Gaudry's Variant against C_{ab} Curves

Seigo Arita

C&C Media Research Laboratories, NEC, Kawasaki Kanagawa, Japan,
arita@ccm.cl.nec.co.jp

Abstract. Gaudry has described a new algorithm (Gaudry's variant) for the discrete logarithm problem (DLP) in hyperelliptic curves. For hyperelliptic curves of small genus on finite field GF(q), Gaudry's variant solves for the DLP in $O(q^2 \log^{\gamma}(q))$ time. This paper shows that C_{ab} curves can be attacked with a modified form of Gaudry's variant and presents the timing results of such attack. However, Gaudry's variant cannot be effective in all of the C_{ab} curve cryptosystems, this paper provides an example of a C_{ab} curve that is unassailable by Gaudry's variant.

1 Introduction

Gaudry has described a new algorithm (Gaudry's variant) for the discrete logarithm problem (DLP) in hyperelliptic curves [7]. Gaudry's variant uses the method for Pollard's rho algorithm [12] with the function field sieving algorithm of Adleman,DeMarrais, and Huang [1]. Gaudry's variant solves the DLP in hyperelliptic curves of genus g defined on the finite field F_q in time $O(q^2 \log^{\gamma}(q))$ when the genus g is sufficiently small in comparison to the order q of the definition field.

Arita and Galbraith et al. have described addition algorithms on the Jacobian group of C_{ab} and superelliptic curves respectively, and have demonstrated algorithm applications in discrete-log-based public key cryptosystems [3,6]. This paper shows that C_{ab} and superelliptic curves can be attacked by a modified Gaudry's variant and presents timing results from the attack.

With hyperelliptic or C_{ab} curve cryptosystems, researchers usually select a sufficiently large genus so that definition fields are less than one word in size to hasten computations [14,3]. Gaudry's variant has excluded out this conventional hastening method. However, Gaudry's variant cannot be effective in all of the non-elliptic algebraic curve cryptosystems. This paper provides an example of a C_{ab} curve that is unassailable by Gaudry's variant.

2 Gaudry's Variant

Take hyperelliptic curve $C : y^2 = x^{2g+1} + a_1 x^{2g} + \cdots a_{2g+1}$ of genus g defined on finite filed F_q. Suppose the genus g is sufficiently small in comparison to the order q of the definition field. Let J_C denote the Jacobian group of the hyperelliptic

H. Imai, Y. Zheng (Eds.): PKC 2000, LNCS 1751, pp. 58–67, 2000.

curve. To handle with the DLP, look for integer λ that satisfies $D_2 = \lambda D_1$ for two elements D_1 and D_2 in J_C.

In Pollard's rho algorithm, we calculate the random linear sums $R_i = \alpha_i D_1 + \beta_i D_2 \ \ (i = 1, 2, \cdots)$ of D_1 and D_2 step by step through a random walk, and wait for a collision $R_i = R_j$. Once it occurs, from $\alpha_i D_1 + \beta_i D_2 = \alpha_j D_1 + \beta_j D_2$, we get $D_2 = (\alpha_i - \alpha_j)/(\beta_j - \beta_i)D_1$, and λ is obtained.

In Gaudry's variant, just as in the case of rho algorithm, we calculate the linear sums $R_i = \alpha_i D_1 + \beta_i D_2 \ \ (i = 1, 2, \cdots)$ of D_1 and D_2 step by step through a random walk. However, we gather smooth R_i value's instead of waiting for a collision. Element D in J_C is called smooth when D is the sum of F_q rational points on C; that is, we take all of the F_q rational points as a factor base. Let all of the F_q rational points on C be $\{P_1, P_2, ..., P_w\}$. Then, w-dimensional vector $M_i = (m_{i,1}, \cdots, m_{i,w})$ corresponds to every smooth R_i through $R_i = \sum_k m_{i,k} P_k$. In the polynomial expression $R_i = [u_i(x), v_i(x)]$ from Cantor's algorithm [4], R_i is smooth if and only if the polynomial $u_i(x)$ is factored into the linear product $\prod_k (x - c_k)$ over F_q, and then we get $R_i = \sum_k (c_k, v(c_k))$. Therefore, by calculating all of the F_q rational points $\{P_1, P_2, \cdots, P_w\}$ in advance at a complexity of $O(q)$, vector M_i is easily obtained.

When g is sufficiently small in comparison to q, about $1/g!$ of all of the elements in J_C are smooth ([7]Prop.4). We encounter a smooth R_i for every $g!$ steps (remember g is small), and we obtain w' ($\geq w$) smooth R_i values after $g! \cdot w'$ steps. The w' vectors $M_i = (m_{i,1}, \cdots, m_{i,w})$ $(i = 1, \cdots, w')$ then become linearly dependent, and by solving the w-dimensional linear equation, we obtain values for γ_i $(i = 1, \cdots, w')$ that satisfy

$$\sum_{i=1}^{w'} \gamma_i M_i = 0.$$

These values for γ_i produce λ through the equation $R_i = \sum_k m_{i,k} P_k$;

$$\lambda = -\sum_i \gamma_i \alpha_i / \sum_i \gamma_i \beta_i.$$

In calculating Gaudry's variant, the most complex step is that of solving the w-dimensional linear equation $\sum_i \gamma_i M_i = 0$ $(i = 1, 2, \cdots, w')$. The matrix $(m_{i,k})$ has a size equal to about $q \times q$ and is sparse since there are only g non-zero elements in each row; thus the linear equation can be solved in q^2 steps. The complexity of Gaudry's variant is given as $O(q^2 \log^\gamma(q))$.

When C has non-trivial automorphism ϕ, Gaudry's variant becomes more powerful. Let m denote the order of ϕ. In this case, all of the F_q rational points are unnecessary. Only the representatives of orbits in F_q rational points for the action of ϕ are needed as a factor base. The number of the orbits is q/m, so the complexity of Gaudry's variant becomes $O((q/m)^2 \log^\gamma(q))$.

However, automorphisms in C can be ignored, according to a theorem from Arbarello et al [2].

Theorem 1. *The order of the automorphism group of a smooth algebraic curve of genus* ≥ 2 *is at most* $84(g-1)$.

Using this theorem, we know that the effect of automorphisms can be ignored by expanding the size of the definition field with $\log_2(84(g-1))$ more bits.

3 C_{ab} and Superelliptic Curve

The C_{ab} curve is a nonsingular affine curve with the equation

$$\sum_{0\le i\le b, 0\le j\le a, ai+bj\le ab} \alpha_{i,j} x^i y^j = 0,$$

where both $\alpha_{b,0}$ and $\alpha_{0,a}$ are not equal to zero [10]. Arita has described an addition algorithm for the Jacobian group of a C_{ab} curve in terms of ideals of the coordinate ring; he has also proposed discrete-log-based public key cryptosystems using C_{ab} curves [3]. For a C_{ab} curve with 160 bits of a Jacobian group, timing results from the addition algorithm are listed in Tables 1,2, and 3.

Table 1. Performance for the C_{35} curve in ms at 266 MHz, using a Pentium II chip.

	simple	random
sum	3.39	3.65
double	3.76	4.21
scalar	862	958

Table 2. Performance for the C_{37} curve in ms at 266 MHz, using a Pentium II chip.

	simple	random
sum	1.15	1.24
double	1.15	1.28
scalar	273	300

Table 3. Performance for the $C_{2,13}$ curve in ms at 266 MHz, using a Pentium II chip.

	simple	random
sum	0.70	0.73
double	0.65	0.68
scalar	158	167

In the tables, "simple" denotes C_{ab} curves with equation $Y^a + \alpha X^b + \beta$, and "random" denotes randomly chosen C_{ab} curves. "Sum", "double", and "scalar"

denote addition, doubling, and scalar multiplication of random elements, respectively.

A superelliptic curve is a nonsingular affine curve of the equation

$$y^n = a_\delta x^\delta + \cdots + a_0,$$

where n is prime to the characteristics of the definition field, and n and δ are prime to each other [6]. Galbraith et al. have described an addition algorithm in the Jacobian group of a superelliptic curve in terms of lattice computation [6].

Given these definitions, C_{ab} curves clearly include superelliptic curves. Therefore, only C_{ab} curves will be examined.

4 Application of Gaudry's Variant to C_{ab} Curves

I modified Gaudry's variant and applied it to C_{ab} curves. The problems I encountered were how to decide if a given element in a Jacobian group is smooth or not, and, if it is smooth, how to represent the element as a sum of F_q rational points.

For example, with a C_{37} curve, where the genus is 6, element R in the Jacobian group is expressed as an ideal of the coordinate ring using the Gröbner basis with respect to the C_{37} order [3]:

$$\begin{aligned} R = \{&a_0 + a_1x + a_2x^2 + a_3y + a_4x^3 + a_5xy + x^4, \\ &b_0 + b_1x + b_2x^2 + b_3y + b_4x^3 + b_5xy + x^2y, \\ &c_0 + c_1x + c_2x^2 + c_3y + c_4x^3 + c_5xy + y^2\}. \end{aligned}$$

Here, a_i, b_i, and c_i are elements in the definition field. The common zeroes of these three equations are six points on the C_{37} curve, that comprise R. When definition field F_q is large enough, for almost any R in the Jacobian group, the six points comprising R have distinct x-coordinates to each other. So, almost any R can be expressed as common zeroes of two polynomials:

$$\begin{aligned} R = \{&\text{sixth degree polynomial of } x, \\ &y + (\text{fifth degree polynomial of } x)\}, \end{aligned}$$

just as in hyperelliptic curves. The expression is nothing but the Grobner basis of (the ideal corresponding to) R with respect to the lexicographic order.

Therefore, in Gaudry's variant against C_{ab} curves, for almost any R_i, the decision regarding the R_i value's smoothness and representation as a sum of F_q rational points follows the same pattern as for hyperelliptic curves, after translating the Gröbner basis of R_i w.r.t. C_{ab} order to another Gröbner basis w.r.t. lexicographic order. It is well known that Gröbner bases can be effectively translated between distinct monomial orders. If exceptional values for R_i are found, they are simply discarded. It is known that Gaudry's variant can also be applied to C_{ab} curves.

Table 4. 93 bits of a C_{37} curve over 17 bits of a prime field

finite field	F_{84211}
defining equation	$1 + 24740x^7 + 32427y^3 = 0$
genus	6
order of Jacobian	$43 \cdot 80689706230162396053189866 17$
order of automorphism	$3 \cdot 7$

I implemented the modified Gaudry's variant with C language and the PARI-GP [11], and then tested it against the C_{37} curve in Table 4. The curve has 93 bits of a Jacobian group over 17 bits of a prime field. Let ζ_3 and ζ_7 denote the primitive seventh root of one and the third root of one, respectively. Since the curve has the automorphism $\phi(x, y) = (\zeta_7 x, \zeta_3 y)$ of order 21, the number of rational points in a factor base should be $84211/21 = 4010.\cdots$ or more. I needed to collect 4011 or more smooth elements and then solve about 4011 dimensional spare linear equations. For solving these linear equations, I used the Lanczos algorithm [9].

I randomly generated two elements, D_1 and D_2, in the Jacobian group:

$$\begin{aligned}
D_1 = \{&x^4 + 77465x^3 + 75875x^2 + (37117y + 57992)x \\
&+ (42876y + 21588), \\
&5485x^3 + (y + 79222)x^2 + (4298y + 50456)x \\
&+ (36882y + 81869), \\
&41971x^3 + 64608x^2 + (26263y + 16207)x \\
&+ (y^2 + 42778y + 62216)\}, \\
D_2 = \{&x^4 + 64296x^3 + 44620x^2 + (29434y + 15779)x \\
&+ (61013y + 42557), \\
&51156 * x^3 + (y + 32172)x^2 + (62401y + 22153)x \\
&+ (78055y + 13056), \\
&79116x^3 + 5028x^2 + (69977y + 21979)x \\
&+ (y^2 + 75761y + 2009)\}.
\end{aligned}$$

Then, running the modified Gaudry's variant, I obtained

$$\lambda = 40822718041348743469836704 15$$

for $D_2 = \lambda D_1$ in the time given in Table 5. The average and variance of the number of steps needed to produce every smooth element were 848.265 and 680786, respectively, reasonable results when compared to Gaudry's theoretical estimate of $6! = 720$ [7].

Table 5. Timing results of modified Gaudry's variant against the C_{37} curve from Table 4 using a 266-MHz Pentium II chip.

Collection of rational points (PARI-GP)	5 m 13 s
Collection of smooth elements (C)	2 h 33 m 6 s
Solving linear equation (C)	32 m 2 s
Total	3 h 11 m 21 s

5 C_{ab} Curves that are Unassailable by Gaudry's Variant

Let a and b be distinct prime numbers. Let $C(p, \alpha, \beta)(= C(p, a, b, \alpha, \beta))$ denote a C_{ab} curve over prime field F_p with the equation

$$\alpha Y^a + \beta X^b + 1 = 0.$$

In this section, I will construct a C_{ab} curve $C(p, \alpha, \beta)$ of a small genus, that is secure against Gaudry's variant. Let h be the order of the Jacobian group of $C(p, \alpha, \beta)$. Conditions for security are:

Condition 1 h has at least 160 bits of prime factor l [12],
Condition 2 prime factor l does not divide $p^k - 1$ for small values of k [5],
Condition 3 prime factor l is not equal to p [13], and
Condition 4 p has $(40 + \log_2(84(g-1)))$ or more bits.

The fourth condition secures a curve against Gaudry's variant. Note that the effect of automorphisms can be ignored by expanding the size of the definition field with $\log_2(84(g-1))$ more bits (refer to Theorem 1).

Because we are handling a curve with a small genus, we do not need to consider Adleman,DeMarrais,and Huang's algorithm [1] or its extension to superelliptic curves [6].

Koblitz used Jacobi sums to calculate the order of Jacobian groups of hyperelliptic curves [8]. Jacobi sums can also be used for curve $C(p, \alpha, \beta)$. For simplicity, $p \equiv 1 \pmod{\mathrm{lcm}(a, b)}$ has been assumed.

Fix the generator w of multiplicative group F_p^*. For a rational number s where $(p-1)s$ is an integer, character χ_s of F_p^* is defined by

$$\chi_s(w) = e^{2\pi i s}.$$

Then, extend χ_s to the whole F_p by setting $\chi_s(0) = 0$ when s is not an integer and setting $\chi_s(0) = 1$ when it is.

For integers $l = 1, 2, \ldots, a-1$ and $m = 1, 2, \ldots, b-1$,

$$j_p(l, m) = \sum_{1+v_1+v_2=0} \chi_{l/a}(v_1)\chi_{m/b}(v_2) \tag{1}$$

is called a Jacobi sum. Here, v_1 and v_2 run over F_p under the condition $1 + v_1 + v_2 = 0$.

Weil has demonstrated that the L function $L_p(U)$ of $C(p,\alpha,\beta)$ can be expressed by Jacobi sums [15]:

$$L_p(U) = \prod_{\substack{l=1,2,\ldots,a-1 \\ m=1,2,\ldots,b-1}} (1+\overline{\chi_{l/a}}(\alpha)\overline{\chi_{m/b}}(\beta)j_p(l,m)U),$$

where $\overline{\chi_s}$ denotes the complex conjugate of χ_s.

Generally, the order of a Jacobian group of a curve is equal to the value of the L function $L(U)$ of the curve at U=1, so the order h of the Jacobian group of $C(p,\alpha,\beta)$ is given as

$$\begin{aligned} h &= L_p(1) \\ &= \prod_{\substack{l=1,2,\ldots,a-1 \\ m=1,2,\ldots,b-1}} (1+\overline{\chi_{l/a}}(\alpha)\overline{\chi_{m/b}}(\beta)j_p(l,m)) \end{aligned} \tag{2}$$

Thus, to know the order h of the Jacobian group, it is sufficient to calculate the Jacobi sums $j_p(l,m)$. However, they cannot be calculated directly using the formula (1) for Jacobi sums, so we use the Stickelberger element to calculate them.

Let $[\lambda]$ denote the largest integer under the rational number λ, and $<\lambda>$ denote $\lambda-[\lambda]$. Take cyclotomic field $\mathbb{Q}(\zeta)$ with a primitive ab-th root ζ of 1. Let σ_t denote the Galois map $\zeta \mapsto \zeta^t$ of $\mathbb{Q}(\zeta)$. An element $\omega(a,b)$ in group ring $\mathbb{Z}[Gal(\mathbb{Q}(\zeta)|\mathbb{Q})]$ defined by

$$\omega(a,b) = \sum_t [<\frac{t}{a}> + <\frac{t}{b}>]\sigma_{-t}^{-1}, \tag{3}$$

where t runs over reduced residue classes mod ab, is called a Stickelberger element. As an ideal of $\mathbb{Q}(\zeta)$,

$$(j_p(l,m)) = P^{\omega(a,b)}, \tag{4}$$

where P denotes an prime ideal of $\mathbb{Q}(\zeta)$ lying over p [16]. By knowing the prime p and the prime ideal P in advance, Equation (4) can be used to determine $j_p(l,m)$ up to the power of $-\zeta$.

By determining the Jacobi sums using a Stickelberger element, the following algorithms can be obtained:

Algorithm 1 (To search for a secure $C(p,\alpha,\beta)$)
Input: a, b
Output: p, α, β, *and the order* h *of the Jacobian group*

1. $g \leftarrow (a-1)(b-1)/2$
2. $m \leftarrow Max(\lceil 160/g \rceil, \lceil 40+log_2(84(g-1)) \rceil)$
   $\lceil n \rceil$ *denotes the least integer over* n.

3. *Search the candidate j for value of a Jacobi sum for some prime p of m or more bits by using Algorithm 2:*

$$(p, j) \leftarrow Algorithm2(m).$$

4. *For every $k = 0, 1, \ldots, ab-1$,*

$$h_k \leftarrow \prod_{\substack{l=1,2,\ldots,a-1 \\ m=1,2,\ldots,b-1}} (1 + (-\zeta)^k j).$$

5. *Check that there is a value h_k that satisfies Conditions 1,2, and 3 for security in the set $\{h_0, h_1, \ldots, h_{ab-1}\}$. If there is not, Go to step 3. If it does, $h \leftarrow h_k$.*
6. *Let ζ_a and ζ_b denote the primitive a-th and b-th root of 1, respectively. For every $l = 0, 1, \cdots, a-1$ and $m = 0, 1, \cdots, b-1$, check if the order of the Jacobian group of $C(p, \zeta_a^l, \zeta_b^m)$ is equal to h or not; for example, check if h times the random element is equal to the unit element, or not. If the order is equal, output $p, \alpha = {\zeta_a}^l, \beta = {\zeta_b}^m$, and h. If there is no such l and m, go to step 3.*

As per step 2, p has $40 + \log_2(84(g-1))$ or more bits, so the curve obtained by Algorithm 1 is secure against Gaudry's variant.

Algorithm 2, contained in Algorithm 1, makes use of Equation (3) and (4) for the Stickelberger element to find the candidate value of the Jacobi sum.

Algorithm 2 (To find the candidate of the Jacobi sum)
Input: m
Output: p and j

1. $\omega \leftarrow \sum_t (< \frac{t}{a} > + < \frac{t}{b} >) \sigma_{-t}^{-1}$
2. *Randomly generate* $\gamma_0 = \sum_{l=0}^{(a-1)(b-1)-1} c_l \zeta^l \ (-20 \le c_l \le 20)$.
3. *For every $i = 1, 2, \ldots,$*
   $\gamma \leftarrow \gamma_0 + i$
   $p \leftarrow \mathrm{Norm}_{\mathbb{Q}(\zeta)|\mathbb{Q}}(\gamma)$
   If $p < 2^m$, then try the next i.
   If $p >> 2^m$, then go to step 2.
   '$>>$' means 'sufficiently larger than.'
   If p is not prime, then try the next i.
4. $j \leftarrow \gamma^\omega$
   Output p and j.

Example Algorithm 1 was run for $a = 3$ and $b = 5$.

1. $g \leftarrow (3-1)(5-1)/2 = 4$
2. $m \leftarrow \max(160/4, \lceil 40 + \log_2(84 \cdot 3) \rceil) = 48$
3. Run Algorithm 2 for $m = 48$.
   (a) $\omega \leftarrow \sum_t (< \frac{t}{3} > + < \frac{t}{5} >) \sigma_{-t}^{-1} = \sigma_1 + \sigma_7 + \sigma_{11} + \sigma_{13}$

(b) $\gamma_0 \leftarrow 20 - 3\zeta - 12\zeta^2 + 20\zeta^3 - 11\zeta^4 - 4\zeta^5 + 3\zeta^6 - 16\zeta^7$ was randomly generated.
For $\gamma \leftarrow \gamma_0 + 60 = 80 - 3\zeta - 12\zeta^2 + 20\zeta^3 - 11\zeta^4 - 4\zeta^5 + 3\zeta^6 - 16\zeta^7$, $p \leftarrow \mathrm{Norm}(\gamma) = 581929936583251$ is a prime of 50 bits. Now, the Jacobi sum candidate j for p is

$$\begin{aligned} j &\leftarrow \gamma\gamma^{(7)}\gamma^{(11)}\gamma^{(13)} \\ &= 18682331 + 1900434\zeta + 3903200\zeta^2 \\ &\quad + 735220\zeta^3 + 2683534\zeta^4 - 6028054\zeta^5 \\ &\quad - 1372490\zeta^6 + 3103044\zeta^7 \end{aligned}$$

4. For every $k = 0, 1, \ldots, 29$, compute $h_k \leftarrow \mathrm{Norm}(1 + (-\zeta)^k j)$:

$h_0 \leftarrow 1146786729413035548075542797106712838272574041037360478824 96$
$\ldots$
$h_6 \leftarrow 1146786991386963085872739586632658113239884227278269151228 81$
$\ldots$
$h_{29} \leftarrow 1146787464216129098443264922470075476506380949859553546524 16$

5. h_6 satisfies Condition 1,2 and 3. In fact,

$$h \leftarrow h_6 = 2511 \cdot l$$

Here,

$$l = 4567052932644217785235920297222851904579387603656985867 1$$

is a prime of 185 bits, distinct from p. Condition 2 is satisfied for $k \leq 1000$.
6. For $\alpha = 579364850535396$ and $\beta = 289406374935593$, it is verified that the order of the Jacobian group of $C(p, \alpha, \beta)$ is equal to h.

Thus, a C_{35} curve is obtained with

$$579364850535396y^3 + 289406374935593x^5 + 1 = 0$$

over the prime field GF(581929936583251) with the Jacobian group of the order

$$2511 \cdot 4567052932644217785235920297222851904579387603656985867 1,$$

which is secure against Gaudry's variant.

6 Conclusion

I have demonstrated that C_{ab} and superelliptic curves can be attacked by a modified Gaudry's variant, and I reported timing results for the attack. Miura has demonstrated that any algebraic curve with at least one rational point has a C_{ab}-type model [10]. Therefore, I determined that Gaudry's variant could be applied to virtually all algebraic curves.

However, Gaudry's variant cannot be effective in all non-elliptic algebraic curve cryptosystems. I have provided an example of a C_{35} curve that is unassailable by Gaudry's variant.

References

1. L.M.Adleman,J.DeMarrais, and M.D.Huang, "A Subexponential Algorithm for Discrete Logarithms over the Rational Subgroup of the Jacobians of Large Genus Hyperelliptic Curves over Finite Fields," ANTS-I, LNCS 877, Springer, 1994. 58, 63
2. E.Arbarello, M.Cornalba, P.A.Griffiths, and J.Harris, "Geometry of Algebraic Curves Volume I," Springer-Verlag, 1984. 59
3. S. Arita, "Algorithms for computations in Jacobian group of C_{ab} curve and their application to discrete-log-based public key cryptosystems," Conference on The Mathematics of Public Key Cryptography, Toronto, 1999. 58, 60, 61
4. D.G.Cantor, "Computing in the Jacobian of a hyperelliptic curve," Mathematics of Computation, 48(177), pp.95-101,1987. 59
5. G.Frey and H.-G.Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves," Math. Comp.,62(206),pp.865-874,1994. 63
6. S.D.Galbraith, S.Paulus, and N.P.Smart "Arithmetic on Superelliptic Curves," preprint,1999. 58, 61, 63
7. P.Gaudry, "A variant of the Adleman-DeMarris-Huang algorithm and its application to small genera," Conference on The Mathematics of Public Key Cryptography, Toronto, 1999. 58, 59, 62
8. N.Koblitz, "A very easy way to generate curves over prime fields for hyperelliptic cryptosystems," Rump-session Crypto'97, 1997. 63
9. B.A.LaMacchia and A.M.Odlyzko, "Solving large sparse linear systems over finite fields," Crypto '90, LNCS 537, Springer, 1990. 62
10. S. Miura, "Linear Codes on Affine Algebraic Curves," Transactions of IEICE, Vol. J81-A, No 10, pp.1398-1421,1998. 60, 66
11. PARI-GP, ftp://megrez.math.u-bordeaux.fr/pub/pari 62
12. J.M.Pollard, "Monte Carlo methods for index computation mod p," Math. Comp.,32(143),pp.918-924,1978. 58, 63
13. H.-G.Rück, "On the discrete logarithm in the divisor class group of curves," Math. Comp.,68(226),pp.805-806,1999. 63
14. Y.Sakai and K.Sakurai, "Design of hyperelliptic cryptosystems in small characteristic and a software implementation over F_{2^n}," Asiacrypt '98, Advances in Cryptology, LNCS 1514, Springer, 1998. 58
15. A.Weil "Numbers of solutions of equations in finite fields," Bull.Amer.Math.Soc.,55,pp.497-508,1949. 64
16. A.Weil "Jacobi Sums as "Grössencharaktere"," Trans.Amer.Math.Soc.,73,pp.487-495,1952. 64