FACTORING $N = pq^2$ WITH THE ELLIPTIC CURVE METHOD

PETER EBINGER AND EDLYN TESKE

ABSTRACT. Various cryptosystems have been proposed whose security relies on the difficulty of factoring integers of the special form $N = pq^2$. To factor integers of that form, Peralta and Okamoto introduced a variation of Lenstra's Elliptic Curve Method (ECM) of factorization, which is based on the fact that the Jacobi symbols $\left(\frac{a}{N}\right)$ and $\left(\frac{a}{p}\right)$ agree for all integers a coprime with q. We report on an implementation and extensive experiments with that variation, which have been conducted in order to determine the speed-up compared with ECM for numbers of general form.

1. INTRODUCTION

In the past years, several cryptographic systems whose security relies on the difficulty of factoring numbers of the form $N = pq^2$ have been proposed. For example, Takagi [Tak98] introduced an RSA-type cryptosystem modulo $p^k q$, where $k \geq 2$. Hühnlein et al. (see [Hue00] and the references given there) consider cryptosystems based on the difficulty of computing discrete logarithms in class groups of non-maximal quadratic orders of discriminant $\Delta_q = \Delta q^2$, where the trapdoor information is the conductor q and hence hidden in the factors of Δ_q. The EPOC [FKM+00] cryptosystems work with a secret key g_p in $(\mathbb{Z}/(p^2q)\mathbb{Z})^*$ whose order modulo p^2 is p. In all these examples, both prime factors are of approximately the same size.

Cryptosystems based on the problem of factoring pq^2 do not necessarily have to work with a larger modulus than the RSA cryptosystem (see [MvOV96]) whose security relies on the difficulty of factoring numbers of the form pq. This is because the most powerful algorithm to factor pq is the number field sieve whose complexity depends on the size of the number to be factored and with which nowadays it is possible to factor 512-bit numbers [CDL+00]. On the other hand, the complexity of ECM is a function of the size of the least factor, and factors up to 180 bits [Zim00] have been found. It is estimated [Len01] that factoring a 1024-bit RSA modulus $N = pq$ is computationally equivalent to finding a 341-bit factor of a 3-prime modulus N of the same size.

In 1996, Okamoto and Peralta [PO96] introduced a variant of ECM [Len87] that was said to speed up the factorization of numbers of the form pq^2 for p sufficiently smaller than q. This algorithm used a concept called Jacobi signatures, and was later improved through the introduction of (pseudo-)random walks based on Jacobi symbols (see [Per01]). Both variants are based on the fact that if $N = pq^2$, then for all $a \not\equiv 0 \pmod{q}$ we have $\left(\frac{a}{N}\right) = \left(\frac{a}{p}\right)$. (If

Date: January 15, 2002.
1991 *Mathematics Subject Classification.* 11Y05, 11Y16, 94A60.
Key words and phrases. Integer factorization problem, elliptic curve method, random walks.

$a \equiv 0 \pmod{q}$ and $\left(\frac{a}{N}\right) \neq \left(\frac{a}{p}\right)$, we have $\left(\frac{a}{N}\right) = 0$ and $\gcd(a, N) \in \{q, q^2\}$, and we can factor N.)

Based on theoretical estimates, Peralta [Per01] states that the improved algorithm using Jacobi symbols speeds up ECM by a factor in the order of $\log p$. However, this is compared with the one-stage variant of ECM (referred to as Standard ECM in this paper) rather than the improved versions of ECM consisting of two stages (called continuations). To estimate the impact of the Peralta-Okamoto algorithm on the factoring problem the question is of how much can the use of Jacobi symbols speed up ECM to factor $N = pq^2$ compared with the fastest available variant of ECM to factor numbers of general form?

Using the implementation of ECM in the computer algebra system LiDIA [LiD00], we implemented the Peralta-Okamoto algorithm, to which we refer as the Jacobi Symbol Continuation, and did extensive experiments to optimize its parameters. We determined average running times for factoring pq^2 with $p \approx q$ and p, q between 10^9 and 10^{14} (30 to 47 bits) with this implementation and compared them with timings using the LiDIA implementation of the so-called Improved Standard Continuation of ECM, which is designed to find factors of integers of general form. We found that the Improved Standard Continuation was, on average, about twice as fast as our implementation of the Jacobi Symbol Continuation. Although these findings seem to suggest that factoring a number of the form $N = pq^2$ ($p \approx q$) with ECM is no easier than finding a factor of approximately the same size of a number of general form, further experiments should be conducted for considerably larger factors where the Fast Fourier Transform Continuation is best.

Our paper is organized as follows. In Section 2 we outline the Elliptic Curve Method and some of its continuations for factoring integers of general form. In Section 3 we present the Jacobi Symbol Continuation of ECM for numbers of the form pq^2. This includes a discussion of how to optimally implement and work with the random walks that are needed in this continuation. Section 4 gives theoretical estimates for the running times of the Improved Standard Continuation and the Jacobi Symbol Continuation. In Section 5 we report on various experiments to optimize the choice of parameters in the Jacobi Symbol Continuation. Section 6 contains a performance comparison with the Improved Standard Continuation. We conclude in Section 7.

2. The Elliptic Curve Method (ECM)

In brief, the idea of ECM is the following: one uses the addition formulae for the group of points of an elliptic curve over a finite field to perform arithmetic operations in the set of points of an elliptic curve defined over $(\mathbb{Z}/N\mathbb{Z})$, ignoring the fact that $(\mathbb{Z}/N\mathbb{Z})$ is not a field for N composite. This either works well, or inversion modulo N fails and thus a factor of N is revealed. Crucial for the performance of ECM is to make sure that the latter happens with a sufficiently high probability.

In our following description of ECM and its variants we restrict ourselves to the essentials. We refer to [CP01] for a survey on various tricks to speed up the algorithms.

2.1. **Elliptic Curves.** Let us first briefly review elliptic curves modulo p (p prime) and modulo N (N composite).

If p is prime, $p \neq 2, 3$ and $\mathbb{F}_p$ is the finite field with p elements, the *elliptic curve* E over $\mathbb{F}_p$ is given by the equation

$$E/\mathbb{F}_p : y^2 = x^3 + ax + b \ ,$$

where a,$b \in \mathbb{F}_p$ and such that $4a^3 + 27b^2 \neq 0$ (in $\mathbb{F}_p$). The set of points (x, y) satisfying this equation, together with the point at infinity that is denoted by $\mathcal{O}$ and serves as neutral element, forms a finite abelian group, $E(\mathbb{F}_p)$, which is written additively. The inverse of the point $P = (x_P, y_P)$ is $-P = (x_P, -y_P)$. To add the points $P = P(x_P, y_P)$ and $Q = (x_Q, y_Q)$, both $P, Q \neq \mathcal{O}$ and $P \neq -Q$ we do the following: Let

$$\lambda = \frac{y_P - y_Q}{x_P - x_Q} \qquad \text{if } x_P \neq x_Q \ , \tag{2.1}$$

$$\lambda = \frac{3x_P^2 + a}{2y_P} \qquad \text{if } x_P = x_Q \ . \tag{2.2}$$

Then $R = P + Q$, where $R = (x_R, y_R)$ with

$$x_R = \lambda^2 - x_P - y_P \quad \text{and} \quad y_R = -y_P - \lambda(x_R - x_P) \ . \tag{2.3}$$

Now let N be an odd integer not divisible by three, i.e., $N = \prod_{i=1}^{r} p_i$ with $p_i > 3$ for all prime factors p_i. Let $a, b \in \mathbb{Z}/N\mathbb{Z}$ with $4a^3 + 27b^2 \in (\mathbb{Z}/N\mathbb{Z})^*$. In the projective plane, we define the elliptic curve

$$E(\mathbb{Z}/N\mathbb{Z}) := \{(x : y : z) \in \mathbb{P}^2(\mathbb{Z}/N\mathbb{Z}) : \\ y^2 z = x^3 + axz + bz^3\} \ .$$

If the p_i are known and pairwise distinct, the addition law in $E(\mathbb{Z}/N\mathbb{Z})$ can be performed by applying the addition law in each group $E(\mathbb{Z}/p_i\mathbb{Z})$ and then using Chinese Remaindering. This works since $E(\mathbb{Z}/N\mathbb{Z}) \cong E(\mathbb{Z}/p_1\mathbb{Z}) \times \cdots \times E(\mathbb{Z}/p_r\mathbb{Z})$. If the p_i are not known, for addition in $E(\mathbb{Z}/N\mathbb{Z})$ we simply use the same formulae as for addition in $E(\mathbb{Z}/p\mathbb{Z}) = E(\mathbb{F}_p)$ with p prime. Then we also can switch to affine coordinates, and can use the formulae given above. However, the arithmetic in (2.1), (2.2), (2.3) is modulo N, rather than modulo the prime p. Thus, to compute λ we have to compute the inverses modulo N of $x_P - x_Q$ and $2y_P$, respectively. If $x_P - x_Q$, or $2y_P$, is not coprime to N, the Euclidean algorithm to compute the respective inverse fails, and this reveals a non-trivial factor of N. It is easy to see that this happens if $U + V = \mathcal{O}_{p_i}$ for some prime factor p_i of N but $U + V \neq \mathcal{O}_N$. (Here $\mathcal{O}_m$ stands for the point at infinity of $E(\mathbb{Z}/m\mathbb{Z})$.) This motivates the strategy of ECM, which is: compute multiples of kP of a point P on the curve modulo N and hope that $kP = \mathcal{O}_{p_i}$ but $kP \neq \mathcal{O}_N$.

2.2. **Standard Version of ECM.** We have the following algorithm:

Algorithm: Standard ECM.
Input: N.
Output: A non-trivial factor of N, or "no success".

1. Choose a random curve $E = E_{a,b}$ and a random point $P = (x, y)$ on E:
$$a, x, y \in_R [0, \ldots, N-1] \text{ and } b = y^2 - x^3 - ax ,$$
and check that $\gcd(4a^3 + 27b^2, N) = 1$. [Here, and throughout the paper, $\in_R$ indicates chosen at random with respect to the uniform distribution.]
2. Choose *smoothness bounds* B and C. ($B, C \in \mathbb{N}$, to be specified below.)
3. For all primes $q < B$ do
$$\begin{aligned} e_q &= \max\{e : q^e \leq C < q^{e+1}\} . \\ P &\leftarrow q^{e_q} \cdot P . \end{aligned}$$
If the latter computation fails because some inverse modulo N cannot be computed, a factor of N is found. Return that factor.
4. If you want to continue, go back to Step 1. Otherwise, output "no success".

Upon completion of Step 3, Standard ECM has computed the multiple $k_{B,C} \cdot P$ of the point P, where
$$k_{B,C} = \prod_{2 \leq q < B} q^{e_q} , \qquad q^{e_q} < C .$$
Now, observe that $k_{B,C}P = \mathcal{O}_p$ if and only if all prime factors of $\operatorname{ord} P$ modulo p are less than B, and all prime powers dividing $\operatorname{ord} P$ are less than C. Here, $\operatorname{ord} P$ modulo p stands for the element order of the point P in $E(\mathbb{Z}/p\mathbb{Z})$.

Remark 2.1. In fact, the scalar multiple in Step 3 is calculated using the Montgomery-Chudnowsky representation of an elliptic curve (see [Mon92]). This representation uses projective coordinates and has the arithmetic advantage that no inversions are needed for doubling and addition of points. A nontrivial factor of N then won't be detected by a failure of an inversion in Step 3. Instead, at the end of Step 3, we compute $\gcd(N, z)$, where z is the z-coordinate of $k_{B,C}P$ in Montgomery-Chudnowsky representation, which is zero modulo p if $k_{B,C}P = \mathcal{O}_p$.

2.3. **Smoothness and semi-smoothness.**

Definition 2.1. Let B be a positive integer. An integer n is called *smooth with respect to* B if all its prime factors are less than B.

For a point P on the elliptic curve $E(\mathbb{Z}/N\mathbb{Z})$ and for a prime factor p of N the Hasse bound on $\#E(\mathbb{Z}/p\mathbb{Z})$ implies that $\operatorname{ord} P \pmod p \leq p + 1 + 2\sqrt{p}$. Thus, with $C = p + 1 + 2\sqrt{p}$, Standard ECM succeeds for any elliptic curve whose order modulo p is B-smooth. In practice we put $B = C$, and accordingly, we let $k_B = k_{B,C}$.

Remark 2.2. The smoothness properties of *randomly chosen integers* are well studied and can be expressed in terms of Dickman's rho function. See, for example, [CP01]. Lenstra [Len87] shows that the smoothness properties of group orders of elliptic curves modulo p in the interval $]p+1-\sqrt{p}, p+1+\sqrt{p}[$ resemble those of randomly chosen integers in that interval.

It is obvious that the larger B is, the larger is the probability that an elliptic curve group order is B-smooth. On the other hand, the larger B is, the more expensive is the cost of Step 3 of Standard ECM. An efficient way to increase the size of the largest possible prime factor of $\#E(\mathbb{Z}/p\mathbb{Z})$ so that ECM still works is the *large-prime variant*. It is based on the fact that for a random integer it is very likely that exactly one prime factor is much bigger than all the other factors.

Definition 2.2. Let B, D be positive integers. An integer n is *semi-smooth with respect to B and D* if $p < B$ for all prime factors p of n with the possible exception of one prime $q \mid n$, for which $q < D$ and $q^2 \nmid n$.

For $u > v > 1$ let $\sigma(u,v)$ denote the asymptotic probability that a random integer modulo p is semi-smooth with respect to $p^{1/u}$ and $p^{1/v}$. Precisely, let $\sigma(u,v) = G(1/u, 1/v)$, where, for $0 < \alpha < \beta < 1$, $G(\alpha,\beta) = \lim_{x\to\infty} \Psi(x, x^\beta, x^\alpha)/x$ with

$$\Psi(x,y,z) = \#\{n \le x \, : \, n = n_1 n_2 \cdots n_r, n_1 \ge n_2 \ge \cdots \ge n_r, n_i \text{ prime }, n_1 \le y, n_2 \le z\} \, .$$

Bach and Peralta [BP96] give explicit formulae how to efficiently compute $\sigma(u,v)$.

2.4. Continuations.

If Standard ECM is not successful in factoring N, we switch from Montgomery-Chudnowsky representation to the Weierstrass representation and proceed with a second stage, called *continuation*. For this, we take the point $W := k_B P$ calculated in Step 3 of Standard ECM and hope that $\operatorname{ord} W$ modulo p is a prime number less than or equal some integer D. This is the case if $\operatorname{ord} P$ is semi-smooth with respect to B and D.

The *Standard Continuation* of ECM works as follows: for all primes $r \in (B, D]$, compute rW. If $rW = \mathcal{O}_p$ for some prime factor p of N but $rW \neq \mathcal{O}_N$, a non-trivial factor of N is revealed. This can be implemented to take $O(D-B)$ group operations.

The *Improved Standard Continuation* (ISC) is: Use the baby-step giant-step method to find a prime $r \in (B, D]$ such that $rW = \mathcal{O}_p$ for some $p \mid N$. If $r \in (B, D]$ and $w = \lceil \sqrt{D} \rceil$, then there exist $0 \le i < w$ and $1 \le j \le w$ such that $r = jw - i$. Then $rW = O_p$ if and only if $jwW = iW$ in $E(\mathbb{Z}/p\mathbb{Z})$. Computing the baby steps iW and the giant steps jwW takes $O(\sqrt{D})$ group operations. Table look-ups for equality checks, as they are usually used in baby-step giant-step applications, are not possible here since p is not known. Instead, for an equality check $iW = jwW$ modulo some prime factor p of N we compute $\gcd(x_{iW} - x_{jwW}, N)$. This is necessary only for i of the form $i = jw - r$ with $r \in (B, D]$ prime, i.e., by the prime number theorem, for about $D/\ln D - B/\ln B$ values of r. By accumulating the differences $(x_{iW} - x_{jwW})$, each such gcd computation can be replaced by a multiplication followed by reduction modulo N.

The optimal choices for the semi-smoothness parameters depend on the size of the prime factor p we want to find. In Table 1 we give sample values for B $(= C)$ and D as used in the LiDIA implementation [LiD00] of the Improved Standard Continuation. We also indicate

Size of p	B	D	u	v	$\sigma(u,v)$	Size of p	B	D	u	v	$\sigma(u,v)$
10^9	349	4297	3.5	2.5	0.08	10^{12}	997	31489	4.0	2.7	0.04
10^{10}	411	8861	3.8	2.5	0.06	10^{13}	1439	54617	4.1	2.7	0.03
10^{11}	659	17981	3.9	2.6	0.05	10^{14}	2111	89501	4.2	2.8	0.02

Table 1. Semi-smoothness bounds - Selected LiDIA parameter for Improved Standard Continuation. Semi-smoothness probabilities.

the corresponding u and v such that $B = p^{1/u}$ and $D = p^{1/v}$, and the semi-smoothness probabilities $\sigma(u,v)$. The latter were computed using software that was made available to us by Peralta.

There is another continuation, based on the *Fast Fourier Transform* [Mon92]. This continuation is the faster than the Improved Standard Continuation only if we look for very big factors p of N, of 20 or more digits [Mue95]. We do not go into the details of this method since in our study, we will be working with smaller factors and thus will be using the Improved Standard Continuation as a reference.

Numerous problems in computational number theory that can be solved using baby-step giant-step techniques can also be tackled using Pollard's rho method [Pol75, Pol78]. Let us look at that option. As before, let $W = k_B P$. Let $t_N = \operatorname{ord} W$ modulo N and $t_p = \operatorname{ord} W$ modulo p. We could use Pollard's rho method with an iterating function F that via $Q_{i+1} = F(Q_i)$ defines a (pseudo-)random walk $(Q_i)_{i\in\mathbb{N}}$ in the subgroup generated by W. However, with the factorization of N unknown, in general this is possible only modulo N. In general, we *cannot* define an iterating function such that $Q_{i+1} = F(Q_i)$ modulo p. Thus the expected running time of a Pollard rho application is $O(\sqrt{t_N})$ and not $O(\sqrt{t_p})$, i.e., in terms of the number N to be factored rather than the factor p.

3. Continuation of ECM with Jacobi Symbols

Let p be an odd prime. For an integer a, consider the congruence

$$x^2 \equiv a \pmod{p} \ . \tag{3.1}$$

We define the *Legendre symbol* $\left(\frac{a}{p}\right)$ as follows: $\left(\frac{a}{p}\right) = 0$ if $p|a$; $\left(\frac{a}{p}\right) = 1$ if $p \not| a$ and (3.1) has solutions; and $\left(\frac{a}{p}\right) = -1$ if (3.1) has no solutions. This definition can be extended to composite numbers, as follows (cf. [Coh93]):

Definition 3.1. Let $N \in \mathbb{N}$. For an integer a, we define the *Jacobi* symbol $\left(\frac{a}{N}\right)$:
$\left(\frac{a}{1}\right) = 1$.
$\left(\frac{a}{2}\right) = 0$ if a is even, and $\left(\frac{a}{2}\right) = (-1)^{(a^2-1)/8}$ if a is odd.
If $N = \prod_{i=1}^{l} p_i$ with the p_i not necessarily distinct, then $\left(\frac{a}{N}\right) = \prod_{i=1}^{l}\left(\frac{a}{p_i}\right)$, where $\left(\frac{a}{p}\right)$ is the Legendre symbol.

From now on, let N be of the form $N = pq^2$. Then, for all $a \not\equiv 0 \pmod q$,

$$\left(\frac{a}{N}\right) = \left(\frac{a}{p}\right) . \tag{3.2}$$

We now explain how this property can be exploited in the *Jacobi Symbol Continuation* (JSC) of ECM.

Let $W = k_B P$ be the point computed by ECM's first stage, i.e., Step 3 of Standard ECM, and let t_p and t_N as above. We define a sequence $(Q_i)_{i\in\mathbb{N}} \subseteq E(\mathbb{Z}/N\mathbb{Z})$ by $Q_0 = W$ and $Q_{i+1} = F(Q_i)$, where

$$F(Q_i) = \begin{cases} 2Q_i \ , & \text{if } \left(\frac{x_{Q_i}}{N}\right) = 1 \ , \\ Q_i + Q_0 \ , & \text{otherwise} \ . \end{cases} \tag{3.3}$$

This sequence eventually becomes periodic. Since $Q = Q' \pmod p$ implies that $F(Q) = F(Q') \pmod p$, its period depends only on t_p, not on t_N. Experimentally, we find that on average this sequence becomes periodic after $\approx 1.9\sqrt{t_p}$ iterations. This value is based on taking averages over several 10000 runs with various values t_p ranging from 10^3 to 10^8, using the same approach as in [Tes98b].

Remark 3.1. A sequence $(Q_i)_{i\in\mathbb{N}}$ generated by a random function F becomes periodic after an expected number of $\sqrt{t_p\pi/2} \approx 1.25\sqrt{t_p}$ iterations [Har60]. We know from [Tes01] that wider iterating functions (in the sense that more partitions are involved) better simulate random functions. For example, if for $m = 1, 2, 3$ we let $m_i \in_R [1, D]$, $V_i = m_i W$ and define

$$F(Q_i) = \begin{cases} 2Q_i & \text{if } \left(\frac{x_{Q_i}}{N}\right) = 1 \text{ and } \left(\frac{x_{Q_i}+1}{N}\right) = 1 \ , \\ Q_i + V_1 & \text{if } \left(\frac{x_{Q_i}}{N}\right) = 1 \text{ and } \left(\frac{x_{Q_i}+1}{N}\right) \neq 1 \ , \\ Q_i + V_2 & \text{if } \left(\frac{x_{Q_i}}{N}\right) \neq 1 \text{ and } \left(\frac{x_{Q_i}+1}{N}\right) = 1 \ , \\ Q_i + V_3 & \text{if } \left(\frac{x_{Q_i}}{N}\right) \neq 1 \text{ and } \left(\frac{x_{Q_i}+1}{N}\right) \neq 1 \ , \end{cases}$$

then we find that on average the sequence given by $Q_0 = W$ and $Q_{i+1} = F(Q_i)$ becomes periodic after only $1.5\sqrt{t_p}$ iterations. However, in contrary to the iterating functions considered in [Tes01], the above widening of the iterating function is rather costly: we need to evaluate two Jacobi symbols in each iteration, instead of just one in (3.3). As a consequence, we find that in terms of CPU-time, the second sequence takes about 1.15 to 1.2 times longer to become periodic. We therefore work with the function in (3.3).

Once two points Q_i and Q_j have been found for which $Q_i = Q_j \bmod p$ but $Q_i \neq Q_j \bmod N$, we are likely to find a nontrivial factor of N as $\gcd(x_{Q_i} - x_{Q_j}, N)$. To find such a match (Q_i, Q_j), we use an algorithm from a family of cycle-finding algorithms due to Brent [Bre80]. This algorithm is more than 1.5 times faster[1] than Floyd's method, which is usually referred to in this context. It works as follows:

[1]We compare the performance of cycle-finding algorithms in terms of its *expected delay factors*: If E_ρ is the expected number of iterations until a match occurs, and $E_{l(\rho)}$ is the expected number of iterations until a match is found by a certain cycle-finding method, the expected delay factor is defined as $\delta = E_{l(\rho)}/E_\rho$. For a random function $F : S \to S$, we have $E_\rho = \sqrt{\pi/2}\sqrt{|S|}$; then for Floyd's method, we have $\delta_{\mathrm{Fl}} \approx 3.09/\sqrt{\pi/2} \approx 2.47$ (cf. [Pol75]), while for Brent's cycle-finding algorithm we (experimentally) find $\delta_{\mathrm{Br}} \approx 1.97/\sqrt{\pi/2} \approx 1.57$.

Brent's cycle-finding algorithm
We have an auxiliary element R and an auxiliary index r. We initialize $R = Q_0$ and $r = 2$. We compute the next term Q_i. If $Q_i = R$, a match is found and we stop. If $i \geq r$, we replace r by $2r$ and R by Q_i. Then we compute the next term of the sequence, and so forth.

That is, each term Q_i is checked for equality with the term Q_j where $j = 0$ if $i = 1, 2$, and $j =$ the largest power of 2 strictly less than i if $i > 2$. Of course, in our application the equality check is done by computing $\gcd(x_R - x_{Q_i}, N)$. As in the Improved Standard Continuation, gcd's can be accumulated: every s-th iteration we compute the gcd of the product of s consecutive differences $(x_R - x_{Q_i})$ and N, where s is optimized to obtain minimal running times.

Summing up, the Jacobi Symbol Continuation works as follows: Let $Q_0 = W$. Use the iterating function (3.3) to define the sequence $(Q_i)_{i\in\mathbb{N}}$ and use Brent's algorithm to find a match $Q_i = Q_j \bmod p$. Such a match reveals a nontrivial factor of N if $Q_i \neq Q_j \bmod N$.

3.1. **When to discard a curve?** In the Improved Standard Continuation, the maximal running time spent with each curve $E(\mathbb{Z}/N\mathbb{Z})$ is predetermined by the semi-smoothness bounds B and D. If after the application of the baby-step giant-step method no factor of N has been found, we conclude that the order of W modulo p was not semi-smooth with respect to B and D, and the curve is to be discarded. In the Jacobi Symbol Continuation, however, the probabilistic character of Pollard's rho method results in a spread of the actual running times for various runs of the algorithm. To illustrate this, let us first consider a randomly chosen iterating function $F : S \to S$ (S any finite set). Let ρ denote the number of iterations until a match $Q_i = Q_j$ occurs. Then the expected value of ρ is, of course, $\beta := \sqrt{|S|\pi/2}$ and, from [Har60], the probability that a match occurs after at most $k\beta$ iterations is given as

$$\mathcal{P}(\rho \leq k\beta) = \int_0^{k\sqrt{\pi/2}} x e^{-x^2/2}\, dx = 1 - e^{-k^2\pi/4} =: \mathcal{P}(k)\ . \tag{3.4}$$

Thus, for example, with probability $\mathcal{P}(1/2) = 1 - e^{-\pi/16} \approx 18\%$ it takes only at most half as long as expected until a match occurs, and with probability $1 - \mathcal{P}(2) = e^{-\pi} \approx 4\%$ it takes at least twice as long as expected. Now, in the Jacobi Symbol Continuation, we expect a match to be detected after approximately $1.9\sqrt{t_p} \cdot \delta_{\mathrm{Br}} \approx 3\sqrt{t_p}$ iterations. Assuming that the corresponding probability distribution is essentially the same as in the random case, we replace β by $\beta' = 3\sqrt{t_p}$ in (3.4). Then we find that in the JSC a match is indeed detected after at most $3\sqrt{t_p}$ iterations with probability only 54%. On the other hand, plugging $k = 4/3, 5/3$ and 2 into (3.4) we find that with probabilities $25\%, 11\%$ and 4% it takes at least $4\sqrt{t_p}$, $5\sqrt{t_p}$ and $6\sqrt{t_p}$ iterations, respectively, to detect a match. Thus, if after $3\sqrt{D}$ iterations no match has been found, this may be either because $\operatorname{ord} W$ is not semi-smooth with respect to B and D, *or* because we simply have bad luck in that particular execution of the rho method. A decision when a curve should be given up must be made, and we decided for our implementation to stop after $6\sqrt{D}$ iterations. It is worth noting that this

Cycle-finding algorithms with even smaller delay factors exist (see [Tes98a]), but they require more equality checks per iteration and thus lose their advantage for our application here.

decision does not influence the actual performance of the optimized algorithm, since the semi-smoothness bounds will be optimized experimentally.

3.2. **A remark on semi-smoothness probabilities.** As a consequence of the spread of the actual number of iterations for a given order t_p, the semi-smoothness probabilities $\sigma(u, v)$ do not exactly reflect the probability of success of the Jacobi Symbol Continuation for given parameters $B = p^{1/u}$ and $D = p^{1/v}$. When executing $6\sqrt{D}$ iterations, we have a 96% chance of success to factor $N = pq^2$ if $\#E(\mathbb{Z}/p\mathbb{Z})$ is semi-smooth with respect to B and D. But at the same time, we *implicitly* work with larger values of D as well. For example, we also have a 54% chance of success to find the factor p should $\#E(\mathbb{Z}/p\mathbb{Z})$ be semi-smooth with respect to B and $4D$. In fact, the probability of success of the JSC for given semi-smoothness parameters B, D is bounded below by

$$\sum_{i=1}^{r} \mathcal{P}(2\sqrt{D/D_i}) \cdot \Big(\sigma\big(u, v(D_i)\big) - \sigma\big(u, v(D_{i-1})\big)\Big) ,$$

for any choice of integers D_i with $0 \le D_0 < D_1 < \cdots < D_r < \infty$, and where $v(D_i)$ is such that $D_i = p^{1/v(D_i)}$.

4. Estimated Running Times

We now give some rough estimates for the expected running times of the Jacobi Symbol Continuation and the Improved Standard Continuation. All constants in the following notation are machine- and implementation-dependent. They also depend on the size N of the number to be factored; this dependence is polynomial in $\log N$.

- M: The cost of calculating a scalar multiple tP of a point on an elliptic curve $E(\mathbb{Z}/N\mathbb{Z})$ in Montgomery-Chudnowsky representation is expressed as $M \log t$.
- A: The cost of adding two points on $E(\mathbb{Z}/N\mathbb{Z})$ in Weierstrass representation. [For simplicity, we treat doubling a point and adding two points equally.]
- G: The cost of calculating the greatest common divisor of two numbers of size N.
- F: The cost of a multiplication of integers of size N followed by reduction modulo N.
- J: The cost of calculating a Jacobi symbol $\left(\frac{a}{N}\right)$ for an integer $a \pmod N$.

In Table 2, we list sample values for some of these costs, measured on a Sun Ultra 60 Workstation using LiDIA.

With p denoting the prime factor to be found, we work with semi-smoothness bounds $B = p^{1/u}$ and $D = p^{1/v}$, where $u > v > 1$. Note that the optimal choices of u and v are not necessarily the same for the ISC and the JSC.

We first estimate the cost of Standard ECM. By the prime number theorem, there are $\pi(B) \approx B/\ln B = up^{1/u}/(\ln p)$ primes less or equal to B. Each such prime is bounded by $B = p^{1/u}$, so that the total cost of Standard ECM is about $1.44Mp^{1/u}$. [Here we use that $\ln p/\log p \approx 1.44$.]

In the Improved Standard Continuation, we now have to add the expected cost for the baby-step giant-step stage. If no factor is found in this stage, this takes roughly $\sqrt{D}$ baby and $\sqrt{D}$ giant steps. Otherwise, we need an expected number of $\sqrt{D}$ baby steps and $0.5\sqrt{D}$

# digits of p, q	# digits of $N = pq^2$	A $[\mu s]$	G $[\mu s]$	J $[\mu s]$
9	27	153.2	19.5	115.9
10	30	169.9	22.3	128.4
11	33	193.3	25.9	143.0
12	36	210.0	29.7	159.4
13	39	228.3	33.1	172.6
14	42	255.9	37.9	185.6

TABLE 2. Sample running times for point addition on $E(\mathbb{Z}/N\mathbb{Z})$, gcd computation, Jacobi symbol computation

giant steps. This results in a total cost $2A\sqrt{D} = 2Ap^{1/(2v)}$ if successful and $3/2 \cdot A\sqrt{D} = 3/2 \cdot Ap^{1/(2v)}$ if not successful. Assuming that s differences $(x_{iW} - x_{jwW})$ are accumulated before a greatest common divisor is computed, we have to perform about $D/(s \ln D)$ gcd computations. This is at an approximate total cost $(F+G/s)D/\ln D = (F+G/s)p^{1/v}v/\ln p$. We expect to have to consider about $1/\sigma(u,v)$ elliptic curves modulo N, so that the total cost of the Improved Standard Continuation can be roughly estimated as

$$T_{\text{ISC}} = \frac{1.44Mp^{1/u} + 2Ap^{1/(2v)} + (F + G/s)vp^{1/v}/\ln p}{\sigma(u,v)} .$$

In the Jacobi Symbol Continuation, if no factor is found for a given elliptic curve, we execute $6\sqrt{D} = 6p^{1/(2v)}$ iterations of the (pseudo-)random walk, and otherwise an expected number of $3\sqrt{t_p} \leq 3\sqrt{D}$ iterations. Each iteration requires one Jacobi symbol evaluation and one addition of points. We further assume that s differences $(x_R - x_{Q_i})$ are accumulated before a greatest common divisor is computed. Assume that we have to consider about $1/\sigma(u,v)$ elliptic curves modulo N (see, however, Section 3.2). Then the total cost of the Jacobi Symbol Continuation can be roughly estimated as

$$T_{\text{JSC}} = \frac{1.44Mp^{1/u} + 6(J + A + F + G/s)p^{1/(2v)}}{\sigma(u,v)} . \tag{4.1}$$

These running times show that asymptotically, the Jacobi Symbol Continuation is faster in factoring $N = pq^2$ than the Improved Standard Continuation, assuming that D is of the same order of magnitude in both algorithms. However, for the range where the ISC is deployed it cannot be easily predicted from these values which method is better. Also, we still have to optimize the choice of B and D for the JSC.

5. Optimizing the Jacobi Symbol Continuation

Due to the doubly probabilistic nature of the Jacobi Symbol Continuation, a large number of experiments needs to be done to determine the optimal semi-smoothness parameters B and D for finding a prime factor of a certain size. We thus decided to experiment with prime factors p having between 10 and 15 decimal digits. Then performing between 10000 and 2500 runs of the Jacobi Symbol Continuation for each pair of parameters (B, D) is feasible given

B \ D	50000	55000	60000	65000	70000	75000	Deviation
2200	58%, 8.56	59%, 8.74	60%, 9.00	61%, 9.19	62%, 8.84	62%, 9.13	+0.1%
2300	56%, 8.76	57%, 8.64	59%, 8.74	59%, 8.40	60%, 8.86	61%, 9.02	−1.9%
2400	55%, 8.70	56%, 8.89	57%, 8.59	58%, 9.16	59%, 8.95	60%, 9.04	−0.2%
2500	54%, 9.03	55%, 8.83	56%, 8.82	57%, 9.11	58%, 9.08	59%, 8.80	+0.5%
2600	54%, 8.98	55%, 8.95	56%, 8.90	57%, 8.65	58%, 9.18	58%, 9.06	+0.6%
2700	52%, 8.33	54%, 8.98	54%, 8.69	56%, 9.09	56%, 9.30	57%, 9.49	+0.9%
Deviation	−2.0%	−0.7%	−1.3%	+0.3%	+1.5%	+2.1%	

TABLE 3. Jacobi Symbol Continuation: Average running times for $p \approx 10^{13}$ (50 different numbers, 50 times each)

our computational resources, and still sufficiently reliable average values should be obtained. For prime factors in that range, the Improved Standard Continuation is the best variant of ECM to factor numbers of general form.

For each order of magnitude of p, i.e., for $p \approx 10^k$, where $k = 9, 10, 11, 12, 13, 14$, we did a first round of experiments to get an idea in which ranges for B and D to look for optimized parameters. The parameter choice for this first round was also influenced by the optimized LiDIA parameters given in Table 1 and by theoretical estimates using (4.1). As a result, for $k = 9, 10, 11, 12, 13, 14$ we selected 6 values each of B and D that we combined to 36 pairs (B, D). For each pair (B, D), we conducted the following experiment:

[Algorithm to determine the average running time for JSC given B, D.]

1. Let $m = 100$ if $k = 9, 10$, and $m = 50$ if $k = 11, 12, 13, 14$.
2. For $i = 1$ to m do
3. Select random primes $p, q \in [10^k, 2 \cdot 10^k)$, and let $N = pq^2$.
4. For $j = 1$ to m do
   (a) Choose a random curve $E(\mathbb{Z}/N\mathbb{Z})$ as described in Standard ECM.
   (b) Apply Step 3 of Standard ECM with parameter $B(= C)$. If p is found, go to Step 4(d).
   (c) Apply the Jacobi Symbol Continuation, with parameter D. If p is found, go to Step 4(d). If p has not been found after $6\sqrt{D}$ iterations, go back to Step 4a.
   (d) Record the running times for Step 4(b) and Step 4(c), and the total running time.
5. Take the average of the total running time over all m^2 rounds of the algorithm.
6. Determine which percentage of the total running time was spent with the Jacobi Symbol Continuation, on average.

As an example, in Table 3 we show the results of this experiment for $k = 13$. Here, for each pair (B, D), the second value shows the average total running time (in seconds) on a Sun Ultra 60 Workstation, while the first value shows the average percentage of the total running time that was spent on the Jacobi Symbol Continuation. Given these results, it is hard to tell which choice of (B, D) is best. First, the average of the running times do not seem to

Size of p	B	D	u	v	$\sigma(u,v)$	Size of p	B	D	u	v	$\sigma(u,v)$
10^9	325	1750	3.6	2.8	0.05	10^{12}	n/av	n/av			
10^{10}	550	7500	3.6	2.6	0.06	10^{13}	2300	50000	3.9	2.8	0.04
10^{11}	950	10000	3.7	2.8	0.05	10^{14}	3400	80000	4.0	2.9	0.03

TABLE 4. Semi-smoothness bounds - Optimized parameter for Jacobi Symbol Continuation. Semi-smoothness probabilities.

be stable, which has to be attributed to the large spread of the running time, and even 2500 runs of the algorithm are not enough to balance that out. Secondly, the differences between the 36 running times are quite small. To pick the best pair (B, D) out of the 36 pairs under consideration, for each value of B, we take the average of the running times for the 6 values of D, and compute the deviation of that average from the average of all 36 running times. We pick that value of B whose deviation is the smallest, i.e. $B_{\mathrm{JSC},13} = 2300$. We do the same with D, and hence pick $D_{\mathrm{JSC},13} = 50000$.

Interestingly, the data in Table 3 show that more than half of the total running time, on average, is spent on the continuation. This a higher proportion than in the case of the Improved Standard Continuation [Mue95], and does not agree with the rule of thumb [CP01, p.307] that only a fraction of 1/4 to 1/2 should be spent on the continuation stage. However, to reduce the proportion spent on the Jacobi Symbol Continuation would require a larger value for B, or a smaller value for D. In our initial experiments, both such choices led to worse running times.

In Table 4 we summarize the optimized semi-smoothness bounds for all ranges of p under consideration. As with Table 1, we also indicate the corresponding u and v such that $B = p^{1/u}$ and $D = p^{1/v}$, and the semi-smoothness probabilities $\sigma(u, v)$. Unfortunately, data for $k = 12$ are not available; we realized too late that the corresponding process on our machine had died.

6. Comparison with the Improved Standard Continuation

We finally compare the running times for the optimized (as in Section 5) Jacobi Symbol Continuation with the Improved Standard Continuation as implemented in LiDIA. For this, we did the same experiment as in Section 5, only with the JSC replaced by the ISC, and with the optimal semi-smoothness bounds as in Table 1. The results are given in Table 5. These data show that the Improved Standard Continuation is about twice as fast as the Jacobi Symbol Continuation. This outcome is rather disappointing, and was to our surprise. After all, the semi-smoothness bound D enters the complexity of the Jacobi Symbol Continuation only in terms of $\sqrt{D}$. On the other hand, the Improved Standard Continuation has to accommodate $D/\ln D$ equality checks, which results in $D/\ln D$ multiplications modulo N when greatest common divisors are accumulated.

However, there is a crucial difference between the ISC and the JSC: Applied to factor $N = pq^2$, the Improved Standard Continuation succeeds if $\#E(\mathbb{Z}/p\mathbb{Z})$ *or* $\#E(\mathbb{Z}/q\mathbb{Z})$ is semi-smooth. That is, the ISC looks for p and q simultaneously. On the other hand, the Jacobi

	Jacobi Symbol Cont.			Impr. Standard Cont.			
Size of p	B	D	Time	B	D	Time	$\text{Time}_{\text{JSC}}/\text{Time}_{\text{ISC}}$
10^9	325	175	0.52s	349	4297	0.27s	1.9
10^{10}	550	7500	1.2s	411	8861	0.56s	2.1
10^{11}	950	10000	2.0s	659	17981	1.1s	1.8
10^{12}	n/av	n/av	n/av	997	31489	1.9s	
10^{13}	2300	50000	8.5s	1439	54617	4.5s	1.9
10^{14}	3400	80000	16.5s	2111	89501	8.1s	2.0

TABLE 5. Comparison of running times.

Symbol Continuation is successful only in the event of semi-smoothness modulo p. This might explain why, despite all our efforts in optimizing the method, the JSC takes twice as long as the ISC.

7. A Few Concluding Remarks

Our implementation of the Jacobi Symbol Continuation of the Elliptic Curve Method for factoring numbers of the form pq^2 results in running times that are about twice as high as for the Improved Standard Continuation. This is for p and q of approximately the same size, and between 10^9 and 10^{14}.

However, we hope our work stimulates further experimental study of this continuation, since there can well be applications where the Jacobi Symbol Continuation is indeed superior. First, consider the case that p is much smaller than q. Then both the ISC and the JSC look for only one factor, p. Second, consider much larger prime factors p and q, so that the good asymptotic properties of the running time of the Jacobi Symbol Continuation have a chance to kick in. Notice that then the reference method would be the Fast Fourier Transform Continuation.

References

[BP96] E. Bach and R. Peralta. Asymptotic semismoothness probabilities. *Mathematics of Computation*, 65:1701–1715, 1996.

[Bre80] R. P. Brent. An improved Monte Carlo factorization algorithm. *BIT*, 20:176–184, 1980.

[CDL+00] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, Chris and Craig Putnam, and P. Zimmermann. Factorization of a 512-bit RSA modulus. In *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2000.

[Coh93] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, Berlin, 1993.

[CP01] R. Crandall and C. Pomerance. *Prime Numbers. A Computational Perspective*. Springer-Verlag New York, 2001.

[FKM+00] E. Fujisaki, T. Kobayashi, H. Morita, H. Oguro, T. Okamoto, S. Okazaki, D. Pointcheval, and S. Uchiyama. EPOC – efficient probabilistic public-key encryption. Submission to NESSIE, 2000. https://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions.html.

[Har60] B. Harris. Probability distributions related to random mappings. *Annals of Math. Statistics*, 31:1045–1062, 1960.

[Hue00] D. Huehnlein. Quadratic orders for NESSIE – overview and parameter sizes of three public key families. Technical Report TI-3/00, TU Darmstadt, Germany, 2000. http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/Welcome.html.

[Len87] H. W. Lenstra Jr. Factoring integers with elliptic curves. *Ann. of Math.*, 126:649–673, 1987.

[Len01] A. K. Lenstra. Unbelievable security. Matching AES security using public key cryptosystems. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[LiD00] LiDIA Group, Technische Universität Darmstadt, Darmstadt, Germany. *LiDIA - A library for computational number theory, Version 2.0*, 2000.

[Mon92] P. Montgomery. *An FFT extension of the elliptic curve method of factorization*. PhD thesis, University of California, Los Angeles, 1992.

[Mue95] A. Mueller. Eine FFT-Continuation für die elliptische Kurvenmethode. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany, 1995. Diplomarbeit.

[MvOV96] A. Menezes, P. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[Per01] R. Peralta. Elliptic curve factorization using a "partially oblivious" function. In K.-Y. Lam, I. Shparlinski, H. Wang, and C. Xing, editors, *Cryptography and Computational Number Theory: Workshop in Singapore 1999*, volume 20 of *Progress in Computer Science and Applied Logic*, pages 123–128. Birkhäuser, 2001.

[PO96] R. Peralta and E. Okamoto. Faster factoring of integers of a special form. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, E79-A(4), 1996.

[Pol75] J. M. Pollard. A Monte Carlo method for factorization. *BIT*, 15(3):331–335, 1975.

[Pol78] J. M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978.

[Tak98] T. Takagi. Fast RSA-type cryptosystem modulo $p^k q$. In *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 318–326. Springer-Verlag, 1998.

[Tes98a] E. Teske. A space efficient algorithm for group structure computation. *Mathematics of Computation*, 67:1637–1663, 1998.

[Tes98b] E. Teske. Speeding up Pollard's rho method for computing discrete logarithms. In *Algorithmic Number Theory Seminar ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 541–554. Springer-Verlag, 1998.

[Tes01] E. Teske. On random walks for Pollard's rho method. *Mathematics of Computation*, 70:809–825, 2001.

[Zim00] P. Zimmermann. New ECM record: up to 60 digits. Communication over the NMBRTHRY list, 5 Jan 2000.

Universität Karlsruhe, Institut für Algorithmen und Kognitive Systeme, Postfach 6980, 76128 Karlsruhe, Germany

E-mail address: `peter@ebinger.de`

University of Waterloo, Department of Combinatorics and Optimization, Waterloo, Ontario, Canada N2L 3G1

E-mail address: `eteske@uwaterloo.ca`