Annales Univ. Sci. Budapest., Sect. Comp. 41 (2013) 295–305

PRODUCING CLASS NUMBERS FOR THE ATKIN–MORAIN PRIMALITY TEST

Gábor Román (Budapest, Hungary)

Dedicated to Professors Zoltán Daróczy and Imre Kátai on their 75th birthday

Communicated by Antal Járai

(Received May 10, 2013; accepted June 12, 2013)

Abstract. During the precomputation phase of the Atkin–Morain primality test, a rather big set of negative fundamental discriminants, moreover the class numbers related to these discriminants are produced. In this article, we present a method which allows us to quickly produce negative fundamental discriminants representable on at most eight bytes. Furthermore we give two methods to compute the class numbers related to these discriminants. With the first method, we can quickly acquire the exact value of the class numbers related to discriminants with absolute value up to approximately 2^{40} (depending on the underlying processor), while with the second method we can estimate the order of magnitude of the class numbers.

1. Introduction

Nowadays, with the elliptic curve primality tests, we can decide the primality of natural numbers, even if the examined numbers have thousands of

Key words and phrases: Elliptic curve primality proving, negative fundamental discriminants, class numbers.
2010 Mathematics Subject Classification: 11Y11.

decimal digits. These primality tests form the keystone of modern cryptography and computational number theory, so the efficient implementation of these methods is essential. The first primality test which uses elliptic curves is from Goldwasser and Kilian [9]. Atkin and Morain gave a faster variant of this primality test [1]. Hereinafter, we look at why the Atkin–Morian primality test is a leap forward, compared to the one created by Goldwasser and Kilian, without going into the detailed description of these tests.

The Goldwasser–Kilian primality test starts as follows: let $n \in \mathbb{N}$ be the number which we want to test for primality. (Here and henceforth $n \neq 1$ and n is relatively prime to 6.) We choose an elliptic curve over $\mathbb{Z}/n\mathbb{Z}$ randomly and we use the cardinality of the points on this elliptic curve for further computations. Counting points on a random elliptic curve is a hard task, although there exists an algorithm for it with polynomial running time (see Schoof's [12] article). Hereafter we will denote the number of points on the random elliptic curve with m.

The Atkin–Morain primality test offers us a significantly faster solution, where first the proper value of m is acquired, then the necessary elliptic curve is computed for this cardinality. This way, we can skip the time expensive algorithm for counting points on the given elliptic curve. During the test, the computation of the proper m cardinality is done with a big set of negative fundamental discriminants. These negative fundamental discriminants are such $D < -1$ integers, which satisfy one of the following two conditions: either $D \equiv 1 \pmod 4$ and D is square-free, or $D \equiv 0 \pmod 4$ and $D/4$ is square-free. If the proper m value is successfully acquired with a given D negative fundamental discriminant, then comes the computation of the required elliptic curve. The probability of success during this step is $1/(2h(D))$, where $h(D)$ is the class number related to the mentioned D discriminant. After the computation of the proper elliptic curve, the Atkin–Morain primality test continues like the Goldwasser–Kilian primality test.

During the precomputation phase of the Atkin–Morain primality test, a rather big set of D negative discriminants and the $h(D)$ class numbers related to these discriminants are produced. This step is independent of the number which we want to test for primality, so we can execute this precomputation phase before the test is performed.

Usually these exact primality tests are not used in practice, because the probabilistic primality tests are much faster and the probability of error is negligible. Most of the computer algebra systems are not able to prove primality. One exception is the Magma system, which has the implementation of the Atkin elliptic curve primality test as a built-in function, which is not able to test numbers with thousands of digits for primality. But with the application of the proper amount of fundamental discriminants, the ability of Magma to prove primality can be significantly improved. Gábor Farkas and Gábor Kallós

met the problem of proving the primality of numbers with more then 1000 decimal digits during the factorisation of certain elements of the generalized Pascal-triangle. In their [7] article, which is published in 2008, they used a François Morain ECPP implementation for primality testing. In 2011 with the partnership of Gyöngyvér Kiss, they used Magma for a similar goal, which details can be read in [8]. They experienced the following phenomena: the exact test of a 1030 digit long number is not finished even in a day. The explanation is, that Magma (the used version) applied relatively few, fixed number of fundamental discriminants. As a result, it had to use more time for the factorisation of the curve orders. The problem is solved through the application of more fundamental discriminants, so the test is finished in one and a half hour.

2. Producing fundamental discriminants

We describe a method which allows us to quickly produce negative fundamental discriminants which can be represented on at most eight bytes. These discriminants form a sufficiently big set of negative fundamental discriminants for the Atkin–Morain primality test. The negative fundamental discriminants are produced from a given interval. To decide that a number from the given interval is a fundamental discriminant, first we have to check that one of the mentioned congruences holds, then comes the verification of square-freeness. We can verify square-freeness with factorisation, but for that we need a fast primality test which is reliable in the mentioned order of magnitude.

2.1. Testing fundamental discriminants for primality

During the primality testing of the D negative fundamental discriminants we mainly use the Miller–Rabin probabilistic primality test (see for example section 8.2 in Cohen's [6] book). We achieve exactness through the usage of strong pseudoprimes (for more details see Jaeschke's [10] article). Hereinafter we look at how this roughly happens.

Let $p_1, p_2, \ldots, p_k$ be the first k primes. Now let ψ_k be the smallest natural number which is strong pseudoprime to the $p_1, p_2, \ldots, p_k$ bases. So if our goal is to decide the primality of an $n < \psi_k$ natural number, it is sufficient to perform only the Miller–Rabin tests with the $p_1, p_2, \ldots, p_k$ bases on n. Jaeschke showed, among other things, that the

$$\psi_8 = 341\ 5500\ 7172\ 8321$$

equality holds, furthermore he gave upper bounds on the values of ψ_9, ψ_{10} and ψ_{11}. So for the natural numbers which are not greater than the value of ψ_8 (approximately the numbers which can be represented on 44 bits), we obtain a fast and exact primality test. But for all numbers which can be represented on eight bytes, this bound is insufficient. We present two methods for the exact primality testing of numbers which can be represented on more than five but at most eight bytes.

Our first option to guarantee exactness is to use more primes during the application of the Miller–Rabin primality test. In this case it's worth relying on the assumptions given in the article of Zhang [13]. By these assumptions it is sufficient to use at most the value of ψ_{12} during the application of the Miller–Rabin primality test for the numbers which can be represented on at most eight bytes. Our second option is the application of a Lucas pseudoprime test (see the 12th chapter in Bressoud's [4] book). During the application of this primality test, we have to compute the appropriate terms of a Lucas sequence modulo the number which we want to test. There exists an efficient algorithm based on the square-and-multiply principle for calculating the terms of the Lucas sequence (see algorithm 8.3 in Bressoud's book), so the primality test can be implemented efficiently.

Finally we note that the application of the Miller–Rabin primality test together with the Lucas pseudoprime test produces a primality test which is even more reliable and still quite fast (see the article of Baillie and Wagstaff [3]). Furthermore the article of Baillie and Wagstaff gives a guideline to the implementation of the Lucas pseudoprime test.

2.2. Factorisation of fundamental discriminants

Below we describe a method which allows us to compute a prime factor of a natural number which can be represented on at most eight bytes. During the algorithm we use Pollard's ϱ method (see section 8.5 in Cohen's book). Pollard's ϱ method uses an $x_{m+1} = f(x_m)$ iteration with some x_0 initial value, where f is a polynomial with integer coefficients. In practice, the $f(x) = x^2 + c$ polynomial is used with the $x_0 = 1$ initial value. Here $c \geq 1$ is an integer parameter which can be chosen freely. Cohen notes in his book, that if one fails to apply the method successfully, then it is worth to change the c parameter instead of changing the x_0 initial value. With the application of the described primality test and Pollard's ϱ method, the main steps of the method that computes a prime factor, are the following:

1. First, using small primes, we try to extract a prime factor from the number which we want to factorise, with trial division. If we succeed, then we have found a prime factor of the number which we want to factorise and the algorithm terminates.

2. If we fail during the previous step, then we perform the described primality test on the number which we want to factorise. If we ensure the reliability of the primality test with the Lucas pseudoprime test, then with the trial division step, the applied primality test matches the one that is recommended by Baillie and Wagstaff. If we find out that the number which we want to factorise is a prime, then the algorithm terminates.

3. If the number which we want to factorise is not a prime, then we perform another trial division step, but with greater primes this time.

4. If we can't find a prime factor of the number which we want to factorise, then we apply Pollard's ϱ method with an increasing c value, where $c = 2$ initially.

5. Now we test the non-trivial divisor which we have found during the application of Pollard's ϱ method, for primality. If the non-trivial divisor is a prime, then the algorithm terminates. Otherwise we apply this algorithm recursively on the non-trivial divisor because we now know that it is composite.

The bounds applied in the trial division steps during the algorithm (that is, how many primes do we apply during the trial divisions) affects the reliability and partly the speed of the factorisation. For example a preferable setting is to use the primes lower than 256 during the first trial division, and the primes greater than 256 but lower than 1000 during the second trial division. This way the first trial division step can be performed quickly, and a significant amount of composite numbers can be factorised in this step. It's not worth to choose a greater bound during the second trial division step, because it makes the factorisation much slower. It's advisable to stay near these bounds, because the running time of Pollard's ϱ method is heuristically proportional to the square root of the smallest prime factor of the number which we want to factorise, so the bigger prime factors (or non-trivial divisors) can be found quickly too. The reliability of the algorithm is influenced by the applied primality test. With the application of the proper amount of primes during the Miller–Rabin primality test or with the usage of the Lucas pseudoprime test, we can guarantee that the primality test will not identify a composite number as a prime by mistake. If the underlying primality test is reliable, then the algorithm will surely find a prime factor of the number which we want to factorise. Namely because of the magnitude of the numbers which we want to factorise, Pollard's ϱ method will surely finds a non-trivial divisor sooner or later with some c parameter.

3. Producing class numbers

After the production of a negative fundamental discriminant, we continue with the computation of the related class number. Cohen, in the fifth chapter of his book, gives a wide spectrum of algorithms, which allows us to compute class numbers related to negative fundamental discriminants. By the nature of the task we need such algorithms, which helps us to quickly compute vast amount of class numbers. Another requirement is that the computation of the class numbers related to different discriminants should be independent, so the calculations could be parallelised. This way, if we want to produce negative fundamental discriminants and the related class numbers from a big interval, then we can divide the interval into smaller pieces, and we can schedule the processing of these smaller pieces between several processors or cores. The algorithms in Cohen's book were examined with these consideration in mind. According to these, we give two methods.

The most suitable algorithm in Cohen's book for the task is in section 5.3.3, and it uses analytic formulas to compute class numbers. Louboutin gives a faster variant of this method in his [11] article. With this method, we can compute the exact value of a class number related to a negative fundamental discriminant. But the application of this method depends on the size of D (in absolute value). This method only worked with acceptable speed in the case of negative fundamental discriminants representable on at most five bytes during the tests on an Intel Core i5-460M processor. The second method is an approximating one and it is based on section 5.4.3 in Cohen's book. The speed of this method is not influenced by the size of the D negative fundamental discriminant, so we can compute class numbers related to arbitrary D discriminants with it, at least approximately. This is not necessarily a problem, because while we want to compute the elliptic curve during the Atkin–Morian primality test the probability of success depends on the magnitude of the class number.

3.1. Producing exact class numbers

This method is based on the computation of the Dirichlet L-functions. Let D be a negative discriminant (which is not necessarily fundamental). Now we define the Dirichlet L-functions, using the Kronecker-symbol with the

$$L_D(s) = \sum_{n \geq 1} \left(\frac{D}{n}\right) n^{-s}$$

formula. This series converges for $\Re(s) > 1$, and defines an analytic function which can be analytically continued to the whole complex plane to an entire function. The connection between the L-functions and the class numbers is due to the following theorem of Dirichlet.

Theorem 3.1 (Dirichlet)**.** *If D is a negative discriminant (not necessarily fundamental), then*

$$L_D(1) = \frac{2\pi h(D)}{w(D)\sqrt{|D|}},$$

where

$$w(D) = \begin{cases} 2, & \text{if } D < -4 \\ 4, & \text{if } D = -4 \\ 6, & \text{if } D = -3 \end{cases}$$

and $h(D)$ is the class number related to the D discriminant.

So the computation of a class number related to a given D negative fundamental discriminant can be reduced to the computation of the value of $L_D(1)$. Cohen in section 5.3.14 in his book gives the

$$h(D) = \sum_{n \geq 1} \left(\frac{D}{n}\right) \left(\operatorname{erfc}\left(n\sqrt{\frac{\pi}{|D|}}\right) + \frac{\sqrt{|D|}}{\pi n} e^{-\pi n^2/|D|} \right)$$

formula for $D < -4$ fundamental discriminants, where the erfc function is the complementary error function. Moreover, Cohen gives the bound depending on the given D discriminant, for which we have to compute the sum in the formula to achieve a smaller relative error than 0.5, so the sought $h(D)$ will be the nearest integer to the result.

Louboutin gives a faster variant of this method in his article, which eliminates the computation of the complementary error function and it shows that the repeated, direct computation of the exponential function can be omitted too. Let χ be a primitive Dirichlet character modulo $f > 1$. Using Louboutin's notations, let

$$L(s, \chi) = \sum_{n \geq 1} \frac{\chi(n)}{n^s} \quad \text{and} \quad S_n(\chi) = \sum_{k=1}^{n} \chi(k),$$

furthermore let $\alpha = \sqrt{\pi/f}$, $e_n = e^{-\alpha^2 n^2}$ and finally let

$$B(t, M, f) = \sqrt{\frac{f\,(t\,\log(f/\pi) + M)}{\pi}} = \alpha^{-1}\sqrt{M - 2t\,\log\alpha}.$$

The following theorem is due to Louboutin.

Theorem 3.2 (Louboutin)**.** *Let $M \geq 1$ be given, let χ be a primitive odd Dirichlet character of conductor f, let m be the least rational integer greater than or equal to $B\left(\frac{1}{2}, M, f\right) = \mathcal{O}\left(f^{0.5+\epsilon}\right)$ and set*

$$L_M(0,\chi) = \frac{1}{\sqrt{\pi}}\left(\frac{W(\chi)}{\alpha}\sum_{n=1}^{m}\frac{\overline{\chi}(n)}{n}e_n + \alpha\sum_{n=1}^{m}(e_n + e_{n+1})S_n(\chi)\right).$$

Then,

$$|L(0,\chi) - L_M(0,\chi)| \leq \frac{3}{2\sqrt{\pi}}e^{-M} + \frac{3}{8\sqrt{f}}.$$

If $f = |D|$, then $\overline{\chi} = \chi$ is the $(D|n)$ Kronecker-symbol, $W(\chi) = 1$ and $L(0,\chi)$ is equal to the $h(D)$ class number related to D negative fundamental discriminant, so it can be approximated with $L_M(0,\chi)$. If we set M to 1 then the absolute error will be less than 0.5, even in the case of $f = 4$, so after the computations, the nearest integer to the result will be the required class number. Because the difference sequence of the square numbers form an arithmetic sequence, using the identities of exponentiation, the computation of e_n can be reduced to multiplication, so we can avoid the computation of the exponential function after the first few steps.

The m bound in the theorem increases fast as f increases, so in the case of negative fundamental discriminants with big absolute values, the method shows a significant slowdown. The tests were performed on an Intel Core i5-460M processor, where the boundary for which the method should be applied turned out to be approximately 2^{40}. Around this boundary the method can only compute a few (around 10) class numbers under a second on the mentioned processor.

3.2. Estimating class numbers

If we want to compute the class numbers for negative fundamental discriminants with big absolute value, the preceding method becomes useless because of the increased computation time. Cohen gives an approximation method in section 5.4.3 of his book, which he uses during the implementation of Shank's baby-step giant-step method. We can read about this approximation method in more detail in the article of Bach [2], and the advantage is that it only uses prime numbers for the computations. The $L_D(s)$ function can be written as the

$$L_D(s) = \prod_{p\in\mathbb{P}}\left(1 - \frac{\left(\frac{D}{p}\right)}{p^s}\right)^{-1}$$

Euler-product, where $\mathbb{P}$ is the set of prime numbers. So, using the connection between the value of $L_D(1)$ and the value of $h(D)$, if we compute the

$$\left\lfloor \frac{\sqrt{|D|}}{\pi} \prod_{\substack{p\in\mathbb{P}\\ p\leq P}} \left(1-\frac{\left(\frac{D}{p}\right)}{p^s}\right)^{-1} \right\rfloor$$

product to a given P boundary, then we get an estimating value of the class number related to the $D<-4$ discriminant. In practice, Shank noticed experimentally that the relative error is around 0.001 when $P=2^{17}$, but Cohen recommends the $P=2^{18}$ boundary during the computations. Bach, based on the extended Riemann hypothesis, gives the

$$|\ln L_D(1)-\ln B(P)| = \mathcal{O}\left(\frac{\ln(|D|P)}{\sqrt{P}}\right)$$

equality in his article, where

$$B(P)=\prod_{\substack{p\in\mathbb{P}\\ p\leq P}} \left(1-\frac{\left(\frac{D}{p}\right)}{p^s}\right)^{-1},$$

when the $P\geq 2$ inequality holds. Moreover, Bach gives the explicit version of this error estimate, which can be written as the

$$|\ln L_D(1)-\ln B(P)| \leq \frac{A\cdot\ln|D|+B\cdot\ln P}{\sqrt{P}}$$

inequality, where the A and B constants depend on the given P boundary. These constants can be found in the 4th table of Bach's article. Let us denote the right side of the previous inequality with $C(P)$. Based on section 9.3.4 in the book of Buchmann and Vollmer [5], if $C(P)<\log 2$, then

$$\left|\frac{L_D(1)}{B(P)}-1\right| < \frac{C(P)}{1-C(P)}.$$

Using this with the $P=2^{18}$ boundary (in this case $A=1.562$ and $B=0.655$ according to Bach's article) the method's relative error is less than 0.18 in the case of the negative fundamental discriminants we want to process. So with this method, we can give an estimate on the order of the magnitude of the class number related to a given discriminant.

Because the chosen P boundary is independent from the given negative fundamental discriminant, the running time of this algorithm is nearly the same

for all discriminants. During the tests, while using the $P = 2^{18}$ boundary, the algorithm was able to compute about 300 class numbers in a second. Bach gives another method in his article (this is the one that is discussed and analysed by Buchmann and Vollmer), which provides a better relative error, but this method requires much more computation.

4. Conclusion and results

With the presented method, we are able to quickly produce negative fundamental discriminants which can be represented on at most eight bytes. Using the method of Louboutin, we can quickly compute the class numbers related to these negative fundamental discriminants with absolute value up to 2^{40}, depending on the underlying processor. Above this boundary, we can give an estimate on the magnitude of the class numbers. If we have access to more processors or cores, we can parallelise the computations.

With the methods presented in this article, all the negative fundamental discriminants from 7 to 2^{30} were produced and the class numbers related to these discriminants were computed. This interval is processed in 2^{28} slices on the mentioned processor with three or four threads. The computations took around seven days.

Acknowledgement. The author wishes to thank Gábor Farkas for his helpful guidance, and Emil Vatai for his corrections and suggestions.

References

[1] **Atkin, A.O.L. and F. Morain,** Elliptic Curves And Primality Proving, *Math. Comp.*, **61(203)** (1993), 29–68.

[2] **Bach, E.,** Improved Approximation for Euler Products, *Canadian Mathematical Society Conference Proceedings*, **15** (1995), 13–28.

[3] **Baillie, R. and S.S. Wagstaff, Jr.,** Lucas Pseudoprimes, *Math. Comp.*, **35(152)** (1980), 1391–1417.

[4] **Bressoud, D.M.,** *Factorization and Primality Testing*, Springer-Verlag, New York, Berlin, Heidelberg, 1989.

[5] **Buchmann, J. and U. Vollmer,** *Binary Quadratic Forms: An Algorithmic Approach*, Springer-Verlag, New York, Berlin, Heidelberg, 2007.

[6] **Cohen, H.,** *A Course in Computational Algebraic Number Theory*, Springer-Verlag, New York, Berlin, Heidelberg, 1996 (third, corrected print).

[7] **Farkas, G. and G. Kallós,** Prime Numbers in Generalized Pascal Triangles, *Acta Technica Jaurinensis*, **1(1)** (2008), 109–117.

[8] **Farkas, G. and G. Kallós and Gy. Kiss,** Large Primes in Generalized Pascal Triangles, *Acta Universitatis Sapientiae Informatica*, **3(2)** (2011), 158–171.

[9] **Goldwasser, S. and J. Kilian,** Almost all primes can be quickly certified, in: Juris Hartmanis (Ed.) *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, Berkeley, California, USA, 1986, 316–329.

[10] **Jaeschke, G.,** On strong pseudoprimes to several bases, *Math. Comp.*, **61(204)** (1993), 915–926.

[11] **Louboutin, S.,** Computation of class numbers of quadratic number fields, *Math. Comp.*, **71(240)** (2001), 1735–1743.

[12] **Schoof, R.,** Counting Points on Elliptic Curves Over Finite Fields, *Théorie des Nombres de Bordeaux*, **7** (1995), 219–254.

[13] **Zhang, Z.** Two kind of strong pseudoprimes up to 10^{36}, *Math. Comp.*, **76(260)** (2007), 2095–2107.

G. Román
Department of Computer Algebra
Faculty of Informatics
Eötvös Loránd University
Pázmány Péter sétány 1/C
H-1117 Budapest
Hungary
romangabor@caesar.elte.hu