.7 \cdot 10^8$ In general the first term in (2 1) is considerably larger than the second Note that the frequency of class numbers $h(-n)$ of fundamental discriminants (and a fortiori of general discriminants) which are divisible by p is larger than $1/p$ and is close to $1/(p-1)$ provided p is small with respect to n These experimental data and some recent calculations of Cohen and H W Lenstra, Jr indicate that class groups $G(-n)$ of fundamental discriminants are distributed like random Abelian groups of order $O(\sqrt{n} \log n)$ Cohen and Lenstra have calculated prob[m divides $|G|$] for $m = 2, 3,$ and for random Abelian groups, where the probability weight of G is proportional to $1/|\mathrm{Aut}(G)|$ The values of Cohen and Lenstra completely match with our experimental data A corresponding observation with respect to prime discriminants has been made by Leopoldt as cited in Zimmer [23]

Recently Canfield Erdos, and Pomerance improved the theoretical lower bound on the second term in (2 1) We refer in particular to the proof in Pomerance [14]

THEOREM 3 *Let* $\Psi(n, v) = \#\{x \leq n \ : \ x \text{ free of primes} > v\}$ *For every* $\varepsilon > 0$ *there exists* c_ε *such that for all* $n \geq 10$ *and all* r *with* $n^{1/r} \geq (\ln n)^{1+\varepsilon}$ $\Psi(n, n^{1/r})/n \geq (c_\varepsilon r \ln r)^{-r}$

In practice however, $\Psi(n, n^{1/r})/n$ is larger than the bound stated in Theorem 3 From experimental data see Table 2, Appendix II, we conclude

(2 2) for all n and $r \leq \sqrt{\ln n}/\ln\ln n$ $\Psi(n, n^{1/r})/n \geq r^{-r}$

COROLLARY 4 *Assume* (2 2) *and* (2 1) *Then for all* $n \geqslant 10^{20}$ *and all* $p_t = n^{1/2r}$ *with* $r \leqslant \sqrt{\ln n/\ln\ln n}$

$$\#\left\{m \leqslant n \quad h(-m)\Big|\prod_{i=1}^{t} p_i^{e}\right\}/(0\,5n) \geqslant 0\,83\,r^{-r}$$

Proof

$$\#\left\{m \leqslant n \quad h(-m)\Big|\prod_{i=1}^{t} p_i^{e}\right\}/(0\,5n)$$

(2 1)
$$\geqslant \Psi(\sqrt{n}, n^{1/2r})/\sqrt{n} - \sum_{i=1}^{t} p_i^{e-1}$$

(2 2)
$$\geqslant r^{-r} - n^{-1/r_t}$$
$$\geqslant r^{-r} - 1\,1n^{\ 1/r} n^{1/2r}/\ln n^{1/2r} \quad (\text{since } t = \pi(n^{1/2r}) \leqslant 1\,1n^{1/2r}/\ln n^{1/2r})$$
$$\geqslant r^{-r} - 2\,2n^{-1/2r} r/\ln n$$
$$\geqslant r^{-r}(1 - 2\,2r/\ln n)$$
$$(\text{in fact } r \leqslant \sqrt{\ln n/\ln\ln n} \text{ implies } r^{\ r} \geqslant n^{-1/2r})$$
$$\geqslant 0\,83r^{-r} \quad \text{for } n \geqslant 10^{20},\ r \leqslant \sqrt{\ln n/\ln\ln n} \qquad \square$$

Runtime of Stage 1 If H^{p^e} is computed by the binary method (see Knuth [7, §4 63]), this takes

$$2\log_2 p_t^e \leqslant 2\log_2 p_t^2 \leqslant 4\log_2 p_t$$

compositions in $G(-n)$ Since there are about

$$t \leqslant p_t/\ln p_t$$

primes $\leqslant p_t$, this yields a worst case bound of

$$\frac{4}{\ln 2} p_t \approx 5\,8p_t \qquad \text{compositions in total}$$

On the average, the binary method is somewhat more efficient It takes about $1\,5\log p_t^e$ compositions to compute H^{p^e} and therefore Stage 1 will only take about $4\,4p_t$ compositions in total All together we have proved the following

THEOREM 5 *Assume* (2 1), (2 2), *and that for every d* scriminant $m \leqslant n$ (*a single*) $H_0 \in G(-n)$ *in Stage* 1 *is chosen at random Then for all* $n \geqslant 10^{20}$ *and all* $p_t = n^{1/2r}$ *with* $r \leqslant \sqrt{\ln n/\ln\ln n}$ *Stage* 1 *factors at least a* $0\,83nr^{-r}$ *fraction of the discriminants* $\leqslant n$ *and takes about* $4\,4p_t$ *compositions in* $G(-n)$

Remark The discriminants which will be factored are "randomly" distributed in $[0, n]$

For *practical applications* we advise to choose somewhat smaller exponents e_t' instead of the e_t

$$e_t' = \max\{\nu \ \ p_t^{\nu} \leqslant p_t\} \quad \text{with } p_t = n^{1/2r}$$

We used the larger e_i for proving Corollary 4 by a crude argument. Assuming $\Psi(n, n^{1/r})/n \doteq O(r^{-r})$ one obtains Corollary 4 for the e_i':

$$\doteq \left(\#\{m \leq n : h(-m) \mid \prod_{i=1}^{t} p_i^{e_i'}\} \right) / (0.5n)$$

(assuming that (2.1) holds for the e_i')

$$\geq r^{-r} - \sum_{i=2}^{t} p_i^{e_i'-1} \frac{\Psi(n^{1/2-1/2r}, n^{1/2r})}{n^{1/2-1/2r}}$$

$$\doteq r^{-r} - O\left(n^{-1/2r} \frac{n^{1/2r}}{\ln(n^{1/2r})} (r-1)^{-(r-1)} \right)$$

$$\geq r^{-r} - O\left(\frac{2r}{\ln n} (r-1)^{-(r-1)} \right)$$

$$\geq r^{-r} - O\left((r-1)^{-(r-1)} / (r \ln \ln n) \right) = r^{-r}(1 - O(1/\ln \ln n))$$

$$(\text{since } r \leq \sqrt{\ln n / \ln \ln n})$$

The choice of the e_i' are justified by our data in Appendix II. Tables 1, 2 show that there are only a few discriminants $h(-m)$, $m \approx 4.7 \cdot 10^8$, such that $h(-m) = \prod_{i=1}^{t} p_i^{e_i}$ with $e_i > e_i'$ for some $i \geq 2$.

Table 4, Appendix II by Odlyzko considers large integers. This table shows that for $r \leq \sqrt{\ln n / \ln \ln n}$

$$\#\{m \leq n : m \text{ even}, m \mid \operatorname{lcm}(2, 3, \ldots, p_t)\}/n > r^{-r}$$

Note that $m \mid \operatorname{lcm}(2, 3, \ldots, p_t) \Leftrightarrow m = \prod_i p_i^{e_i}$ with $e_i \leq e_i'$ for all i. Hence in practice Corollary 4 even holds when the e_i' are taken for the e_i, and the constant 0.83 can be replaced by some constant > 1. If we use the e_i then Stage 1 will only take about $2.2 p_t$ compositions.

3 Using a Pollard-Brent Recursion in Stage 2. If $h(-n) = 2^e \cdot \prod_{i=2}^{t} p_i^{e_i}$ with $e_i = \lfloor \log n / \log p_i \rfloor$, $e_* = \lfloor \log_2 \sqrt{n} \rfloor$ then Stage 1 fails to factor n and computes

$$\bar{H} = H_0^{\prod_{i=2}^{t} p_i^{e_i}}, \qquad H = \bar{H}^{2^e}.$$

Stage 2 uses H, $\bar{H}$ and will most likely find a proper divisor of n within $O(p_t)$ steps, provided that $\operatorname{ord}(H) \leq p_t^2$ and $\operatorname{ord}(H_0)$ is even.

Stage 2 generates a random walk through the cyclic group $\langle H \rangle$ with generator H. With some function $f: \langle H \rangle \to \langle H \rangle$ let

$$H_1 = H, \qquad H_{i+1} = f(H_i).$$

The function f must be chosen such that

(3.1) f is easy to compute.

(3.2) f is sufficiently random.

(3.3) every relation $H_j = H_k$ with $j \neq k$ yields an ambiguous class S depending on H, f, j, k.

It is known (see Knuth [7, Exercise 3.1.12]) that some $j < k \leq \sqrt{\pi/2}\, p_t$ with $H_j = H_k$ can be expected if f is sufficiently random and $\operatorname{ord}(H) \leq p_t$.

We have two methods to design f and to associate the ambiguous class S to H, f, j, k. Both methods will produce ambiguous classes S with $S \neq 1$ whenever ord(H) is even. Experience must decide which of the methods is more efficient.

Method 1. For some $q \in \mathbf{N}$ choose random integers $a_i \in [p_i^2, 2p_i^2]$ for $i = 1, \ldots, q$. Precompute $F_i = H^{a_i}$, $i = 1, \ldots, q$. For some random function $g: \langle H \rangle \to \{1, \ldots, q\}$ and recursively compute

$$H_1 = H, \qquad H_{i+1} = H_i F_{g(H_i)}.$$

Use the procedure *search* below in order to find some $j < k$ with $H_j = H_k$. Then

$$H_k H_j^{-1} = H^T = 1 \quad \text{with } T = \sum_{i=j}^{k-1} g(H_i).$$

Most likely we will have $k \leqslant 2p_i$ which implies $T \leqslant 4p_i^3$.

Now suppose that ord$(\bar{H}) \equiv 2^e \bmod 2^{e+1}$. We can easily compute c with $T \equiv 2^e c \bmod 2^{e+1}$. Then $\bar{H}^{Tc}$ has order 2^e and yields an ambiguous class

$$S = \bar{H}^{Tc} \qquad \text{with } S \neq 1 \text{ provided } e \geqslant 1.$$

Comment. A theoretical analysis of this method has been done by Sattler and Schnorr [16]. For small values of q, e.g. $q = 2, 3, 4$, the commutativity of the recursion steps increases the number of recursion steps as compared with a pure random recursion $f: \langle H \rangle \to \langle H \rangle$. By experience this slow-down is negligible as soon as q is $\geqslant 16$. We have tested this recursion scheme in class groups (see Table 5, Appendix II) and in cyclic groups $\langle H \rangle = \mathbf{Z}/n\mathbf{Z}$, in particular with n prime. Method 1 even works well for nonrandom a_i like $a_i = c^i$ with c fixed. The advantage of Method 1 over Method 2 is that it explicitly yields a multiple T of the order of H. Also, Method 1 only takes a single group operation (i.e. composition in the case of class groups) per recursion step.

Method 2. Choose a random function $g: \langle H \rangle \to \{1, \ldots, q\}$, choose random values $a_1, \ldots, a_q \in [p_1, 2p_1^2]$ and precompute $F_i = H^{a_i}$, $i = 1, \ldots, q$.

Recursion on H. (We compute $H = H^c$ and d_i with $d_i \equiv c_i \bmod 2^{32}$.)

$$H_1 = H \qquad d_1 = 1,$$

for $i = 1, 2, \ldots$ till *search* finds some $j < k$ with $H_j = H_k$ do

$$(H_{i+1}, d_{i+1}) = \begin{cases} (H^3, 3d \bmod 2^{32}) & \text{if } g(H_i) \leqslant q/2 \\ \left(H_i F_{g(H_i)}, d_i + a_{g(H_i)} \bmod 2^{32}\right) & \text{otherwise} \end{cases}$$

Use the procedure *search* below in order to find some $j < k$ with $H_j = H_k$. Since $H_i = H^{c_i}$ and $H = \bar{H}^{2^e}$, it follows that $\bar{H}^{2^e(c_k - c_j)} = 1$. We compute t such that $d_k - d_j \equiv 2 \bmod 2^{t+1}$. Almost surely t will be less than 32, and this implies $c_k - c_j = 2^t m$ for some odd m. It remains to compute $\bar{H}^m$, since $\bar{H}^{m \cdot 2^r}$ is ambiguous for some $r \leqslant t$. We do not compute m explicitly, but we retrace the above recursion on H. In the following assume $t > 1$. If $t = 0$ then $\bar{H}^n$ can easily be computed from the $g(H_i)$.

Recursion on H. (We compute $H_i = \bar{H}^{\lfloor c_i / 2^t \rfloor}$ and $t_i = c_i \bmod 2^t$ for $t \geqslant 1$.)

$$\overline{H}_1 = 1 \text{ (the unit class)}, \qquad r_1 = 1,$$

for $i = 1, 2, \ldots, k$ do

$$(\overline{H}_{i+1}, r_{i+1}) = \begin{cases} (\overline{H}_i^3 \overline{H}^s, 3r_i - s2^t) \text{ with } s = \lfloor 3r_i/2^t \rfloor & \text{if } g(H_i) \leqslant q/2, \\ (\overline{H}_i \overline{H}^s, r_i + a_{g(H)} - s2^t) & \\ \qquad \text{with } s = \left\lfloor (r_i + a_{g(H)})/2^t \right\rfloor & \text{if } g(H_i) > q/2 \end{cases}$$

It can easily be verified that $\overline{H}_i = \overline{H}^{\lfloor c^i/2 \rfloor}$. Hence $\overline{H}_j \overline{H}_k^{-1} = \overline{H}^{(c^j - c^k)/2} = \overline{H}^m$ with m odd. This yields

Fact 6. Let $\operatorname{ord}(\overline{H}) = 2^e \bmod 2^{e+1}$, $e < 32$, and $H_j H_k^{-1} = 1$, then $\operatorname{ord}(\overline{H}_j \overline{H}_k^{-1}) = 2^e$.

Therefore $S = (\overline{H}_j \overline{H}_k^{-1})^{2^{e-1}}$ is an ambiguous class with $S \neq 1$ whenever $e \neq 0$.

Comment. We have tested Method 2 in class groups and in cyclic groups $\langle H \rangle = \mathbf{Z}/n\mathbf{Z}$, in particular with n prime. For random functions $g: \langle H \rangle \to \{1, \ldots, q\}$ we obtained average values of about $\sqrt{\pi/2}\ \sqrt{n}$ for the smallest index k such that there exists some $j < k$ with $H_j = H_k$, see Table 6, Appendix II. On the average, Method 2 takes 1.5 group operations (i.e. compositions in the case of the class group) per recursion step. A recursion step takes 2 compositions if $H_{i+1} = H_i^3$ and 1 composition if $H_{i+1} = H_i F_{g(H)}$. By reducing the frequency of the $H_{i+1} = H_i^3$-steps the average number of compositions per recursion step can still be reduced. Method 2 also works well with nonrandom a_i like $a_i = c^i$, $i = 1, \ldots, q$, with c fixed. Because of the noncommutativity of the recursion steps, Method 2 works with a smaller number q of multipliers $F_i = H^a$ than Method 1. We successfully applied Method 2 with $q = 4$.

The following pseudo-random function $g: \langle H \rangle \to \{1, \ldots, q\}$ works well for both methods (let (a, b, c) be the reduced form in H):

$$g(H) = \lfloor (b^2 \bmod p)\, q/p \rfloor + 1$$

with p a prime, $q < p < \sqrt{\Delta}$, see Tables 5, 6, Appendix II, for $p = 2^{13} - 1$.

The search for $H_k = H_j$ with $j < k$. Let $H_1 = H$, $H_{i+1} = f(H_i)$. We follow an idea of Brent [1] and do not store all the H but only a fixed number of them. When computing H_i, the stored classes

$$H_{\sigma(\nu)}, \qquad \nu = 1, 2, \ldots, 7,$$

for sufficiently large i, will be such that

$$\sigma(\nu) \approx \sigma(1) 1.1^{\nu}, \qquad \nu = 1, \ldots, 7,$$
$$\text{with } 1.1^7 \sigma(1) < i < 1.1^8 \sigma(1) \approx 2.14 \sigma(1).$$

The recursion for H_i is continued until some $H_k = H_{\sigma(\nu)}$ has been found. The corresponding program looks like:

Search $H_1 = H$, $\sigma(\nu) = 1$ for $\nu = 1, \ldots, 7$, for $i = 2, \ldots$ do

- compute H_i from H_{i-1}
- if $\exists \nu$ $H_{\sigma(\nu)} = H_i$ then [$j = \sigma(\nu)$ $k = i$ stop]
- if $1.1^8 \sigma(1) < i + 1$ then
  - store H_i instead of $H_{\sigma(1)}$
  - $\sigma(\nu) \leftarrow \begin{cases} \sigma(\nu+1) & \text{for } \nu \neq 7 \\ i & \text{for } \nu = 7 \end{cases}$

Let λ be the *period* and μ *the length of the nonperiodic segment* of the sequence H_i, e g

$$H_\mu = H_{\mu+\lambda}, \qquad H_i \neq H_l \quad \text{for } i < l < \mu + \lambda$$

Fact 7 The procedure *search* finds some $j < k$ with $H_k = H_j$ within $\leqslant 1\,1m + \lambda$ recursion steps, $m = \max(\lambda, \mu)$

Proof Since σ increases by the factor 1 1, σ will take some value $\sigma(\nu)$ with $m \leqslant \sigma(\nu) \leqslant 1\,1m$ Hence the for-loop stops, at the latest, with

$$k = \sigma(\nu) + \lambda \leqslant 1\,1m + \lambda, \qquad j = \sigma(\nu),$$

and finds the equality $H_k = H_j$ □

Under the assumption that each of the $\operatorname{ord}(H)^{\operatorname{ord}(H)}$ functions $f\ \langle H\rangle \to \langle H\rangle$ has probability $\operatorname{ord}(H)^{-\operatorname{ord}(H)}$, the stochastic behavior of μ, λ have been well analyzed (see Knuth [7, Exercise 3 1 12])

The expected values of μ and λ are

$$1 + E(\mu) = E(\lambda) \approx \sqrt{\frac{\pi \operatorname{ord}(H)}{8}} + 1/3,$$

$$E(\mu + \lambda) \approx 1\,25\sqrt{\operatorname{ord}(H)} - 1/3,$$

$$\operatorname{Prob}\left[\mu + \lambda \leqslant \sqrt{\frac{\pi}{2}\operatorname{ord}(H)}\right] \approx e^{-\pi/4} \approx 0\,46$$

We conclude from Fact 7 that the number of recursion steps in *search* will be about

$$1\,1\,(E(\mu) + E(\lambda)) \approx 1\,32\sqrt{\operatorname{ord}(H)},$$

provided that $f\ \langle H\rangle \to \langle H\rangle$ is sufficiently random

If in Stage 2 we compute the H_i for $i \leqslant 1\,32\ p_t$, then most likely some relation $H_j = H_k$, $j < k$, will be found, provided $\operatorname{ord}(H) \leqslant p_t^2$ It remains to analyze the chance that $\operatorname{ord}(H) \leqslant p_t^2$ For each prime p, $p_t < p \leqslant p_t^2$, we assume that the frequency of class numbers $h(-m)$, $m \leqslant n$, which are divisible by p is $\geqslant p^{-1}$, and we assume that $h(-m)/p$ factors like random integers of size $\sqrt{n}/p$ By retracing the proof of Corollary 4, we conclude from the assumptions (2 1) (2 2)

(3 4)
$$\begin{aligned}
&\text{For all } r, n, t \text{ with } n \geqslant n_0, p_t = n^{1/2r}, r \leqslant \sqrt{\ln n/\ln\ln n} \\
&\text{and for all primes } p = n^{1/2s} < p_t^2 \\
&\#\left\{m \leqslant n \;\; h(-m) \mid p\prod_{i=1}^{t} p^e\right\}/(0\,5\,n) \\
&\geqslant 0\,83p^{-1}(r - r/s)^{(r-r/s)}
\end{aligned}$$

Summing over all p, $n^{r^{-1}/(1+\varepsilon)} < p < n^{1/r}$, this yields

$$\begin{aligned}
&\#\left\{m \leqslant n \;\; h(-m) \mid p\prod_{i=1}^{t} p_i^{e_i} \text{ with } p < p_t^2\right\}/(0\,5\,n) \\
&\geqslant 0\,83 \sum_{n^{1/(1+\varepsilon)} < p < n^{1/r}} p^{-1}(r - 2/(1+\varepsilon))^{(r-2/(1+\varepsilon))} \\
&\text{(using Theorem 4 27 in Hardy and Wright [6] there follows)} \\
&\geqslant 0\,83\ln(1+\varepsilon)(r - 2/(1+\varepsilon))^{(r-2/(1+\varepsilon))}
\end{aligned}$$

Conclusion. Assume (2.1), (2.2) and that for every discriminant $m \leqslant n$ H_1 in $G(-m)$ is chosen at random. Then $\forall \varepsilon > 0$ $\exists c_\varepsilon > 0$ $\forall n \geqslant n_0$ and all $p_t = n^{1/2r}$, $r \leqslant \sqrt{\ln n / \ln \ln n}$ Stages 1 and 2 with $O(p_t)$ compositions, factor at least $c_\varepsilon (r - 2 + \varepsilon)^{(r-2)/r} n$ discriminants $\leqslant n$.

If one assumes that very large class numbers $h(-n)$ factor like even integers of size $\sqrt{n}$, then we can compare the efficiency of Stages 1 and 2 by Odlyzko's Table 4, Appendix II. The table indicates that for class numbers $h(-m) \approx 10^l$, $l = 15, 20, 25, 30$ the success frequency of Stages 1 and 2 is at least r^{r-2} and is at most er^2 times the success frequency of Stage 1. Note that $(r-2)^{(r-2)}/r^{r-2}$ approaches er^2 for large r.

Remark. There is a well-known deterministic method for doing Stage 2 within $\sqrt{2}\, p_t$ compositions and with $O(p_t)$ storage. The method is explained in Shanks [17, p. 419] in terms of "baby" and "giant" steps. In our situation we can even speed this method by a constant factor if we exploit the fact that $\operatorname{ord}(H)$ will most likely have no prime divisor $\leqslant p_t$.

4. The Main Algorithm. The new algorithm can be used for factoring any composite integer n. We apply Stage 1 to multiples ns of n such that $-ns$ is a discriminant. Here we exploit the observation that class numbers $h(-ns)$ of fundamental discriminants $-ns$ are uncorrelated for distinct values of s. The nonfundamental discriminants $-ns$ should be discarded as far as possible. The discriminant Δ is *fundamental* if

$$\neg \exists w \in \mathbf{N}\ w \neq 1\ \ \Delta / w^2 \text{ is a discriminant}$$

In fact, the class number formula (see Dirichlet [8]),

$$h(-m) = \frac{\sqrt{m}}{\pi} \prod_{p \text{ prime}} \left(1 - \frac{1}{p}\left(\frac{-m}{p}\right)\right)^{-1} \quad \text{for } m < -4$$

implies for $\gcd(w, m) = 1$ and w square free

$$h(-mw^2)/h(-m) = w \prod_{p \mid w} \left(1 - \frac{1}{p}\left(\frac{-m}{p}\right)\right) \Big/ \prod_{p \mid w} \left(p - \left(\frac{-m}{p}\right)\right)$$

Hence for small w, $h(-m)$ and $h(-mw^2)$ have the same large prime divisors provided $\gcd(m, w) = 1$.

Main Algorithm. Let n be the number to be factored and $p_1 = 2$, $p_2 = 3$, $\ldots$, p_t the first t primes, $p_t = n^{1/2r}$ (the appropriate choice of t, r will be determined by the subsequent analysis).

1 $s = 0$
2 take the next s with $\gcd(n, s) = 1$, $ns \equiv 0 \bmod 4$ and $[\exists w \in \mathbf{N}\ w^2 \mid s,\ w \neq 1,\ -ns/w^2 \equiv 0, 1 \bmod 4]$
3 run Stage 1 on ns, which takes $O(p_t)$ compositions. If Stage 1 yields an ambiguous class S then go to 4, otherwise return to 2 and take the next s
4 if S yields a factorization of n then stop, otherwise go to 5
5 return to 3 and repeat Stage 1 on ns with independently chosen classes $H_0 \in G(ns)$ until some factorization of n has been found. In order to prevent that merely useless ambiguous classes are generated, continue to build up the 2-Sylow group $S_2(ns)$ of $G(-ns)$. Use Stage 1 to generate classes in

$S_2(-ns)$ Apply the recursion step of the method below whenever a new $A \in S_2(-ns)$ has been found

The 2-Sylow group $S_2(-ns)$ is a direct product of cyclic groups of order 2^{m_i}, $m_i > 0$ $S_2(-ns) \cong \oplus_{i=1}^{\lambda} \mathbf{Z}/_{2^{m_i}}\mathbf{Z}$ Let λ be the number of cyclic components, then $S_2(-ns)$ has 2^{λ} ambiguous classes Let d be the number of odd prime factors of ns Then by Theorem III, Appendix I, λ is $d-1$, d or $d+1$ depending on the maximal power of 2 which divides ns The ambiguous classes that do not yield a factorization of n form a subgroup $S_2(-ns)$ of $S_2(-ns)$ Let d_n, d_s be the numbers of distinct odd primes of n and s Since $\gcd(n, s) = 1$, we have $d = d_n + d_s$ It follows immediately from Theorem III Appendix I, that the number λ of cyclic components of $S_2(-ns)$ is $\leqslant \lambda - d_n \leqslant d_s + 1$

Constructing $S_2(-ns)$ till a factorization of n is found Given a procedure that generates elements of $S_2(-ns)$ (this will be done by Stage 1) we recursively construct subsets $\{A_1, \quad, A_\lambda\} \subset S_2(-ns), \bar{\lambda} \leqslant \lambda$ such that

$$(4\,1) \qquad |\langle A_1 \quad, A_\lambda \rangle| = \prod_{i=1}^{\lambda} \operatorname{ord}(A_i)$$

Let $\operatorname{ord}(A_i) = 2^{\bar{m}}$, $B_i = A_i^{2^{\bar{m}-1}}$, $\langle A_1, \quad A_\lambda \rangle = \bar{S}_2(-ns)$ Then $\bar{S}_2(-ns) \cong \oplus_{i=1}^{\bar{\lambda}} \mathbf{Z}/_{2^{\bar{m}}}\mathbf{Z}$ and $B_1, \quad, B_{\bar{\lambda}}$ generate the subgroups of ambiguous classes of $\bar{S}_2(-ns)$ After each recursion step either a factorization of n has been found or the new group $\bar{S}_2(-ns)$ will be the subgroup of $S_2(-ns)$ which is generated by the previous $\bar{S}_2(-ns)$ and the element $A \in S_2(-ns)$ obtained in step 1

(4 2) *Algorithm for $S_2(-ns)$*

0 $\bar{\lambda} = 0$ $A_1 = 1$ (= the unit class)
1 generate another $A \in S_2(-ns)$ $A \neq 1$
2 compute $A, A^2 \quad, A^{2^{\bar{m}-1}} \neq 1$ $A^{2^{\bar{m}}} = 1$ and put $B = A^2$
3 if B yields a factorization of n then stop
4 test whether $B \in \langle B_1, \quad B_{\bar{\lambda}} \rangle$, if $B \notin \langle B_1 \quad B_{\bar{\lambda}} \rangle$ then go to 5 else compute $I \subset \{1 \quad, \bar{\lambda}\}$ with $B = \prod_{j \in I} B_j$ and go to 6
5 $A_{\bar{\lambda}+1} = A$ $B_{\bar{\lambda}+1} = B$ $\bar{\lambda} = \bar{\lambda} + 1$ return to 1
6 If $\exists j \in J$ $m_j < m$ then select $j \in J$ with m_j minimal and interchange A with A_j B with B_j, and m with m_j
7 (we have $A^{2^{m-1}} = \prod_{j \in J} A_j^{2^{m_j - 1}}$ $m \leqslant m_j$ for $j \in J$) Put $A = A \prod_{j \in J} A_j^{2^{m_j - m}}$, if $A = 1$ go to 1 else go to 2 (the new m to be computed in 2 will be smaller than the present m since $A^{2^{m-1}} = 1$ holds for the new A)

Run Time Analysis of the Main Algorithm We separately bound
1 the number $T(n)$ of bit operations to be done till some s has been reached with

$$h(-ns) \prod_{i=1}^{t} p_i^{e_i} \qquad e_i = \max\{\nu \mid p_i^{\nu} \leqslant p_t^2\}$$

2 The number $\bar{T}(n)$ of bit operations for building up the 2-Sylow group $S_2(-ns)$ of $G(-ns)$ till a factorization of n is found

1 $T(n)$ We will assume that Corollary 4 extends to multiples of n

$$\exists c, n_0 > 0 \quad \forall m \quad \forall n \geqslant n_0 \quad \forall p_i = (nm)^{1/2r} \text{ with } r \leqslant \sqrt{\ln n/\ln\ln n}$$

$$(4.3) \qquad \#\left\{ns \;\; s \leqslant m \wedge h(-ns) \mid \prod_{i=1}^{t} p_i^{e}\right\}/(0.5m) \geqslant cr^{-r}$$

Our experimental data in fact confirm the lower bound r^{-r} The assumption (4 3) implies

$$\forall n \geqslant n_0 \quad \forall p_i = (n3r^r/c)^{1/2r},\ r \leqslant \sqrt{\ln n/\ln\ln n}$$

$$\exists s \leqslant 3r^r/c \quad h(-ns) \mid \prod_{i=1}^{t} p_i^{e}$$

Since Stage 1 takes $O(p_t)$ compositions, we have

$$T(n) = O\left(p_t r^r (\ln n)^2\right) = O\left(n^{1/2r} r^{r+1/2} (\ln n)^2\right)$$

Here $O(\ln n)^2$ takes into account the costs for the arithmetic We choose $r = \sqrt{\ln n/\ln\ln n}$, $p_t \approx (n3r^r/c)^{1/2r} = O(n^{1/2r}\sqrt{r})$ Then all together (4 3) implies

$$T(n) = o\left(\exp\sqrt{\ln n \ln\ln n}\right)$$

2 $\bar{T}(n)$ In order to factor n we need only to find at most $d_s + 2$ cyclic components of $S_2(-ns)$ If the passes through Stage 1 generate independent elements of $S_2(-ns)$ then k passes of Stage 1 with probability $\geqslant 1 - 2^{-k}$ detect a new cyclic component of $S_2(-ns)$ Hence almost surely we need at most $O(d_s)$ passes through Stage 1, and each pass takes $O(p_t)$ compositions The number of steps for updating the information on $S_2(-ns)$ can be bounded as $O(s)$ the most costly operation in Algorithm (4 2) is to check whether $B \in \langle B_1, \ldots, B_\lambda \rangle$ (step 4) Since $\lambda \leqslant \lambda' \leqslant d_s + 1$ this can be done in a crude way by comparing B with each of the $2^\lambda \leqslant 2^{d_s+1} = O(s)$ elements of $\langle B_1, \ldots, B_{\lambda'} \rangle$ This takes $O(s)$ steps and is sufficient for our purposes We obtain

$$\bar{T}(n) = O\left(d_s(p_t + s)(\ln n)^2\right) = O\left(\log s\,(n^{1/2r}\sqrt{r} + s)(\ln n)^2\right)$$

with $s \leqslant r^r$, $r \leqslant \sqrt{\ln n/\ln\ln n}$ Here again $O(\ln n)^2$ bounds the cost for the arithmetic It follows immediately that $\bar{T}(n) = o(T(n))$

Conclusion If (4 3) holds, then the Main Algorithm, using only Stage 1 takes $o(\exp\sqrt{\ln n \ln\ln n})$ bit operations to factor arbitrary, composite integers n

If we also apply Stage 2, then s will be bounded as $O((r-2)^{(r-2)})$, and this will save a time factor of about $r^2 \approx \ln n/\ln\ln n$

5. Some Computational Experience The new factoring method has been programmed in Fortran on a DEC-1091 at Frankfurt University The core of the algorithm is a subroutine for composition of quadratic forms written in machine language and based on the improved composition method proposed by Seysen [20] The arithmetic operations and the gcd-calculations have been programmed for two-word integers, i e for integers $< 2^{70}$ This means that the program can factor integers $< 2^{130} \approx 10^{39}$ using multipliers $< 2^{10}$ Stage 1 uses the exponents $e_i' = \max\{\nu \mid p_i^\nu \leqslant p_t\}$ Hence the number of compositions per multiplier for Stage 1 is about 2 2 p_t Both methods for Stage 2 have been tested and they are comparable in efficiency

TABLE 0

	$n \approx 10^{38}$	$n \approx 10^{42}$	$n \approx 10^{48}$
p_t	4093	8191	16 381
t = # of primes	563	1027	1899
# compositions per multiplier in Stage 1	10^4	$2 \cdot 10^4$	$4 \cdot 10^4$
in Stage 2	$3 \cdot 10^3$	$3.7 \cdot 10^3$	$3 \cdot 10^4$
average number of multipliers	14.4	18	31
median of the numbers of multipliers	8	9	23
# of integers factored	50	20	20
seconds per composition	$1.4 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	$1.57 \cdot 10^{-3}$
average time for factoring	4.4 min	10.7 min	57 min
median of the factoring time	2.4 min	5.3 min	42 min

The integers n which have been factored for Table 0 are products of two distinct primes p_1, p_2 of nearly the same size. It turns out that the median of the factoring time is considerably smaller than the average factoring time. This is due to a small fraction of integers n which take extremely many multipliers. On the other hand there is a considerable fraction of integers which only take very few multipliers. For instance the seventh Fermat number $F_7 = 2^{2^7} + 1 \approx 3.4 \cdot 10^{38}$ only took 7 multipliers and was factored in about 7 minutes. Here we used $p_t = 16381$ but we run Stage 2 for only 7500 compositions, hence each multiplier took about 1 minute. The multiplier 15 has been successful.

We observed that the factoring method is somewhat faster for integers with more than 2 prime divisors. By our observation class numbers of discriminants with many prime divisors tend to have fewer large prime divisors compared with class numbers of discriminants which are prime or products of two primes. For instance, for $n \approx 10^{30}$, n a product of 5 primes $p \approx 10^6$, the algorithm on the average only took 8.7 multipliers. The median of the number of multipliers has been 5 compared with 14.4 and 8 in Table 0. We have factored a sample of 200 of these integers n.

Appendix 1 on Quadratic Forms. We report classical theorems and algorithms on quadratic forms, see Gauss [5], Mathews [10]. A quadratic form $ax^2 + bxy + cy^2$ with $a, b, c \in Z$ is denoted as $(a\ b\ c)$. Its discriminant is $\Delta = b^2 - 4ac$. $(a\ b\ c)$ is *positive* if $a > 0$, *primitive* if $\gcd(a\ b\ c) = 1$. Two forms $(a\ b\ c)$, $(\tilde{a}\ \tilde{b}\ \tilde{c})$ are equivalent if there exists a linear transformation with integer coefficients and determinant 1 transforming the one form into the other, i.e.

$$I \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} I = \begin{pmatrix} \tilde{a} & \tilde{b}/2 \\ \tilde{b}/2 & \tilde{c} \end{pmatrix}$$

for some integer matrix T with $\det T = 1$. Let $[(a, b, c)]$ be the class represented by (a, b, c). For negative discriminants we always restrict to positive forms.

Two classes $[(a, b, c)]$, $[(\bar{a}, \bar{b}, c)]$ yield a new class $[(A, B, C)]$ by *composition* as follows (for explanation see Lenstra [9]):

$$d = \gcd(a, \bar{a}, (b + \bar{b})/2)$$

Let $\alpha, \beta, \gamma \in Z$ be such that $\alpha a + \beta \bar{a} + \frac{\gamma}{2}(b + \bar{b}) = d$

$$A = a\bar{a}/d^2 \tag{5.1}$$

$$B = \bar{b} + \frac{2\bar{a}}{d}\left[\left(\beta\frac{b - \bar{b}}{2} - \gamma c\right) \bmod \frac{a}{d}\right]$$

$$C = (-\Delta + B^2)/(4A)$$

$[(A, B, C)]$ does not depend on the particular choice for α, β, γ, B, and C. (A, B, C) will be primitive, if (a, b, c) and $(\bar{a}, \bar{b}, c)$ are primitive.

THEOREM 1. *The equivalence classes of primitive quadratic forms with discriminant Δ form an Abelian group $G(\Delta)$ under composition. Its order $h(\Delta)$ is the* ***class number.***

The unit class 1 in $G(\Delta)$ is represented by the form

$$(1, 0, -\Delta/4) \quad \text{if } \Delta \equiv 0 \bmod 4,$$
$$(1, 1, (1 - \Delta)/4) \quad \text{if } \Delta \equiv 1 \bmod 4.$$

The inverse of $[(a, b, c)]$ is $[(a, -b, c)]$.

A class $H \in G(\Delta)$ is *ambiguous* if $H^2 = 1$. The following assertions are equivalent:

(1) H is ambiguous,

(2) every form (a, b, c) in H is equivalent to $(a, -b, c)$,

(3) $T\begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix}T^t = \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix}$ for some integer matrix T with $\det T = -1$,

(4) there is a form (a, b, c) in H with $a \mid b$.

For negative discriminants Δ, classes in $G(\Delta)$ correspond to reduced forms. (a, b, c) is *reduced* if (1) $|b| \leqslant a \leqslant c$ and (2) $b \geqslant 0$ if $|b| = a$ or $a = c$ (i.e. if $[(a, b, c)]$ is ambiguous).

THEOREM 11 (GAUSS [5, Art. 172]). *In every equivalence class with negative discriminant there is exactly one reduced form.*

Gauss also gave a gcd like reduction algorithm which transforms a given form (a, b, c) into an equivalent reduced form.

(5.2) *reduction process for (a, b, c)*

1. find $r \in \mathbf{Z}$, $-|a| < b + 2ra \leqslant |a|$
2. $b = b + 2ra$, $c = (b^2 - \Delta)/(4a)$
3. if (a, b, c) is not reduced then replace (a, b, c) by $(c, -b, a)$ and go to 1.

The reduced forms of ambiguous classes with $\Delta < 0$ are of either of the following types:

(i) $b = 0$ (ii) $a = b$ (iii) $a = c$

These forms are called *ambiguous* Every ambiguous form corresponds to a factorization of the discriminant as

$$\text{(i)}\quad \Delta = -4ac, \qquad \text{(ii)}\quad \Delta = b(b-4c), \qquad \text{(iii)}\quad \Delta = (b-2a)(b+2a)$$

In order to describe this correspondence more precisely, let

$$F(n) = \{(n_1, n_2) \in \mathbf{N}^2 \mid n = n_1 n_2, \gcd(n_1, n_2) = 1\}$$

be the set of relatively prime, ordered factor pairs of $n \in \mathbf{N}$ We have $\#F(n) = 2^d$ where d is the number of distinct prime divisors of n

Let $\Delta \equiv 2^e \bmod 2^{e+1}$, $\Delta < 0$, then the set $A(\Delta)$ of ambiguous forms with discriminant Δ is either in 2 1 or in 1-1 or in 1-2-correspondence with $F(-\Delta/2^e)$

THEOREM III *Let the discriminant $\Delta < 0$ have d odd prime divisors Then the set $A(\Delta)$ of ambiguous (reduced, positive primitive) forms with discriminant $\Delta = -2^e n$ n odd, is obtained from the $(n_1, n_2) \in F(n)$ with $n_1 \leqslant n_2$ as follows*

case $e = 0$ (i e $\Delta \equiv 1 \bmod 4$)	$\#A(\Delta) = \frac{1}{2}\#F(n) = 2^{d-1}$
(i) $(n_1, n_1, (n_1 + n_2)/4)$	*if* $3n_1 < n_2$
(iii) $((n_1 + n_2)/4, (n_2 - n_1)/2\ (n_1 + n_2)/4)$	*if* $3n_1 \geqslant n_2$
case $e = 2$, $n \equiv 1 \bmod 4$, $n \neq 1$	$\#A(\Delta) = \#F(n) = 2^d$
(i) $(n_1, 0, n_2)$	
(ii) $(2n_1, 2n_1, (n_1 + n_2)/2)$	*if* $3n_1 \leqslant n_2$
(iii) $((n_1 + n_2)/2, n_2 - n_1, (n_1 + n_2)/2)$	*if* $3n_1 > n_2$
case $e = 2$, $n \equiv -1 \bmod 4$	$\#A(\Delta) = \frac{1}{2}\#F(n) = 2^{d-1}$
(i) $(n_1, 0, n_2)$	
case $e = 3, 4$	$\#A(\Delta) = \#F(n) = 2^d$
(i) $(\min(n_1 2^{e-2}, n_2), 0, \max(n_1 2^{e-2}, n_2))$	
$(\min(n_2 2^{e-2}, n_1), 0, \max(n_2 2^{e-2}, n_1))$	
case $e \geqslant 5$	$\#A(\Delta) = 2\#F(n) = 2^{d+1}$
(i) $(\min(n_1 2^{e-2}, n_2), 0, \max(n_1 2^{e-2}, n_2))$	
$(\min(n_2 2^{e-2}, n_1), 0, \max(n_2 2^{e-2}, n_1))$	
(ii) $(4, 4, 1 + 2^{e-4} n_1 n_2)$	
(iii) $(2^{e-4} n_2 + n_1, 2^{e-3} n_2 - 2n_1 2^{e-4} n_1 + n_1)$	*if* $3n_1 > n_2 2^{e-4}$
$(2^{e-4} n_1 + n_2, 2^{e-3} n_1 - 2n_2 2^{e-4} n_1 + n_2)$	*if* $3n_1 < n_2 2^{e-4}$

We have listed pairwise inequivalent forms corresponding to distinct positive ambiguous classes They have been arranged according to their types (i), (ii) (iii) as introduced above

Theorem III can easily be obtained from Gauss [5, Art 257–259] Observe that our classes with discriminant $\Delta \equiv 0 \bmod 4$ ($\Delta \equiv 1 \bmod 4$ resp) correspond to primitive Gauss classes with determinant $D = \Delta/4$ (improper primitive Gauss classes with determinant $D = \Delta$, resp) The number of ambiguous classes has also been listed in Cassels [2, p 342]

The Efficiency of Composition An efficient composition algorithm is the main requirement for a satisfactory implementation of our factoring algorithm All

calculations in $G(\Delta)$ are done with reduced forms Composition consists of two parts

1 evaluation of (5.1) $(a, b, c), (\bar{a}\ \bar{b}, \bar{c}) \to (A, B, C)$ (this amounts to an extended gcd-calculation on integers of size $O(\sqrt{|\Delta|})$),

2 reduction of (A, B, C)

If the reduction is done as in (5 2) this corresponds to an extended gcd-calculation on integers of size $O(|\Delta|)$ However, M Seysen [20] found a faster reduction algorithm for this particular situation Reducing (A, B, C) by this algorithm corresponds to only half an extended gcd-calculation on integers of size $O(\sqrt{|\Delta|})$

Appendix II Statistical Tables Table 1 shows the distribution of class numbers $h(-m)$ without large prime divisors for discriminants $-m$ in the interval $I = [-472\ 650\ 003\ \ -472\ 600\ 000]$ There are 25 002 discriminants, the minimal, maximal, and average class numbers are 1518 47 452 and 9 469 77 We put

$$\bar{e}_t(m) = \max\{v \quad p_t^v | h(-m)\}$$

For every prime $p_t = 2, 3, \quad 89$ we record the percentage of those discriminants $-m \in I$ satisfying the following conditions

column 1 $h(-m)$ is free of primes $> p_t$, i e $h(-m) = \prod_{i=1}^{t} p_i^{e_i(m)}$

column 2 for all $i \geq 2$ $e_i(m) \leq e_i = \max\{v \ \ p_i^v \leq p_i^2\}$,

column 3 for all $i \geq 2$ $\bar{e}_i(m) \leq e_i' = \max\{v \ \ p_i^v \leq p_i\}$,

column 4 $h(-m) \mid \prod_{i=1}^{t} p_i^{e_i(m)} \cdot q$ for some $q \leq p_t^2$ q prime

column 5 $h(-m) \mid \prod_{i=2}^{t} p_i^{e_i} \cdot q \quad 2^{e_1(m)}$ for some $q \leq p_t^2$ q prime,

column 6 $h(-m) \mid \prod_{i=2}^{t} p_i^{e_i} \cdot q \quad 2^{\bar{e}_1(m)}$ for some $q \leq p_t^2$, q prime

Moreover we note in

column 7 $r = \ln n/(2 \ln p_t)$

column 8 $10^3 \cdot r \cdot$

Observe that the entries in columns 1–3 of Table 1 are always greater than $10^2 r \cdot$ which confirms Corollary 4 For $r \leq \sqrt{\ln n/\ln\ln n}$ (i e $r \geq 2\ 58$, $p_t \geq 53$) the entries in column 3 are only slightly smaller than those in columns 1, 2 This suggests that Stage 1 should be done with the smaller exponents e_i' instead of the e_i

Table 2 has the same meaning as Table 1 but is restricted to fundamental discriminants in the same interval I Minimal maximal and average class numbers are 1518 47 425 and 10 033 9 There are 15195 fundamental discriminants in I

TABLE 1

p_i	col 1	col 2	col 3	col 4	col 5	col 6	col 7	col 8
2	0 18	0 18	0 18	0 44	0 44	0 44	14 408	000
3	2 08	0 82	0 44	5 39	2 82	1 46	9 090	000
5	5 70	2 70	1 00	18 32	12 55	6 46	6 205	001
7	10 21	7 55	2 00	32 52	28 95	14 45	5 132	023
11	14 43	12 60	6 12	49 58	47 47	35 57	4 165	263
13	18 87	17 64	8 92	58 26	56 95	43 27	3 894	503
17	22 47	21 75	11 51	68 90	68 16	53 20	3 525	1 179
19	26 04	25 55	14 12	73 57	73 08	57 57	3 392	1 586
23	29 28	28 79	16 77	79 60	79 11	63 43	3 185	2 497
29	32 30	31 93	24 73	85 48	85 30	77 38	2 966	3 978
31	34 70	34 52	27 13	87 27	87 09	79 09	2 908	4 484
37	37 00	36 87	29 32	90 69	90 56	82 46	2 766	5 999
41	39 24	39 10	31 46	92 49	92 36	84 25	2 689	6 992
43	41 30	41 16	33 43	93 30	93 16	85 03	2 655	7 480
47	43 18	43 12	35 25	94 43	94 38	86 16	2 594	8 438
53	44 92	44 88	38 93	95 71	95 67	89 58	2 515	9 825
59	46 48	46 45	40 46	96 66	96 63	90 52	2 449	11 147
61	48 02	48 00	41 98	97 02	96 99	90 88	2 429	11 574
67	49 44	49 41	43 38	97 64	97 61	91 50	2 375	12 813
71	50 65	50 63	44 59	98 02	98 00	91 88	2 343	13 606
73	51 58	51 86	45 74	98 13	98 11	91 98	2 328	13 993
79	53 06	53 04	46 97	98 56	98 54	92 41	2 286	15 117
83	54 08	54 07	49 59	98 77	98 76	94 27	2 260	15 837
89	55 12	55 11	50 63	99 00	98 99	94 50	2 225	16 875

TABLE 2

p_i	col 1	col 2	col 3	col 4	col 5	col 6	col 7	col 8
2	0 17	0 17	0 17	0 31	0 31	0 31	14 408	000
3	1 26	0 51	0 31	3 17	1 69	1 00	9 090	000
5	3 69	1 65	0 67	12 03	8 90	4 62	6 205	001
7	6 96	4 98	1 36	25 44	22 53	11 79	5 132	023
11	10 22	8 68	3 96	41 54	39 70	29 83	4 165	263
13	14 02	13 04	6 19	50 71	49 63	37 90	3 894	503
17	17 09	16 46	8 37	61 77	61 11	48 16	3 525	1 179
19	20 14	19 69	10 57	67 04	66 57	53 10	3 392	1 586
23	23 09	22 64	12 89	74 04	73 58	59 89	3 185	2 497
29	25 73	25 55	19 23	81 03	80 85	73 6	2 966	3 978
31	28 21	28 03	21 47	83 22	83 04	75 73	2 908	4 484
37	30 42	30 29	23 57	87 39	87 27	79 86	2 766	5 999
41	32 66	32 53	25 69	89 78	89 65	82 22	2 689	6 992
43	34 68	34 56	27 61	90 82	90 69	83 24	2 655	7 480
47	36 59	36 54	29 44	92 28	92 23	84 69	2 594	8 438
53	38 37	38 33	33 02	94 01	93 97	88 48	2 515	9 825
59	39 94	39 91	34 56	95 26	95 23	89 73	2 449	11 147
61	41 46	41 43	36 06	95 75	95 73	90 22	2 429	11 574
67	42 84	42 81	37 42	96 63	96 60	91 10	2 375	12 813
71	44 02	44 01	38 59	97 16	97 14	91 62	2 343	13 606
73	45 08	45 07	39 63	97 29	97 28	91 75	2 328	13 993
79	46 45	46 44	40 97	97 90	97 89	92 36	2 286	15 117
83	47 60	47 59	43 35	98 17	98 16	93 89	2 260	15 837
89	48 69	48 69	44 44	98 50	98 49	94 23	2 225	16 875

TABLE 3

ν	–	2	3	4	5	6	7	8	9	10
all discr		93 64	51 25	82 53	24 70	48 18	16 92	63 44	20 30	23 18
fund discr		91 83	42 49	78 51	23 94	38 97	16 41	56 90	15 41	21 97
ν	11	12	13	14	15	16	17	18	19	20
all d	9 83	42 42	8 33	15 98	12 58	43 54	6 07	19 09	5 36	20 50
fund	9 35	33 15	8 42	15 16	9 95	36 62	5 96	14 11	5 22	18 85
ν	21	22	23	24	25	26	27	28	29	30
all d	8 77	9 27	4 45	32 64	5 03	7 77	7 15	14 13	3 62	11 85
fund	7 13	8 65	4 48	23 88	4 99	7 67	5 27	13 02	3 67	9 12
ν	31	32	33	34	35	36	37	38	39	40
all d	3 13	27 09	5 10	5 68	4 15	16 90	4 66	5 01	4 26	15 73
fund	3 17	21 38	3 91	5 44	4 00	12 04	2 71	4 76	3 61	13 55
ν	41	42	43	44	45	46	47	48	49	50
all d	2 53	8 30	2 27	8 21	5 10	4 19	2 04	22 33	2 42	4 70
fund	2 64	6 58	2 34	7 40	3 56	4 15	2 13	15 42	2 29	4 57
ν	51	52	53	54	55	56	57	58	59	60
all d	3 12	6 89	1 86	6 74	2 35	10 74	2 75	3 40	1 66	10 43
fund	2 54	6 58	1 95	4 81	2 22	9 44	2 26	3 39	1 70	7 71
ν	61	62	63	64	65	66	67	68	69	70
all d	1 65	2 95	3 46	15 67	2 03	4 84	1 49	4 96	2 16	3 88
fund	1 67	2 95	2 59	11 66	2 03	3 62	1 49	4 57	1 75	3 65
ν	71	72	73	74	75	76	77	78	79	80
all d	1 27	13 10	1 18	2 53	2 53	4 39	1 62	4 02	1 28	10 87
fund	1 27	8 73	1 08	2 53	2 09	4 00	1 53	3 32	1 42	8 65
ν	81	82	83	84	85	86	87	88	89	90
all d	2 45	2 38	1 03	7 33	1 36	2 11	1 82	6 38	1 06	4 81
fund	1 84	2 44	1 18	5 67	1 30	2 12	1 57	5 36	1 12	3 25
ν	91	92	93	94	95	96	97	98	99	100
all d	1 35	3 78	1 64	1 89	1 26	13 85	1 00	2 30	1 99	4 19
fund	1 35	3 63	1 40	1 92	1 17	8 91	1 05	2 12	1 45	3 98

Table 3 shows the percentages of discriminants (fundamental discriminants, resp.) $-m \in I$ such that ν divides $h(-m)$ for $\nu = 2, \ldots, 100$. These percentages are always greater than $100/\nu$, which confirms hypothesis (2.1). For small primes p these frequencies are close to $100/(p-1)$.

Table 4 is due to A. Odlyzko. The entry $a_{l,k}$ in the line starting with l, k and column headed with ν ($\nu = 2k, 8 \le \ldots \le 0$) is the number of integers m from among the first 100,000 even integers $> 10^l$ which have the property that

$$m/\gcd(m, \operatorname{lcm}(1, \ldots, 2^k)) \le 2^{2k}.$$

The last two columns record $r = \ln 10^l / \ln 2^k$ and $10^5 r^{-1}$.

TABLE 4

l	k	$2k$	8	6	4	2	0	r	$10^5 r^{-r}$
15	6	0	0	0	0	0	0	8 305	002
	7	0	0	1	2	2	3	7 118	086
	8	1	2	6	15	36	49	6 229	1 127
	9	6	17	68	129	210	336	5 537	7 674
	10	27	162	316	530	859	1 282	4 983	33 460
	11	110	585	1 043	1 661	2 447	3 445	4 530	106 655
	12	326	1 806	2 805	4 059	5 565	7 388	4 152	270 747
	13	691	4 075	5 854	7 956	10 434	13 307	3 833	579 832
	14	1 425	7 716	10 462	13 538	16 981	20 804	3 559	1090 481
	15	2 416	12 853	16 521	20 531	24 871	29 519	3 322	1853 443
	16	3 852	19 174	23 694	28 542	33 620	39 114	3 114	2907 505
	17	5 691	26 485	31 787	37 273	43 148	49 122	2 931	4276 765
	18	7 979	34 478	40 349	46 620	52 592	58 262	2 768	5968 217
20	9	0	0	0	0	0	0	7 382	039
	10	0	0	1	3	6	17	6 644	344
	11	0	9	22	48	85	120	6 040	1 917
	12	6	70	118	190	271	384	5 537	7 674
	13	25	218	345	510	725	967	5 111	23 945
	14	71	625	907	1 258	1 672	2 182	4 746	61 742
	15	163	1 410	1 937	2 579	3 331	4 258	4 429	137 169
	16	320	2 787	3 689	4 719	5 930	7 353	4 152	270 747
	17	604	4 952	6 257	7 800	9 590	11 639	3 908	485 833
	18	1 019	7 744	9 640	11 764	14 156	16 640	3 691	806 566
25	11	0	0	0	0	0	0	7 550	024
	12	0	2	3	3	3	3	6 921	153
	13	0	7	9	16	21	33	6 388	711
	14	3	35	48	63	93	134	5 932	2 590
	15	8	98	128	182	262	357	5 537	7 674
	16	21	240	339	474	635	809	5 191	19 365
	17	43	567	767	1 014	1 273	1 610	4 885	43 121
	18	105	1 176	1 532	1 911	2 367	2 895	4 614	86 333
30	12	0	0	0	0	0	0	8 305	002
	13	0	0	0	0	0	1	7 666	017
	14	0	0	0	0	1	1	7 118	086
	15	0	1	1	3	5	11	6 644	344
	16	0	9	15	22	35	56	6 229	1 127
	17	0	43	66	94	130	166	5 862	3 144
	18	6	122	170	234	297	375	5 537	7 674
35	15	0	0	0	0	0	0	7 751	013
	16	0	2	3	3	3	3	7 267	055
	17	1	8	8	9	9	13	6 839	195
	18	3	21	22	22	27	34	6 459	585

Table 4 also confirms our assumption (2.2). Note that $a_{l,k,1}$ is the number of integers m among the first 100,000 even integers $> 10^l$ such that

$$m = \prod_i \prod_j p_i^{e_i(j)} \quad \text{with } p_i^{e_i(j)} \leqslant 2^k$$

(which implies $m \mid \prod_{i=1}^{t} p_i^{e_i}$ for the first prime $p_t > 2^k$). The table shows

$$a_{l,k,1} > 10^5 r^{-r} \quad \text{for } r = \ln 10^l / \ln 2^k \leqslant \tfrac{1}{3} \ln 10^l / \ln \ln 10^l$$

This suggests an even stronger assumption than (2.2)

$$\#\left\{m \leqslant n \;\middle|\; m \mid \prod_{i=1}^{t} p_i^{e_i}\right\} / n \geqslant r^{-r}$$

for all n, $r \leqslant \sqrt{\ln n / \ln \ln n}$ and $p_t \leqslant n^{1/r}$. Here $e_t' = \max\{\nu : p_t^\nu \leqslant p_t\}$.

Table 4 can be used to balance Stages 1 and 2. If we factor a discriminant $n \approx 10^{2l}$, then $h(-n)$ will be about 10^l. We choose $p_t \approx 2^k$. Then hypothesis (2.1) suggests that there is some $s \leqslant 10^5 / a_{l,k,\nu}$ with

$$h(-ns) \mid \prod_{i=1}^{t} p_i^{e_i} \cdot q \quad \text{and} \quad q \leqslant 2^{2k-\nu}.$$

Hence Stages 1 and 2 will run on at most $10^5 / a_{l,k,\nu}$ multiples $-ns$. Stage 1 with the exponents e_i takes about $2.2 p_t$ compositions. If we run Stage 2 with Method 2 for $2^{k-\nu/2}$ recursion steps, then Stages 1 and 2 will most likely factor this particular ns (see Table 6, Appendix II, for the performance of Method 2 in Stage 2). In this way Stage 2 takes about $1.5\, 2^{k-\nu/2}$ compositions. Therefore the total number of compositions of the Main Algorithm will be bounded by

$$b_{l,k,\nu} = \frac{10^5}{a_{l,k,\nu}} 2^k (2.2 + 1.5\, 2^{-\nu/2}).$$

Examples. $n \approx 10^{30}$. Choose $k = 12$, $\nu = 0$. $a_{15,12,0} = 7.388$. We have $\lfloor b_{15,12,0} \rfloor = 205132$, and n will be factored in about $2 \cdot 10^5 \approx n^{0.18}$ compositions.

$n \approx 10^{40}$. Choose $k = 14$, $\nu = 0$. $a_{20,14,0} = 2.182$. We have $\lfloor b_{20,14,0} \rfloor = 2778221$, and n will be factored in about $2.8 \cdot 10^6 \approx n^{0.14}$ compositions.

$n \approx 10^{50}$. Choose $k = 17$, $\nu = 0$. $a_{25,17,0} = 1.610$. We have $\lfloor b_{25,17,0} \rfloor = 30122136$, and n will be factored in about $2.9 \cdot 10^7 \approx n^{0.15}$ compositions.

$n \approx 10^{60}$. Choose $k = 18$, $\nu = 0$. $a_{30,18,0} = 375$. We have $\lfloor b_{30,18,0} \rfloor = 258648746$, and n will be factored in about $2.6 \cdot 10^8 \approx n^{0.14}$ compositions.

The examples show that the number of compositions while factoring n is smaller than $\exp \sqrt{\ln n \ln \ln n}$. For instance, for $n \approx 10^{60}$ we have $2.5 \cdot 10^8 \approx 0.00116 \exp \sqrt{\ln n \ln \ln n}$. The examples indicate that our algorithm will be faster on integers $n > 10^{40}$ than the Morrison-Brillhart algorithm. Wunderlich [22] reports that the Morrison-Brillhart algorithm for $n \approx 10^{40}$ takes about $322\, n^{0.152} \approx 3.8 \cdot 10^8 \approx n^{0.21}$ divisions of Q_t, $Q_t = O(\sqrt{n})$, by small primes p. Meanwhile the above estimations have been verified by a program running on the DEC 1091 in Frankfurt, see Section 5.

Table 5 demonstrates the performance of Method 1 of Stage 2. We choose the pseudo-random function g, $G(\Delta) \to \{1, \ldots, 16\}$

$$g(H) = \left\lfloor \left[b^2 \bmod (2^{13} - 1) \right] 16 / (2^{13} - 1) \right\rfloor + 1$$

where (a, b, c) is the reduced form corresponding to H. We consider the method with 6 distinct samples of exponents $a_1, \ldots, a_{16}$: three samples with a_i chosen at random and three samples with regular a, $a_i = c^{(i-1)!} \bmod 2^{70}$, $i = 1, \ldots, 16$ with $c = 2, 3, 5$. We have the recursion

$$H_{i+1} = H_i H_0^{g(H_i)}.$$

TABLE 5

10	10	10^{10}	10^{12}	10^{14}
sample 1	29	84	252	698
	16	47	99	281
	71	176	535	1401
	49	109	332	783
sample 2	28	87	237	605
	19	56	130	241
	53	171	531	1314
	38	109	413	930
sample 3	28	88	219	606
	20	49	117	266
	57	184	469	1501
	45	130	402	982
$a_i = 2^{31} \bmod 2^{71}$	19	61	176	504
	9	29	98	269
	44	121	319	1010
	31	85	163	692
$a_i = 3^{31} \bmod 2$	31	65	211	585
	20	28	102	380
	57	168	477	1455
	40	111	385	945
$a_i = 5^{31} \bmod 2^{71}$	28	79	217	819
	20	56	152	388
	59	174	480	1733
	44	113	414	1013

with $H_0 \in G(\Delta)$ chosen at random. We apply this recursion to the 50 largest discriminants $\Delta < -10^m$ with $\Delta \equiv 1 \bmod 4$ for $m = 8, 10, 12, 14$.

For every sample the 10^m-column records four values:

1. the average period length,
2. the median of the period lengths,
3. the average number of recursion steps till *search* finds some k with $\exists j < k$, $H_j = H_k$,
4. the median of the number of recursion steps.

The particular favorable performance of $a_i = 2^{i+31} \bmod 2^{70}$ and of $a_i = 3^{i+31} \bmod 70$ can be explained by the fact that the order of most class groups is even and is a multiple of 3 for about half of the class groups. Despite this favorable performance for random class groups the choice of $a_i = 2^{i+31} \bmod 2^{70}$ is unfavorable for the factoring algorithm since in the particular situation of Stage 2 the class numbers are free of small prime divisors.

Table 6 shows the performance of Method 2 in Stage 2 for the pseudo-random function $g: G(\Delta) \to \{1, \ldots, 4\}$

$$g(H) = \left\lfloor \left[b^7 \bmod (2^{13} - 1) \right] 4 / (2^{13} - 1) \right\rfloor + 1$$

where (a, b, c) is the reduced form corresponding to H. We used $a_i = d^i$, $i = 3, 4$, with constants $d = 2, 3, \ldots, 8$. The recursion scheme is

$$H_{i+1} = \begin{cases} H_i^3 & \text{if } g(H_i) \leqslant 2, \\ H_i H_0^{d^{g(H_i)}} & \text{otherwise,} \end{cases}$$

$H_0 \in G(\Delta)$ is chosen at random

For every $d = 2, 3, \ldots, 8$ and $m = 8, \ldots, 14$ we applied this recursion to the 50 largest discriminants $\Delta \approx -10^m$ with $\Delta \equiv 1 \bmod 4$. For every d and m the table records four values

1. the average period length
2. the median of the period lengths,
3. the average number of recursion steps till *search* finds some k with $\exists j < k$ $H_j = H_k$,
4. the median of the number of recursion steps

TABLE 6

base d	10^8	10^{10}	10^{12}	10^{14}
2	21	63	178	684
	12	42	112	289
	51	158	395	1513
	43	92	322	1099
3	28	116	231	783
	17	53	86	555
	63	219	503	1519
	50	157	260	1118
4	24	76	185	780
	15	41	83	277
	48	165	360	1260
	31	123	285	698
5	34	103	273	746
	22	58	143	429
	74	224	570	1613
	50	161	357	1013
6	23	43	98	435
	12	24	79	196
	55	114	299	927
	35	72	188	684
7	28	90	276	713
	24	74	154	376
	63	224	516	1533
	51	159	386	925
8	26	76	209	763
	13	47	129	445
	53	144	391	1449
	41	94	276	918

Acknowledgement Several members of the Frankfurt working group participated in implementing this algorithm. H. G. Franke produced Tables 1–3 and implemented large integer arithmetic and an efficient composition algorithm in machine language. J. Sattler produced Tables 5–6 which compare Methods 1–2 of Stage 2.

Thanks are due to the computer centres of Frankfurt and Amsterdam university for providing computing time on the DEC-1091 in Frankfurt and the CDC-Cyber in Amsterdam We also like to thank A Odlyzko for the permission to include in this paper Table 4 which is part of a larger statistic made at Bell Laboratories

Fachbereich Mathematik
Universitat Frankfurt
6 Frankfurt am Main West Germany

Mathematisch Instituut
Universiteit Amsterdam
1018 WB Amsterdam The Netherlands

1 R P BRENT An improved Monte Carlo factorization algorithm *BIT* v 20 1980 pp 176–184
2 J W S CASSELS *Rational Quadratic Forms* Academic Press London New York 1978
3 H COHEN & H W LENSTRA JR *Primality by Small Primes of Class Numbers* Personal communication 1982
4 J D DIXON Asymptotically fast factorization of integers *Math Comp* v 36 1981 pp 255–260
5 C F GAUSS *Disquisitiones Arithmeticae* Leipzig 1801 English transl by A A Clarke Yale University Press New Haven and London 1966
6 G H HARDY & E M WRIGHT *An Introduction to the Theory of Numbers* 5th ed Oxford Univ Press Oxford 1979
7 D E KNUTH *The Art of Computer Programming* Vol 2 *Seminumerical Algorithms* 2nd ed Addison Wesley Reading Mass 1981
8 P G LEJEUNE DIRICHLET & R DEDEKIND *Vorlesungen uber Zahlentheorie* Braunschweig 1893 reprint New York 1968
9 H W LENSTRA JR *On the Calculation of Regulators and Class Numbers of Quadratic Fields* Journees Arithmetiques 1980 (J V Armitage Ed) Cambridge Univ Press Oxford 1982 pp 123–150
10 G B MATHEWS *Theory of numbers* 1892 Reprint Chelsea New York 1962
11 L MONIER *Algorithmes de Factorisation et Entiers* These d'informatique Univ Paris Sud 1980
12 M A MORRISON & J BRILLHART A method of factorization and the factorization of F_7 *Math Comp* v 29 1975 pp 183–205
13 J M POLLARD A Monte Carlo method for factorization *BIT* v 15 1975 pp 331–334
14 C POMERANCE Analysis and comparison of some integer factoring algorithms *Computational Methods in Number Theory* (R Tijdeman and H Lenstra Eds) Mathematisch Centrum Amsterdam 1981
15 R L RIVEST A SHAMIR & L ADLEMAN A method for obtaining digital signatures and public key cryptosystems *Comm ACM* v 21 1978 pp 120–126
16 J SATTLER & C P SCHNORR Ein Effizienzvergleich der Faktorisierungsverfahren von Morrison Brillhart und Schroeppel *Computing* v 30 1983 pp 91–110
17 D SHANKS Class Number, A Theory of Factorization and Genera Proc Symp Pure Math vol 20 Amer Math Soc Providence R I 1971 pp 415–440
18 J SATTLER & C P SCHNORR *Generating Random Walks in Groups* Preprint Universität Frankfurt 1983 submitted for publication
19 C P SCHNORR Refined analysis and improvements on some factoring algorithms *J Algorithms* v 3 1982 pp 101–127
20 C P SCHNORR & M SEYSEN *An Improved Computation Algorithm* Preprint Universität Frankfurt 1982 submitted for publication
21 C L SIEGEL Über die Klassenzahl quadratischer Zahlkörper *Acta Arith* v 1 1936 pp 83–86
22 S S WAGSTAFF & M C WUNDERLICH *A Comparison of Factorization Methods* Unpublished manuscript
23 H G ZIMMER *Computational Problems, Methods and Results in Algebraic Number Theory* Lecture Notes in Math Vol 262 Springer Berlin and New York 1972