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 compositions in the class group) satisfies $\text{prob}[T(m) \leq n^{1/2r}] \geq (r - 2)^{-r/2}$ for random $m \in \left[ n/2, n \right]$ and $r \geq 2$. So far it is unpredictable which numbers will be factored fast. Running the algorithm on all discriminants $-ns$ with $s \leq r'$ and $r = \sqrt{\ln n / \ln \ln n}$, every composite integer $n$ will be factored in $o(\exp \ln n \ln \ln n)$ 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 positive forms $ax^2 + bxy + cy^2$ (notation $(a, b, c)$), $a, b, c \in \mathbb{Z}$, with discriminant $b^2 - 4ac = -n$ (for $b^2 - 4ac = +n$, 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 $A$ form a group under composition, the class group $G(A)$. The order $h(A)$ of this group is the class number. Multiplication in $G(A)$ 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 $A < 0$ 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 $A > 0$, under a different concept of reduction, there are $O(\sqrt{A} \ln A)$ 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
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 \((\frac{\alpha}{\beta})\) denote the Kronecker symbol):

\[
\lim_{m \to \infty} \frac{\sqrt{n}}{m} \prod_{p \leq m} \frac{p}{p - \left(\frac{-n}{p}\right)} = h(-n),
\]

which by the generalized Riemann hypothesis has an error term \(O(\ln(mn)n^{1/2}/m^{1/2})\).

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

\[
H^2 = \prod_{p} H_p^{a_p}, \quad a_{p,i} \in \mathbb{Z},
\]

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

\[
H = \prod_i H_i^{h_i}, \quad F = \prod_p H_p^{(\Sigma_i a_{p,i})/2}
\]

such that \(H^2 = F^2, H \neq F\). Then \(HF^{-1}\) is ambiguous. Under reasonable assumptions \(n\) will be factored with \(o(\exp(\sqrt{\frac{1}{2}} \ln n \ln \ln n))\) steps and \(o(\exp(\frac{1}{3} \ln n \ln \ln n))\) storage.

The new algorithm, given the first \(t\) primes \(p_1 = 2, p_2 = 3, \ldots, p_t = n^{1/2r}\), needs only to store a fixed number of forms which takes \(O(\log n)\) bits. Let \(e_1 = \max\{\nu: p_1^\nu \leq p_2^2\}\). Then Stage 1 of the new algorithm computes

\[
H = H_0^{\prod_{i=2}^t p_i^e_i}
\]

for an arbitrarily chosen \(H_0 \in G(-n)\). Then compute \(H^{2^k}\) for the smallest \(k < \log_2 \sqrt{n}\) such that \(H^{2^k} = 1\). Clearly \(H^{2^{k-1}}\) is ambiguous. \(n\) will be factored by Stage 1 if \(h(-n)\) divides \(2^k \prod_{i=2}^t p_i^{e_i}\) 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 \(\text{ord}(H^{2^k}) < p_2^2\) for some \(k\), i.e. if \(h(-n)\) divides \(2^k \prod_{i=2}^t p_i^{e_i} q\) for some \(q \leq p_2^2\). 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 \(\geq (r-2)^{r-2}, r \geq 2\), detects a proper divisor of \(m\). Running Stage 1 on the integers \(ns\) for \(s \leq r' = \sqrt{\ln n / \ln \ln n}\), 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 \(u, v \leq \sqrt{n}\) computes \(r, s \in \mathbb{N}\) with \(ru + sv = \gcd(u, v)\). Using standard algorithms for multiplication and division this takes \(O(\ln n)^2\) bit operations, i.e. 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 \(-n \equiv 1 \mod 4\) or \(-n \equiv 0 \mod 4\). 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 \(-ns \equiv 0, 1 \mod 4\). If \(-n\) is a discriminant, we can easily construct forms \((a, b, c)\) with discriminant \(-n\): choose a small odd prime \( p \) with \((\frac{-n}{p}) = 1\) and solve \( b^2 \equiv -n \mod 4p \) which yields \( b^2 = -n + 4pc \) for some \( c \in \mathbb{Z} \). Hence \((p, b, c)\) has discriminant \(-n\).

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

Stage 1. Let \( n \in \mathbb{N}, -n \equiv 1 \mod 4 \), 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. \( H := H_0^{11-p_1} \) with \( e := \max \{ v: v \leq p_i^2 \} \); if \( H = 1 \), then stop (in this case \( \text{ord}(H_0) \) is odd and another \( H_0 \) must be chosen),
4. \( H := H, e_0 := \left[ \log_2 \sqrt{n} \right] \),
5. for \( v = 1, 2, \ldots, e_0 \) do \([S := H, H := H^2, \text{if } H = 1 \text{ go to 7}],\)
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 \( h(-n) \prod_{i=1}^{r} p_i^e \) and \( \text{ord}(H_0) \) is even, then Stage 1 generates an ambiguous class \( S \neq 1 \).
In case $-n \equiv 1 \mod 4$ every ambiguous class $S \neq 1$ yields a proper divisor of $n$. In particular, when $n$ has $d$ odd prime divisors, then $2^{d-1} | h(-n)$, 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 $\text{prob}[\text{ord}(H_0) \text{ even}] \geq 1/2$. Hence Stage 1 has a chance $\geq 1/2$ to find a proper divisor of $n$, provided $h(-n) | \Pi_{i=1}^t p_i^{e_i}$ and $n$ is composite.

**Fact 2.** Suppose $h(-n) | \Pi_{i=1}^t p_i^{e_i}$ 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 $m \leq n$:

$$h(-m) \prod_{i=1}^t p_i^{e_i}, \quad e_i = \max\{v : p_i^v \leq p_i^2\}.$$ 

Siegel [21] proved:

$$\forall \epsilon : \exists n : \forall m \geq n : h(-m) \in [m^{1/2-\epsilon}, m^{1/2+\epsilon}].$$

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

For all $n$ and $t$:

$$\# \left\{ m \leq n : h(-m) \prod_{i=1}^t p_i^{e_i} \right\} / (0.5n)$$

(2.1) \[ \geq \# \left\{ m \leq \sqrt{n} : \prod_{i=1}^t p_i^{e_i} \right\} / \sqrt{n} .

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 $\text{prob}[m \text{ divides } |G|]$ for $m = 2, 3, \ldots$ and for random Abelian groups, where the probability weight of $G$ is proportional to $1/|\text{Aut}(G)|$. 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, Erdős, 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 $\Psi(n, v) := \# \left\{ x \leq n : x \text{ free of primes } > v \right\}$. For every $\epsilon > 0$ there exists $c_\epsilon$ such that for all $n \geq 10$ and all $r$ with $n^{1/r} \geq (\ln n)^{1+\epsilon}$: $\Psi(n, n^{1/r})/n \geq (c_\epsilon r \ln r)^{-\epsilon}$.

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) \[ \quad \text{for all } n \text{ and } r \leq \sqrt{\ln n / \ln \ln n} : \Psi(n, n^{1/r})/n \geq r^{-\epsilon}. \]
Corollary 4. Assume (2.2) and (2.1). Then for all \( n \geq 10^{20} \) and all \( p_t = n^{1/2r} \) with \( r \leq \sqrt{\ln n / \ln \ln n} \):

\[
\# \left\{ m \leq n : h(-m) \prod_{i=1}^{t} p_i^{e_i} \right\} / (0.5n) \geq 0.83 r^{-r}.
\]

Proof.

\[
\# \left\{ m \leq n : h(-m) \prod_{i=1}^{t} p_i^{e_i} \right\} / (0.5n)
\] (2.1)

\[
\geq \Psi(\sqrt{n}, n^{1/2r}) / \sqrt{n} - \sum_{i=1}^{t} p_i^{e_i-1}
\] (2.2)

\[
\geq r^{-r} - n^{-1/r},
\]

\[
\geq r^{-r} - 1.1 n^{-1/r} n^{1/2r} / \ln n^{1/2r} \quad (\text{since } t = \pi(n^{1/2r}) \leq 1.1 n^{1/2r} / \ln n^{1/2r})
\]

\[
\geq r^{-r} - 2.2 n^{-1/2r} / \ln n
\]

\[
\geq r^{-r} (1 - 2.2 r / \ln n)
\]

(in fact \( r \leq \sqrt{\ln n / \ln \ln n} \) implies \( r^{-r} \geq n^{-1/2r} \))

\[
\geq 0.83 r^{-r} \quad \text{for } n \geq 10^{20}, r \leq \sqrt{\ln n / \ln \ln n}. \quad \Box
\]

Runtime of Stage 1. If \( H^{p_t} \) is computed by the binary method (see Knuth [7, §4.63]), this takes

\[
2 \log_2 p_t^{e_i} \leq 2 \log_2 p_t^2 \leq 4 \log_2 p_t
\]

compositions in \( G(-n) \). Since there are about

\[
t \leq p_t / \ln p_t
\]

primes \( \leq p_t \), this yields a worst case bound of

\[
\frac{4}{\ln 2} p_t \approx 5.8 p_t \quad \text{compositions in total.}
\]

On the average, the binary method is somewhat more efficient. It takes about

\[
1.5 \log p_t^{e_i} \text{ compositions to compute } H^{p_t} \text{ and therefore Stage 1 will only take about } 4.4 \text{ compositions in total.}
\]

All together we have proved the following

\[
\text{Theorem 5. Assume (2.1), (2.2), and that for every discriminant } m \leq n \text{ (a single) } H_0 \in G(-n) \text{ in Stage 1 is chosen at random. Then for all } n \geq 10^{20} \text{ and all } p_t = n^{1/2r} \text{ with } r \leq \sqrt{\ln n / \ln \ln n} : \text{ Stage 1 factors at least a } 0.83 n r^{-r} \text{ fraction of the discriminants } \leq n \text{ and takes about } 4.4 \text{ compositions in } G(-n).
\]

Remark. The discriminants which will be factored are “randomly” distributed in \([0, n]\).

For practical applications we advise to choose somewhat smaller exponents \( e_i' \) instead of the \( e_i \):

\[
e_i' := \max \{ \nu : p_i^\nu \leq p_t \} \quad \text{with } p_t = n^{1/2r}.
\]
We used the larger $e_i$ for proving Corollary 4 by a crude argument. Assuming 
$\Phi(n, n^{1/r})/n = O(r^{-r})$ one obtains Corollary 4 for the $e'_i$:

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

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

$$r^{-r} - \sum_{i=1}^{t} p_i^{-e'_i} \psi\left( \frac{n^{1/2-1/2r}, n^{1/2r}}{n^{1/2-1/2r}} \right)$$

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

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

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

(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 $h(-m), m \approx 4.7 \times 10^8$, such that $h(-m) = \prod_{i=1}^{t} p_i^{\bar{e}_i}$ with $\bar{e}_i > e'_i$ for some $i \geq 2$.

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

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

Note that $m | \text{lcm}(2, 3, \ldots, p_t)$ if and only if $m = \prod_i p_i^{\bar{e}_i}$ with $\bar{e}_i \leq e'_i$ 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 $p_r$ compositions.

3. Using a Pollard-Brent Recursion in Stage 2. If $h(-n) + 2^{e'_*} \prod_{i=2}^{t} p_i^{e'_i} \text{ with } e'_i = \left\lfloor \log_2 n/\log_2 p_i \right\rfloor, e'_* = \left\lfloor \log_2 \sqrt{n} \right\rfloor$, then Stage 1 fails to factor $n$ and computes

$$\bar{H} := H_0^{\prod_{i=2}^{t} p_i^{e'_i}}, \quad H := \bar{H}^{2^{e'_*}}.$$  

Stage 2 uses $H, \bar{H}$ and will most likely find a proper divisor of $n$ within $O(p_2)$ steps, provided that $\text{ord}(H) \leq p_2$ and $\text{ord}(H_0)$ 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

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

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 $j < k \leq \sqrt{\pi/2} \times p_i$ with $H_j = H_k$ can be expected if $f$ is sufficiently random and $\text{ord}(H) \leq p_i^2$.  

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use
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 \( \text{ord}(H) \) is even. Experience must decide which of the methods is more efficient.

**Method 1.** For some \( q \in \mathbb{N} \) choose random integers \( a_i \in [p^2, 2p^2] \) 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:

\[
H_1 = H, \quad H_{i+1} = H_i F_{g(H_i)}.
\]

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 \( k \leq 2p \) which implies \( T \leq 4p^2 \).

Now suppose that \( \text{ord}(H) = 2^e \mod 2^{e+1} \). We can easily compute \( e \) with \( T = 2^e \mod 2^{e+1} \). Then \( H^{2^{e+1}} \) has order \( 2^e \) and yields an ambiguous class

\[
S := H^{2^{e+1}} \quad \text{with } S \neq 1 \text{ provided } e > 1.
\]

**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 \( \geq 16 \). We have tested this recursion scheme in class groups (see Table 5, Appendix II) and in cyclic groups \( \langle H \rangle = \mathbb{Z}/n\mathbb{Z} \), 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 \( a_1, \ldots, a_q \in [p^2, 2p^2] \) and precompute \( F_i := H^{a_i}, i = 1, \ldots, q \).

**Recursion on \( H \).** (We compute \( H_i = H^{c_i} \) and \( d_i \) with \( d_i \equiv c_i \mod 2^{32} \).)

\[
H_1 := H, \quad d_1 := 1;
\]

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

\[
(H_{i+1}, d_{i+1}) := \begin{cases} 
(H_i^3, 3d_i \mod 2^{32}) & \text{if } g(H_i) < q/2, \\
(H_i F_{g(H_i)}, d_i + a_{g(H_i)} \mod 2^{32}) & \text{otherwise}.
\end{cases}
\]

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

**Recursion on \( H \).** (We compute \( H_i = H^\left\lfloor c_i/2^t \right\rfloor \) and \( r_i = c_i \mod 2^t \) for \( t \geq 1 \).)
for $i = 1, 2, \ldots, k$ do
\[
\begin{aligned}
(H_i; H_i^*) \text{ with } s = \left\lfloor 3r_i/2^i \right\rfloor & & \text{if } g(H_i) \leq q/2, \\
(H_i; H_i^*) \text{ with } s = \left\lfloor (r_i + a g(H_i))/2^i \right\rfloor & & \text{if } g(H_i) > q/2.
\end{aligned}
\]

It can easily be verified that $H_i = H_{\lfloor c_i/2^i \rfloor}$. Hence $H_j H_k^{-1} = H_{\lfloor c_j - c_k \rfloor/2^i} = H_m$ with $m$ odd. This yields

**Fact 6.** Let $\text{ord}(H) = 2^e \mod 2^{e+1}$, $e < 32$, and $H_j H_k^{-1} = 1$, then $\text{ord}(H_j H_k^{-1}) = 2^e$.

Therefore $S = (H_j H_k^{-1})^{2^{e-1}}$ 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 $\langle H \rangle = \mathbb{Z} / n \mathbb{Z}$, in particular with $n$ prime. For random functions $g: \langle H \rangle \rightarrow \{1, \ldots, q\}$ we obtained average values of about $\sqrt{n}/\sqrt{2}$ 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 $H_{i+1} = H_i F g(H_i)$. 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 $F_i = H^a_i$ than Method 1. We successfully applied Method 2 with $q = 4$.

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

\[
g(H) = \left\lceil (b^2 \mod p) q/p \right\rceil + 1
\]

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

\[
H_{a(\nu)}, \quad \nu = 1, 2, \ldots, 7,
\]

for sufficiently large $i$, will be such that

\[
\sigma(\nu) \approx \sigma(1) 1.1^\nu, \quad \nu = 1, 2, \ldots, 7,
\]

with $1.1^\sigma(1) < i < 1.18 \sigma(1) \approx 2.14 \sigma(1)$.

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

\[
\begin{aligned}
\text{compute } H_i \text{ from } H_{i-1} \\
\text{if } \exists \nu: H_{\sigma(\nu)} = H_i \text{ then } [j := \sigma(\nu), k := i \text{ stop}] \\
\text{if } 1.1^\sigma(1) < i + 1 \text{ then } \\
\text{store } H_i \text{ instead of } H_{\sigma(1)} \\
\text{if } \sigma(\nu) := \begin{cases} 
\sigma(\nu + 1) & \text{for } \nu \neq 7 \\
i & \text{for } \nu = 7
\end{cases}
\end{aligned}
\]
Let \( \lambda \) be the period and \( \mu \) the length of the nonperiodic segment of the sequence \( H_i \), e.g.

\[
H_\mu = H_{\mu+\lambda}, \quad H_i \neq H_j \text{ for } i < l < \mu + \lambda.
\]

**Fact 7.** The procedure search finds some \( j < k \) with \( H_k = H_j \) within \( \leq 1.1m + \lambda \) recursion steps, \( m := \max(\lambda, \mu) \).

**Proof.** Since \( \sigma \) increases by the factor 1.1, \( \sigma \) will take some value \( \sigma(\nu) \) with \( m \leq \sigma(\nu) \leq 1.1m \). Hence the for-loop stops, at the latest, with

\[
k = \sigma(\nu) + \lambda \leq 1.1m + \lambda, \quad j = \sigma(\nu),
\]

and finds the equality \( H_k = H_j \). \( \square \)

Under the assumption that each of the \( \text{ord}(H)^{\text{ord}(H)} \) functions \( f : \langle H \rangle \to \langle H \rangle \) has probability \( \text{ord}(H)^{-\text{ord}(H)} \), the stochastic behavior of \( \mu, \lambda \) have been well analyzed (see Knuth [7, Exercise 3.1.12]).

The expected values of \( \mu \) and \( \lambda \) are

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

\[
E(\mu + \lambda) = 1.25\sqrt{\text{ord}(H)} - 1/3,
\]

\[
\text{Prob}\left[ \mu + \lambda \leq \sqrt{\frac{\pi}{2} \text{ord}(H)} \right] \approx e^{-\pi/4} \approx 0.46.
\]

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

\[
1.1 \left( E(\mu) + E(\lambda) \right) \approx 1.32\sqrt{\text{ord}(H)},
\]

provided that \( f : \langle H \rangle \to \langle H \rangle \) is sufficiently random.

If in Stage 2 we compute the \( H_i \) for \( i \leq 1.32 p_i \), then most likely some relation \( H_j = H_k, j < k \), will be found, provided \( \text{ord}(H) \leq p_i^2 \). It remains to analyze the chance that \( \text{ord}(H) \leq p_i^2 \). For each prime \( p, p_i < p < p_i^2 \), we assume that the frequency of class numbers \( h(-m), m \leq n \), which are divisible by \( p \) is \( \geq p^{-1} \), 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):

\[
\sum_{p \leq n^{1/(1+e)}} p^{-1}(r - r/s)^{-(r - r/s)}.
\]

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

\[
\sum_{n^{r-1/(1+\epsilon)} < p < n^{1/r}} p^{-1}(r - 2/(1 + \epsilon))^{-(r - 2A(1 + \epsilon))}
\]

(assuming Theorem 4.27 in Hardy and Wright [6] there follows)

\[
\geq 0.83 \ln(1 + \epsilon)(r - 2/(1 + \epsilon))^{-(r - 2A(1 + \epsilon))}.
\]
Conclusion. Assume (2.1), (2.2) and that for every discriminant \( m \leq n \), \( H_0 \) in \( G(-m) \) is chosen at random. Then \( \forall \varepsilon > 0: \exists c_\varepsilon > 0: \forall n \geq n_0 \) and all \( p_i = n^{1/2r} \), \( r \leq \sqrt{\ln n / \ln \ln n} \): Stages 1 and 2 with \( O(p_i) \) compositions, factor at least \( c_\varepsilon (r - 2 + \varepsilon)(r - 2 + \varepsilon) n \) discriminants \( \leq n \).

If one assumes that very large class numbers \( h(-m) \) 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 \( r^{-r} \) and is at most \( r e^2 \) times the success frequency of Stage 1. Note that \( (r - 2)^{(r - 2)/r - r} \) approaches \( e^2 \) for large \( r \).

Remark. There is a well-known deterministic method for doing Stage 2 within \( \sqrt{2} p \), compositions and with \( O(p_j) \) 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 \( \text{ord}(H) \) will most likely have no prime divisor \( \leq p_i \).

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 \( \Delta \) is fundamental if

\[-\exists w \in \mathbb{N}, w \neq 1: \Delta / w^2 \text{ is a discriminant.}\]

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

\[ h(-m) = \frac{\sqrt{m}}{\pi} \prod_{\substack{p \text{ prime} \mid p \mid w}} \left( 1 - \frac{1}{p} \left( \frac{-m}{p} \right) \right)^{-1} \text{ for } m < -4, \]

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

\[ h(-mw^2) / h(-m) = w \prod_{\substack{p \mid w \mid p \mid w}} \left( 1 - \frac{1}{p} \left( \frac{-m}{p} \right) \right) = \prod_{\substack{p \mid w \mid p \mid w}} \left( p - \left( \frac{-m}{p} \right) \right). \]

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, \( p_i = n^{1/2r} \) (the appropriate choice of \( t, r \) will be determined by the subsequent analysis).

1. \( s := 0 \)
2. take the next \( s \) with \( \gcd(n, s) = 1, -ns \equiv 0, 1 \text{ mod } 4 \) and \(-\exists w \in \mathbb{N}: w^2 \mid s, w \neq 1, -ns / w^2 \equiv 0, 1 \text{ mod } 4 \)
3. run Stage 1 on \( ns \), which takes \( O(p_i) \) 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

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use
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 $2^{m_i}$, $m_i > 0$: $S_2(-ns) \cong \otimes_{i=1}^{\lambda} \mathbb{Z}/2^{m_i} \mathbb{Z}$. Let $\lambda$ be the number of cyclic components, then $S_2(-ns)$ has $2^\lambda$ ambiguous classes. Let $d$ be the number of odd prime factors of $ns$. Then by Theorem III, Appendix I, $\lambda$ 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 $\lambda'$ of cyclic components of $S'_2(-ns)$ is $\leq \lambda - d_n \leq d_s + 1$.

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 $\{A_1, \ldots, A_{\lambda}\} \subset S_2(-ns)$, $\lambda' \leq \lambda'$ such that

\begin{equation}
|\langle A_1, \ldots, A_{\lambda'} \rangle| = \prod_{i=1}^{\lambda'} \text{ord}(A_i).
\end{equation}

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

(4.1) Algorithm for $S_2(-ns)$.

0. $\lambda := 0, A_1 := 1$ (= the unit class)
1. generate another $A \in S_2(-ns), A \neq 1$
2. compute $A, A^2, \ldots, A^{2^{m_i}} \neq 1, A^{2^\lambda} = 1$ and put $B := A^{2^{m_i}}$
3. if $B$ yields a factorization of $n$ then stop
4. test whether $B \in \langle B_1, \ldots, B_{\lambda'} \rangle$; if $B \notin \langle B_1, \ldots, B_{\lambda'} \rangle$ then go to 5 else compute $J \subset \{1, \ldots, \lambda\}$ with $B = \prod_{j \in J} B_j$ and go to 6
5. $A_{\lambda+1} := A, B_{\lambda+1} := B, \lambda := \lambda + 1$ 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 $A^{2^{m_i}} = \prod_{j \in J} A_j^{2^{m_i}}$, $m \leq m_j$, for $j \in J$). Put $A := A \cdot \prod_{j \in J} A_j^{2^{m_i}}$; 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 $A^{2^{m_i}} = 1$ 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
   \[ h(-ns) \prod_{i=1}^{t} p_i^{e_i}, \quad e_i = \max \{v: p_i^v \leq p_i^2\}. \]
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 \( n \):

\[
\exists c, n_0 > 0 : \forall m : \forall n \geq n_0 : \forall p_i = (nm)^{1/2r} \text{ with } r \leq \sqrt{\ln n / \ln \ln n} : \\
\# \left\{ ns : s \leq m \wedge h(-ns) \left| \prod_{i=1}^{t} p_i^{e_i} \right| / (0.5 m) \right\} \geq cr^{-r}.
\]

Our experimental data in fact confirm the lower bound \( r^{-r} \). The assumption (4.3) implies

\[
\forall n \geq n_0 : \forall p_i = (n3r'/c)^{1/2r}, r \leq \sqrt{\ln n / \ln \ln n} \\
\exists s \leq 3r'/c : h(-ns) \left| \prod_{i=1}^{t} p_i^{e_i} \right|.
\]

Since Stage 1 takes \( O(p_i) \) compositions, we have

\[
T(n) = O\left( p_i r'^{\left( \ln n \right)^2} \right) = O\left( n^{1/2r} r'^{1/2} \left( \ln n \right)^2 \right).
\]

Here \( O(\ln n)^2 \) takes into account the costs for the arithmetic. We choose \( r = \sqrt{n / \ln \ln n} \), \( p_i = (n3r'/c)^{1/2r} = O(n^{1/2r} \sqrt{r}) \). Then all together (4.3) implies

\[
T(n) = o(\exp \sqrt{n \ln n \ln \ln n} ).
\]

2. \( \overline{T}(n) \). 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 \( \geq 1 - 2^{-k} \) 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 \( O(p_i) \) 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 \( B \in \langle B_1, \ldots, B_X \rangle \) (step 4). Since \( \bar{x} \leq \bar{y} \leq d_s + 1 \) this can be done in a crude way by comparing \( B \) with each of the \( 2^{\bar{x}} \leq 2^{d_s+1} = O(s) \) elements of \( \langle B_1, \ldots, B_X \rangle \). This takes \( O(s) \) steps and is sufficient for our purposes. We obtain

\[
\overline{T}(n) = O\left( d_s (p_i + s)(\ln n)^2 \right) = O\left( \log s (n^{1/2r} \sqrt{r} + s)(\ln n)^2 \right)
\]

with \( s \leq r', r \leq \sqrt{\ln n / \ln \ln n} \). Here again \( O(\ln n)^2 \) bounds the cost for the arithmetic. It follows immediately that \( \overline{T}(n) = o(T(n)) \).

Conclusion. If (4.3) holds, then the Main Algorithm, using only Stage 1, takes \( o(\exp \sqrt{n \ln n \ln n} ) \) 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 \( r^2 = \ln n / \ln \ln n \).

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 \( \leq 2^{70} \). This means that the program can factor integers \( \leq 2^{130} \approx 10^{39} \) using multipliers \( \leq 2^{10} \). Stage 1 uses the exponents \( e'_i := \max \{ v : p_i^v \leq p_i \} \). Hence the number of compositions per multiplier for Stage 1 is about \( 2.2 p_i \). Both methods for Stage 2 have been tested and they are comparable in efficiency.
A MONTE CARLO FACTORING ALGORITHM WITH LINEAR STORAGE

### Table 0

<table>
<thead>
<tr>
<th></th>
<th>$n \approx 10^{30}$</th>
<th>$n \approx 10^{34}$</th>
<th>$n \approx 10^{38}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>4093</td>
<td>8191</td>
<td>16381</td>
</tr>
<tr>
<td>$t = #$ of primes</td>
<td>563</td>
<td>1027</td>
<td>1899</td>
</tr>
<tr>
<td>$#$ compositions per multiplier in Stage 1</td>
<td>$10^4$</td>
<td>$2 \cdot 10^4$</td>
<td>$4 \cdot 10^4$</td>
</tr>
<tr>
<td>in Stage 2</td>
<td>$3 \cdot 10^3$</td>
<td>$3.7 \cdot 10^3$</td>
<td>$3 \cdot 10^4$</td>
</tr>
<tr>
<td>average number of multipliers</td>
<td>14.4</td>
<td>18</td>
<td>31</td>
</tr>
<tr>
<td>median of the numbers of multipliers</td>
<td>8</td>
<td>9</td>
<td>23</td>
</tr>
<tr>
<td>$#$ of integers factored</td>
<td>50</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>seconds per composition</td>
<td>$1.4 \cdot 10^{-3}$</td>
<td>$1.5 \cdot 10^{-3}$</td>
<td>$1.57 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>average time for factoring</td>
<td>4.4 min</td>
<td>10.7 min</td>
<td>57 min</td>
</tr>
<tr>
<td>median of the factoring time</td>
<td>2.4 min</td>
<td>5.3 min</td>
<td>42 min</td>
</tr>
</tbody>
</table>

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 $p_1 = 16381$ 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 $p_i \approx 10^6$, 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 I 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 $a, b, c \in \mathbb{Z}$ 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 $(a, b, c)$, $(\bar{a}, \bar{b}, \bar{c})$ are equivalent if there exists a linear transformation with integer coefficients and determinant 1 transforming the one form into the other, i.e.

$$
T = \begin{pmatrix}
\bar{a} & \bar{b}/2 \\
\bar{b}/2 & \bar{c}
\end{pmatrix} = \begin{pmatrix}
a & b/2 \\
b/2 & c
\end{pmatrix}^{-1}
$$

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use
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 $[(a, b, c)], [(\bar{a}, \bar{b}, \bar{c})]$ 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 $\alpha, \beta, \gamma \in \mathbb{Z}$ be such that $aa + \beta\bar{a} + \gamma(b + \bar{b}) = d$.

$$A = \frac{a\bar{a}}{d^2}$$

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

$$C = \frac{(-\Delta + B^2)/(4A)}{2}.$$

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

**Theorem I.** The equivalence classes of primitive quadratic forms with discriminant $\Delta$ 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

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

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

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. $T\begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix}T' = \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix}$
   for some integer matrix $T$ with $\det T = -1$,
4. there is a form $(a, b, c)$ in $H$ with $a|b$.

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

**Theorem II (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 $v \in \mathbb{Z}$: $-|a| < b + 2va \leq |a|$
2. $b := b + 2va, c := (b^2 - \Delta)/(4a)$
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

(i) \( \Delta = -4ac \),  
(ii) \( \Delta = b(b - 4c) \),  
(iii) \( \Delta = (b - 2a)(b + 2a) \).

In order to describe this correspondence more precisely, let

\[ F(n) = \{ (n_1, n_2) \in \mathbb{N}^2 : n = n_1n_2, \gcd(n_1, n_2) = 1 \} \]

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

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

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 \( \Delta = -2^en, n \) odd, is obtained from the \( (n_1, n_2) \in F(n) \) with \( n_1 \leq n_2 \) as follows:

\[ \#A(\Delta) = \frac{1}{2} \#F(n) = 2^{d-1} \]

if \( 3n_1 < n_2 \)

\[ \frac{n_1}{2}, \frac{n_2}{2} \]

if \( 3n_1 \geq n_2 \)

\[ \frac{n_1 + n_2}{2}, \frac{(n_2 - n_1)/2, (n_1 + n_2)/2} \]

case \( e = 2, n \equiv 1 \mod 4, n \neq 1 \):

\[ (n_1, n_2) \]

\[ (2n_1, 2n_1, (n_1 + n_2)/2) \]

\[ (n_2 + n_1, (n_1 + n_2)/2) \]

case \( e = 2, n \equiv -1 \mod 4 \):

\[ (n_1, n_2) \]

\[ (\min(n, 2e^{-2}, n_2), 0, \max(n, 2e^{-2}, n_2)) \]

\[ (\min(2e^{-2}, n_1), 0, \max(2e^{-2}, n_1)) \]

\[ (\min(2e^{-2}, n_2), 0, \max(2e^{-2}, n_2)) \]

\[-(n_1, n_2) \]

\[ (4, 4, 1 + 2e^{-4}n_1n_2) \]

\[ (2e^{-4}n_2 + n_1, 2e^{-3}n_2 - 2n_1, 2e^{-4}n_2 + n_1) \]

\[ (2e^{-4}n_1 + n_2, 2e^{-3}n_1 - 2n_2, 2e^{-4}n_1 + n_2) \]

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 \( \Delta \equiv 0 \mod 4 \) (\( \Delta \equiv 1 \mod 4 \), 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): $(a, b, c), (\bar{a}, \bar{b}, \bar{c}) \rightarrow (A, B, C)$ (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

$$\bar{e}_i(m) := \max \{ v: p_i^v | h(-m) \}.$$ 

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

- **column 1**: $h(-m)$ is free of primes $> p_i$, i.e. $h(-m) = \prod_{i=1}^{t} p_i^{\bar{e}_i(m)}$,

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

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

- **column 4**: $h(-m) | \prod_{i=1}^{t} p_i^{\bar{e}_i(m)} \cdot q$ for some $q \leq p_i^{2}$, $q$ prime,

- **column 5**: $h(-m) | \prod_{i=2}^{t} p_i^{e_i(m)} q \cdot 2^{\bar{e}_i(m)}$ for some $q \leq p_i^{2}$, $q$ prime,

- **column 6**: $h(-m) | \prod_{i=2}^{t} p_i^{e_i(m)} q \cdot 2^{e_i(m)}$ for some $q \leq p_i^{2}$, $q$ prime.

Moreover we note in

- **column 7**: $r = \ln n/(2 \ln p_i)$

- **column 8**: $10^2 \cdot r^{-r}$.

Observe that the entries in columns 1–3 of Table 1 are always greater than $10^2 r^{-r}$, which confirms Corollary 4. For $r \leq \sqrt{\ln n/\ln \ln n}$ (i.e. $r \geq 2.58, p_i \geq 53$) 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

<table>
<thead>
<tr>
<th>$p_i$</th>
<th>col 1</th>
<th>col 2</th>
<th>col 3</th>
<th>col 4</th>
<th>col 5</th>
<th>col 6</th>
<th>col 7</th>
<th>col 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.18</td>
<td>0.18</td>
<td>0.44</td>
<td>0.44</td>
<td>0.44</td>
<td>14.408</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.08</td>
<td>0.17</td>
<td>0.44</td>
<td>0.44</td>
<td>0.44</td>
<td>14.408</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4.34</td>
<td>4.34</td>
<td>4.34</td>
<td>4.34</td>
<td>4.34</td>
<td>4.34</td>
<td>4.34</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>14.70</td>
<td>14.70</td>
<td>14.70</td>
<td>14.70</td>
<td>14.70</td>
<td>14.70</td>
<td>14.70</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>22.47</td>
<td>22.47</td>
<td>22.47</td>
<td>22.47</td>
<td>22.47</td>
<td>22.47</td>
<td>22.47</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>29.28</td>
<td>29.28</td>
<td>29.28</td>
<td>29.28</td>
<td>29.28</td>
<td>29.28</td>
<td>29.28</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>32.10</td>
<td>32.10</td>
<td>32.10</td>
<td>32.10</td>
<td>32.10</td>
<td>32.10</td>
<td>32.10</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>34.70</td>
<td>34.70</td>
<td>34.70</td>
<td>34.70</td>
<td>34.70</td>
<td>34.70</td>
<td>34.70</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>37.00</td>
<td>37.00</td>
<td>37.00</td>
<td>37.00</td>
<td>37.00</td>
<td>37.00</td>
<td>37.00</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>37.62</td>
<td>37.62</td>
<td>37.62</td>
<td>37.62</td>
<td>37.62</td>
<td>37.62</td>
<td>37.62</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>41.30</td>
<td>41.30</td>
<td>41.30</td>
<td>41.30</td>
<td>41.30</td>
<td>41.30</td>
<td>41.30</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>43.18</td>
<td>43.18</td>
<td>43.18</td>
<td>43.18</td>
<td>43.18</td>
<td>43.18</td>
<td>43.18</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>44.92</td>
<td>44.92</td>
<td>44.92</td>
<td>44.92</td>
<td>44.92</td>
<td>44.92</td>
<td>44.92</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>46.48</td>
<td>46.48</td>
<td>46.48</td>
<td>46.48</td>
<td>46.48</td>
<td>46.48</td>
<td>46.48</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>48.02</td>
<td>48.02</td>
<td>48.02</td>
<td>48.02</td>
<td>48.02</td>
<td>48.02</td>
<td>48.02</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>49.44</td>
<td>49.44</td>
<td>49.44</td>
<td>49.44</td>
<td>49.44</td>
<td>49.44</td>
<td>49.44</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>50.65</td>
<td>50.65</td>
<td>50.65</td>
<td>50.65</td>
<td>50.65</td>
<td>50.65</td>
<td>50.65</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>51.58</td>
<td>51.58</td>
<td>51.58</td>
<td>51.58</td>
<td>51.58</td>
<td>51.58</td>
<td>51.58</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>53.06</td>
<td>53.06</td>
<td>53.06</td>
<td>53.06</td>
<td>53.06</td>
<td>53.06</td>
<td>53.06</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>55.12</td>
<td>55.12</td>
<td>55.12</td>
<td>55.12</td>
<td>55.12</td>
<td>55.12</td>
<td>55.12</td>
<td></td>
</tr>
</tbody>
</table>

### Table 2

<table>
<thead>
<tr>
<th>$p_i$</th>
<th>col 1</th>
<th>col 2</th>
<th>col 3</th>
<th>col 4</th>
<th>col 5</th>
<th>col 6</th>
<th>col 7</th>
<th>col 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.17</td>
<td>0.17</td>
<td>0.17</td>
<td>0.17</td>
<td>0.17</td>
<td>0.17</td>
<td>0.17</td>
<td>0.17</td>
</tr>
<tr>
<td>3</td>
<td>1.26</td>
<td>1.26</td>
<td>1.26</td>
<td>1.26</td>
<td>1.26</td>
<td>1.26</td>
<td>1.26</td>
<td>1.26</td>
</tr>
<tr>
<td>5</td>
<td>1.80</td>
<td>1.80</td>
<td>1.80</td>
<td>1.80</td>
<td>1.80</td>
<td>1.80</td>
<td>1.80</td>
<td>1.80</td>
</tr>
<tr>
<td>11</td>
<td>37.28</td>
<td>37.28</td>
<td>37.28</td>
<td>37.28</td>
<td>37.28</td>
<td>37.28</td>
<td>37.28</td>
<td>37.28</td>
</tr>
<tr>
<td>13</td>
<td>42.84</td>
<td>42.84</td>
<td>42.84</td>
<td>42.84</td>
<td>42.84</td>
<td>42.84</td>
<td>42.84</td>
<td>42.84</td>
</tr>
<tr>
<td>17</td>
<td>48.46</td>
<td>48.46</td>
<td>48.46</td>
<td>48.46</td>
<td>48.46</td>
<td>48.46</td>
<td>48.46</td>
<td>48.46</td>
</tr>
<tr>
<td>19</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
<td>54.08</td>
</tr>
<tr>
<td>23</td>
<td>61.46</td>
<td>61.46</td>
<td>61.46</td>
<td>61.46</td>
<td>61.46</td>
<td>61.46</td>
<td>61.46</td>
<td>61.46</td>
</tr>
<tr>
<td>29</td>
<td>67.90</td>
<td>67.90</td>
<td>67.90</td>
<td>67.90</td>
<td>67.90</td>
<td>67.90</td>
<td>67.90</td>
<td>67.90</td>
</tr>
<tr>
<td>31</td>
<td>73.90</td>
<td>73.90</td>
<td>73.90</td>
<td>73.90</td>
<td>73.90</td>
<td>73.90</td>
<td>73.90</td>
<td>73.90</td>
</tr>
<tr>
<td>37</td>
<td>79.90</td>
<td>79.90</td>
<td>79.90</td>
<td>79.90</td>
<td>79.90</td>
<td>79.90</td>
<td>79.90</td>
<td>79.90</td>
</tr>
<tr>
<td>41</td>
<td>85.90</td>
<td>85.90</td>
<td>85.90</td>
<td>85.90</td>
<td>85.90</td>
<td>85.90</td>
<td>85.90</td>
<td>85.90</td>
</tr>
<tr>
<td>43</td>
<td>91.90</td>
<td>91.90</td>
<td>91.90</td>
<td>91.90</td>
<td>91.90</td>
<td>91.90</td>
<td>91.90</td>
<td>91.90</td>
</tr>
<tr>
<td>47</td>
<td>97.90</td>
<td>97.90</td>
<td>97.90</td>
<td>97.90</td>
<td>97.90</td>
<td>97.90</td>
<td>97.90</td>
<td>97.90</td>
</tr>
<tr>
<td>53</td>
<td>103.90</td>
<td>103.90</td>
<td>103.90</td>
<td>103.90</td>
<td>103.90</td>
<td>103.90</td>
<td>103.90</td>
<td>103.90</td>
</tr>
</tbody>
</table>
Table 3 shows the percentages of discriminants (fundamental discriminants, resp.) \(-m \in I\) such that \(\nu\) 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 \(a_{l,k,\nu}\) in the line starting with \(l\), \(k\) and column headed with \(\nu\) \((\nu = 2k, 8, \ldots, 0)\) is the number of integers \(m\) from among the first 100,000 even integers > 100 that have the property that

\[
m / \gcd(m, \text{lcm}(1, \ldots, 2^k)) \leq 2^{2k-\nu}.
\]

The last two columns record \(r = \ln 10^l / \ln 2^k\) and \(10^5 r^{-r}\).
Table 4 also confirms our assumption (2.2). Note that \( a_{t,k,1} \) is the number of integers \( m \) among the first 100,000 even integers > 10\(^t\) such that

\[
m = \prod_{i} p_i^{e_i(m)} \quad \text{with} \quad p_i^{e_i(m)} \leq 2^k
\]

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

\[
a_{t,k,1} > 10^5 r^{-r} \quad \text{for} \quad r = \ln 10^t/\ln 2^k \leq \sqrt{\ln 10^t/\ln \ln 10^t}.
\]
This suggests an even stronger assumption than (2.2):

\[
\# \left\{ m \leq n : m \left| \prod_{i=1}^t p_i^{e_i} \right| / n \geq r \right\}
\]

for all \( n, r \leq \sqrt{\ln n / \ln \ln n} \) and \( p_i \leq n^{1/r} \). Here \( e_i := \max \{ \nu : p_i^\nu \leq p_i \} \).

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

\[
h(-ns) \left| \prod_{i=1}^t p_i^{e_i}q \right. \text{ and } q \leq 2^{2k-\nu}.
\]

Hence Stages 1 and 2 will run on at most \( 10^5/a_{k,\nu} \) multiples \(-ns\). Stage 1 with the exponents \( e_i \) takes about \( 2.2p_i \) 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 \( 1.5 \cdot 2^{k-\nu/2} \) compositions. Therefore the total number of compositions of the Main Algorithm will be bounded by

\[
b_{i, k, \nu} := \frac{10^5}{a_{i, k, \nu}} \cdot 2^k (2.2 + 1.5 \cdot 2^{-\nu/2}).
\]

**Examples.** \( n \approx 10^{30} \). Choose \( k = 12, \nu = 0, a_{15,12,0} = 7388 \). We have \( b_{15,12,0} = 205132 \), and \( n \) will be factored in about \( 2 \cdot 10^5 \approx 10^{0.18} \) compositions.

\( n \approx 10^{40} \). Choose \( k = 14, \nu = 0, a_{20,14,0} = 2182 \). We have \( b_{20,14,0} = 2778221 \), and \( n \) will be factored in about \( 2.8 \cdot 10^6 \approx 10^{0.16} \) compositions.

\( n \approx 10^{50} \). Choose \( k = 17, \nu = 0, a_{25,17,0} = 1610 \). We have \( b_{25,17,0} = 3012236 \), and \( n \) will be factored in about \( 2.9 \cdot 10^7 \approx 10^{0.15} \) compositions.

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

The examples show that the number of compositions while factoring \( n \) is smaller than \( \exp \sqrt{\ln n / \ln \ln n} \cdot \exp \sqrt{\ln n / \ln \ln n} \). For instance, for \( n \approx 10^{60} \) we have \( 2.5 \cdot 10^8 \approx 0.00116 \cdot \exp \sqrt{\ln n / \ln \ln n} \). The examples indicate that our algorithm will be faster on integers \( n \geq 10^{40} \) than the Morrison-Brillhart algorithm. Wunderlich [22] reports that the Morrison-Brillhart algorithm for \( n \approx 10^{40} \) takes about \( 322 \cdot 10^{152} \approx 3.8 \cdot 10^8 \approx 10^{0.21} \) divisions of \( Q_i, Q_i = O(\sqrt{n}) \), 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 \( g : G(\Delta) \rightarrow \{1, \ldots, 16\} \)

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

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 \( a_i, a_i = c^{i+31} \mod 2^7 \) for \( i = 1, \ldots, 16 \) with \( c = 2, 3, 5 \). We have the recursion

\[
H_{i+1} := H_i H_0^{-a_{i}(H_i)}
\]
Table 5

<table>
<thead>
<tr>
<th>$10^m$</th>
<th>$10^8$</th>
<th>$10^{10}$</th>
<th>$10^{12}$</th>
<th>$10^{14}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample 1</td>
<td>29</td>
<td>84</td>
<td>252</td>
<td>698</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>47</td>
<td>99</td>
<td>281</td>
</tr>
<tr>
<td></td>
<td>71</td>
<td>176</td>
<td>535</td>
<td>1401</td>
</tr>
<tr>
<td></td>
<td>49</td>
<td>109</td>
<td>332</td>
<td>783</td>
</tr>
<tr>
<td>sample 2</td>
<td>28</td>
<td>87</td>
<td>237</td>
<td>605</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>56</td>
<td>130</td>
<td>241</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>171</td>
<td>531</td>
<td>1314</td>
</tr>
<tr>
<td></td>
<td>38</td>
<td>109</td>
<td>413</td>
<td>930</td>
</tr>
<tr>
<td>sample 3</td>
<td>28</td>
<td>88</td>
<td>219</td>
<td>606</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>49</td>
<td>117</td>
<td>266</td>
</tr>
<tr>
<td></td>
<td>57</td>
<td>184</td>
<td>469</td>
<td>1501</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>130</td>
<td>402</td>
<td>982</td>
</tr>
<tr>
<td>$a_i = 2^{i+31} \mod 270$</td>
<td>19</td>
<td>61</td>
<td>176</td>
<td>504</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>29</td>
<td>98</td>
<td>269</td>
</tr>
<tr>
<td></td>
<td>44</td>
<td>121</td>
<td>319</td>
<td>1010</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>85</td>
<td>163</td>
<td>692</td>
</tr>
<tr>
<td>$a_i = 3^{i+31} \mod 270$</td>
<td>31</td>
<td>65</td>
<td>211</td>
<td>585</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>28</td>
<td>102</td>
<td>380</td>
</tr>
<tr>
<td></td>
<td>57</td>
<td>168</td>
<td>477</td>
<td>1485</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>111</td>
<td>385</td>
<td>945</td>
</tr>
<tr>
<td>$a_i = 5^{i+31} \mod 270$</td>
<td>28</td>
<td>79</td>
<td>217</td>
<td>819</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>56</td>
<td>152</td>
<td>388</td>
</tr>
<tr>
<td></td>
<td>59</td>
<td>174</td>
<td>480</td>
<td>1733</td>
</tr>
<tr>
<td></td>
<td>44</td>
<td>113</td>
<td>414</td>
<td>1013</td>
</tr>
</tbody>
</table>

with $H_0 \in G(\Delta)$ chosen at random. We apply this recursion to the 50 largest discriminants $\Delta < -10^m$ with $\Delta = 1 \mod 4$ 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 $H_j = H_k$,
4. the median of the number of recursion steps.

The particular favorable performance of $a_i = 2^{i+31} \mod 270$ and of $a_i = 3^{i+31} \mod 70$ 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 $a_i = 2^{i+31} \mod 270$ 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 $g: G(\Delta) \to \{1, \ldots, 4\}$

$$g(H) = \lceil \lfloor b^2 \mod (2^{13} - 1) \rfloor 4/(2^{13} - 1) \rceil + 1,$$

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
\[ H_{i+1} = \begin{cases} H_i^3 & \text{if } g(H_i) \leq 2, \\ H_i H_0^{d_i(H_i)} & \text{otherwise}, \end{cases} \]

\( H_0 \in G(\Delta) \) 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 \( \Delta = 1 \mod 4 \). 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**

<table>
<thead>
<tr>
<th>base ( d )</th>
<th>( 10^8 )</th>
<th>( 10^{10} )</th>
<th>( 10^{12} )</th>
<th>( 10^{14} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>21</td>
<td>63</td>
<td>178</td>
<td>684</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>42</td>
<td>112</td>
<td>289</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>158</td>
<td>395</td>
<td>1513</td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>92</td>
<td>322</td>
<td>1099</td>
</tr>
<tr>
<td>3</td>
<td>28</td>
<td>116</td>
<td>231</td>
<td>783</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>53</td>
<td>86</td>
<td>555</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>219</td>
<td>503</td>
<td>1519</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>157</td>
<td>260</td>
<td>1118</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>76</td>
<td>185</td>
<td>780</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>41</td>
<td>83</td>
<td>277</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>165</td>
<td>360</td>
<td>1260</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>123</td>
<td>285</td>
<td>688</td>
</tr>
<tr>
<td>5</td>
<td>34</td>
<td>103</td>
<td>273</td>
<td>746</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>58</td>
<td>143</td>
<td>429</td>
</tr>
<tr>
<td></td>
<td>74</td>
<td>224</td>
<td>570</td>
<td>1613</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>161</td>
<td>357</td>
<td>1013</td>
</tr>
<tr>
<td>6</td>
<td>23</td>
<td>43</td>
<td>98</td>
<td>435</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>24</td>
<td>79</td>
<td>196</td>
</tr>
<tr>
<td></td>
<td>55</td>
<td>114</td>
<td>299</td>
<td>927</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>72</td>
<td>188</td>
<td>684</td>
</tr>
<tr>
<td>7</td>
<td>28</td>
<td>90</td>
<td>276</td>
<td>713</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>74</td>
<td>154</td>
<td>376</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>224</td>
<td>516</td>
<td>1533</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>159</td>
<td>386</td>
<td>925</td>
</tr>
<tr>
<td>8</td>
<td>26</td>
<td>76</td>
<td>209</td>
<td>763</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>47</td>
<td>129</td>
<td>445</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>144</td>
<td>391</td>
<td>1449</td>
</tr>
<tr>
<td></td>
<td>41</td>
<td>94</td>
<td>228</td>
<td>918</td>
</tr>
</tbody>
</table>

**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
Universität Frankfurt
6 Frankfurt am Main, West Germany

Mathematisch Instituut
Universiteit Amsterdam
1018 WB Amsterdam, The Netherlands

22. S. S. WAGSTAFF & M. C. WUNDERLICH, A Comparison of Two Factorization Methods, Unpublished manuscript.