Efficient Implementation of Schoof's Algorithm in Case of Characteristic 2

Tetsuya Izu, Jun Kogure, and Kazuhiro Yokoyama

FUJITSU LABORATORIES LTD., 4-1-1 Kamikodanaka
Nakahara-ku Kawasaki 211-8588, Japan
{izu, kogure}@flab.fujitsu.co.jp, yokoyama@sec.flab.fujitsu.co.jp

Abstract. In order to choose a secure elliptic curve for Elliptic Curve Cryptosystems, it is necessary to count the order of a randomly selected elliptic curve. Schoof's algorithm and its variants by Elkies and Atkin are known as efficient methods to count the orders of elliptic curves. Intelligent Choice System(ICS) was proposed to combine these methods efficiently. In this paper, we propose an improvement on the ICS. Further, we propose several implementation techniques in the characteristic 2 case.

1 Introduction

When we use the Elliptic Curve Cryptosystems [8,16] (*ECC* for short), we first have to define an elliptic curve over a finite field. Then, all cryptographic operations will be performed on the group of rational points on the curve and the security of ECC depends on the order of the group. (We call this order *the order of the curve* for short.) In order to avoid attacks that utilize some particular character of curves, such as MOV reduction [15] or SSSA attack [21,22,18], it is important to choose a curve randomly. Hence we need to count the order of a randomly selected curve defined over a finite field.

When the curve is defined over a large prime field, we can use the SEA (Schoof-Elkies-Atkin) algorithm [3,20,13] as the most efficient method to count the order of a curve. Thanks to Lercier [11] and Couveignes [10], we can also apply the SEA algorithm to curves over finite fields of characteristic 2. The SEA algorithm consists of two completely different stages of subprocedures and is a probabilistic algorithm. Thus, for an efficient implementation, it is important to find an efficient combination of such subprocedures. In [6] the Intelligent Choice System (*ICS* for short) was proposed to give an efficient combination of subprocedures in the first stage and was applied to counting the orders of curves over large prime fields.

In this paper, we propose an improvement on the ICS by considering not only the cost of the subprocedures in the first stage but also that of the second stage. Thus, the ICS is considered as a *practical optimization algorithm* which finds an efficient path through two different stages. Further, we introduce "two-cycles" subprocedure, by making use of the characters of the characteristic 2 fields. We also propose an implementation of [11] using Gröbner Basis.

H. Imai, Y. Zheng (Eds.): PKC 2000, LNCS 1751, pp. 210–222, 2000.

The experiments show that we can count the order of an elliptic curve over $GF(2^{239})$ in 212 seconds in average on a Pentium II 400MHz PC. Incorporating with the *early abort* strategy [12], we can generate one secure curve over $GF(2^{160})$ within 266 seconds. Thus, we can generate secure elliptic curves quickly also in the characteristic 2 field case.

In Section 2, we briefly look over the SEA algorithm and the ICS. In Section 3, we propose an improvement on the ICS and apply it to the curves defined over finite fields of characteristic 2. In Section 4, we also propose a new subprocedure(two-cycles) and the implementation of [11] using Gröbner Basis. In section 5, we give our experimental results to show the new ICS works efficiently. We also give our results on generating secure curves for ECC. We note that a part of the results (Section 4) was already presented in [7] *in Japanese*.

2 SEA Algorithm and ICS

In this section, we recall the SEA algorithm and give brief overview of the ICS [6]. Here we consider an elliptic curve E over a finite field $GF(q)$ of q elements.

2.1 Schoof's Algorithm

First we will briefly recall the Schoof's algorithm [19]. We denote the subgroup of ℓ-torsion points of E by $E[\ell]$ and the Tate module by $T_\ell(E)$ for a prime ℓ. The *Frobenius endomorphism* $\phi : (x, y) \to (x^q, y^q)$ of E is defined on $T_\ell(E)$ as a linear map and satisfies the equation: $\phi^2 - t\phi + q = 0$, where t is the *trace* of ϕ which does not depend on ℓ. Then $\#E(GF(q)) = q+1-t$. If we find an integer t_ℓ such that

$$\phi^2(P) + qP = t_\ell\phi(P) \tag{1}$$

for any $P \in E[\ell]$, we get $t \equiv t_\ell \pmod{\ell}$. By Hasse's theorem, t must satisfy $-2\sqrt{q} \le t \le 2\sqrt{q}$. Therefore if we compute $t \bmod \ell$ for various small primes until their product exceeds $4\sqrt{q}$, we can uniquely determine the cardinality of the curve by means of the Chinese Remainder Theorem.

We denote the ℓ-th *division polynomial* by f_ℓ, whose degree is $(\ell^2-1)/2$ for $\ell \ge 3$, and which vanishes precisely on the x-coordinates of the ℓ-torsion points. As we compute (1) in the ring $GF(q)[x, y]/(e(x, y), f_\ell(x))$, where $e(x, y)$ is the defining polynomial of E, the dominant steps is the computation of x^q and y^q in that ring. From this, the complexity of this algorithm will be $O(\log^8(p))$.

2.2 SEA Algorithm and Isogeny Cycles

Elkies' idea is to make use of a degree $(\ell-1)/2$ factor g_ℓ of f_ℓ when it is possible to compute g_ℓ in $GF(q)[x]$. (In this case, ℓ is called an *Elkies prime*. Otherwise ℓ is called an *Atkin prime*). The factor g_ℓ represents an eigenspace of ϕ, which can be computed as a kernel of an isogeny map. Thus, $t \bmod \ell$ is calculated by the eigenvalue of the eigenspace. (We note that the ratio of Elkies primes is

expected as 1/2.) This method will reduce the complexity to $O(\log^6(p))$. Rather than determining the unique value of $t \bmod \ell$, Atkin obtained certain restrictions on the value for Atkin prime case. Then the true value of t is found among a lot of candidates by the match-and-sort technique [10]. (See [20] for the detail.)

The SEA algorithm is obtained by combining the above two and consists of two stages, namely, (I) *collecting information stage* and (II) *trial search stage*:

Outline of the SEA algorithm

(I) Collecting informations on $t \bmod \ell$ for various ℓ's until $\prod \ell > 4\sqrt{q}$:
  (1) Compute the modular polynomial $\Phi_\ell(x, j)$.
  (2) Check if $\bar{\Phi}(x) = \Phi_\ell(x, j(E))$ has a root in $GF(q)$.
    (2-E) If $\bar{\Phi}(x)$ has a root in $GF(q)$ (we call ℓ an Elkies prime) calculate $t_\ell = t \bmod \ell$ using g_ℓ.
    (2-A) Otherwise (we call ℓ an Atkin prime) calculate possible values of $t \bmod \ell$. We denote the set of possible values of $t \bmod \ell$ by $\mathcal{T}_\ell$.

  We denote the set of Elkies primes and the set of Atkin primes by $\mathcal{E}$ and $\mathcal{A}$, respectively.

(II) Determining the value t by trial search: Now, there are a lot of candidates T for the value of t, where

$$T \bmod \ell = t_\ell \text{ for } \ell \in \mathcal{E} \text{ and } T \bmod \ell \in \mathcal{T}_\ell \text{ for } \ell \in \mathcal{A}.$$

  The value of t is (uniquely) determined by testing if $(q + 1 - T)P = \mathcal{O}$ for each candidate T, where P is a sample rational point of E and $\mathcal{O}$ is the point of infinity.

The isogeny cycles method [3] was proposed to improve the SEA algorithm, where $t \bmod \ell^2$, $t \bmod \ell^3, \ldots$ can be computed efficiently from $t \bmod \ell$ when ℓ is an Elkies prime. Incorporating the *isogeny cycles* method, the stage (I) allows information on $t \bmod \ell^k$ for some positive integer k. Thus, in the stage (I), we will gather informations of $t \bmod \ell^k$ for some primes ℓ and integers $k > 0$. We call the product of all primes or prime powers the `counter` whose information will be used in the stage (II). When `counter` exceeds $4\sqrt{q}$ in the stage (I), we enter the next stage (II).

2.3 ICS

In implementation of SEA, the following choices are very important for the total efficiency;

1. decision whether to apply the isogeny cycles method for $t \bmod \ell^k$ or not, when we find an Elkies prime ℓ,
2. decision whether to compute the candidates for the value of t or just abandon it, when we find an Atkin prime, and
3. the setting on the upper bound `CanMAX` on the number of candidates of the trace and usage of informations with respect to Atkin primes.

Thus, to give the most efficient realization (implementation), we have to find good combinations (choices) of subprocedures. In [6], this "combination problem" was dealt with and the ICS was proposed as a systematic and practical answer. It has the following functions corresponding to the problem.

The functions of the ICS [6]

(A) At each step in the stage (I), the subprocedure M with the smallest *estimated complexity* `complex`(M) is chosen among the possible subprocedures. Usually, `complex`(M) is set by the complexity of the dominant steps of M.
(B) For each Atkin prime ℓ, the decision if we use the information on $t \bmod \ell$ in the stage (II) will be made by introducing an index (*Atkin index*).
(C) By the *virtual (Atkin/isogeny cycles)* methods, we can enter the stage (II) from the stage (I) *dynamically*.

When we handle curves over large prime fields $GF(p)$, the ICS chooses one from the following at each step in the stage (I): (See Figure 1.)

(a) Schoof's original algorithm,
(b) Atkin/Elkies' method,
(c) the isogeny cycles method,
(d) the virtual method (a short cut to the stage (II)) proposed in [6].

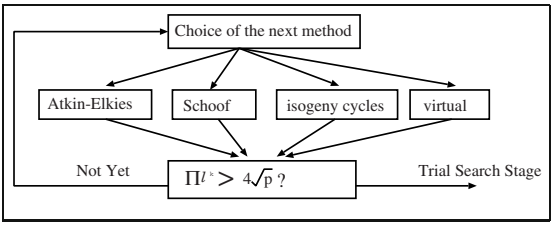

Fig. 1. The diagram of the ICS

3 The New ICS for Curves over $GF(2^n)$

In order to apply the ICS to the characteristic 2 case efficiently, we enhance the ICS as follows:

(i) We find an efficient path from the stage (I) to the stage (II).
(ii) We add a special subprocedure which makes a good use of the speciality of the characteristic 2.

We note that the enhancement (i) is also applicable to the order counting of curves over large prime fields.

3.1 Finding an Efficient Path from Stage (I) to Stage (II)

In the original ICS we set the upper-bound `CanMAX` on the total number of candidates of t which will be tested in the stage (II). However, to get the best total timings, we have to take care of the cost of the stage (II) and find an efficient path (flow of computations) from the stage (I) to the stage (II).

To find such an efficient path, we consider the virtual method as a procedure between the stage (I) and the stage (II). That is, one can decide to enter the stage (II) even if `counter` does not exceed the bound $4\sqrt{q}$. In this case, we apply the virtual method before the computation in the stage (II).

$$\boxed{\text{Stage (I)} \rightarrow \text{virtual method if necessary} \rightarrow \text{Stage (II)}}$$

At each step in the stage (I), the choice of the next subprocedure and the decision whether one enters the stage (II) or not are made at the same time by the new function `Single-Depth-Search` (`SDS` in short). Here, we introduce a subprocedure M_0 which does nothing. We call M_0 *no-operation*. Let other possible subprocedures be $M_1, \ldots, M_k$.

`Single-Depth-Search`:

(1) Compute the estimate $\texttt{ET}(M)$ of total time for each subprocedure M as follows:
   (i) For M_0, estimate the total time when one enters the stage (II) directly. (Between the stage (I) and stage (II), one executes the virtual method if necessary.)
   (ii) For other M_i, $1 \leq i \leq k$, estimate the total time when one executes M_i and then enters the stage (II). (Between the stage (I) and stage (II), one executes the virtual method if necessary.)

(2) Among $\texttt{ET}(M_0), \ldots, \texttt{ET}(M_k)$, find the smallest one, say $\texttt{ET}(M_s)$.
If $M_s = M_0$, one enters the stage (II).
Otherwise, one chooses M_s as the next procedure.

Repeating the function `SDS` at each step, we have a *practical realization* of `multiple-depth-search` which will give the most efficient flow of computations. The new ICS has the following functions: (See Figure 2.)

The functions of the new ICS with `SDS`

(N-A) At each step in the stage (I), the subprocedure with the smallest *estimated total time* is chosen among the possible subprocedures by the function `SDS`. If the output is M_0, we enter the stage (II).

(N-B) For each Atkin prime ℓ, the decision if we use the information on t mod ℓ in the stage (II) will be made by introducing an index (Atkin index).

Next, we will give the details of estimating the total time.

Estimate of the Total Time: Let M be the chosen method and N_i the expected number of the candidates in the stage (II) after the execution of M with probability p_i for each i. Then, the estimate $\texttt{ET}(M)$ of the total time is given by

$$c_M \texttt{complex}(M) + \sum_i c_{m-s,i} p_i \sqrt{N_i}, \tag{2}$$

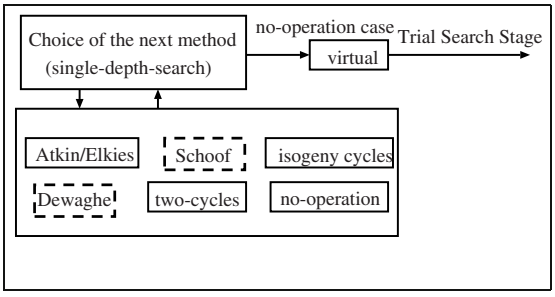

Fig. 2. The diagram of the new ICS

where $c_M, c_{m-s,i}$ are *weights* (in $\mathbf{R}$) and $\texttt{complex}(T)$ denotes the complexity of the dominant step of T over the field arithmetics, which is also used in the function (A) in the original ICS. We determine the weights $c_M, c_{m-s,i}$ by data of actual computations on a real computer. Of course, these depend on the computational environment (the CPU, the compiler, *etc.*). We note that, for the Atkin/Elkies's method for a prime ℓ, the expected information on t is $t \bmod \ell$ with $1/2$ possibility and a set $\mathcal{T}_\ell$ of candidates with $1/2$ possibility. From these, we estimate the expected number of candidates.

3.2 Additional Subprocedures of ICS for Curves over $GF(2^n)$

We can use the following special subprocedures (e),(f) in the stage (I) in the SEA algorithm, by which we can improve the effect of the ICS. (For the large prime field case, the method (f) is also applicable, but the method (e) is not, since the method (e) uses the special characters of characteristic 2 fields.)

(e) A method to compute $t \bmod 2^m$ for positive integers m,
(f) A method based on Dewaghe's idea [4] to compute $t \bmod \ell$ for an Atkin prime ℓ.

For simplicity, we call the method (e) the *two-cycles method*, and the method (f) *Dewaghe's method*. Compared with the methods (b) and (c) in the original ICS in Section 2.3, the methods (e) and (f) do not seem to have much "contribution" in the stage (I), however, they seem to work very well as "short cuts" to the stage (II). We will give details of two-cycles method in Section 4.1 and show its practical effect in Section 5.1. As our current implementation does not support Dewaghe's method, we omit its description.

Remark 1 *In the author's experiments, we computed* $t \bmod 2^k$ *for appropriate* k *at the beginning of the SEA algorithm.*

3.3 Effects of the new ICS in Characteristic 2 Case

The original ICS requires the upper-bound $\texttt{CanMAX}$ on the total number of candidates of t, on which the total efficiency depends heavily. To get an appropriate

CanMAX, we have to measure the timings on a number of examples. On the other hand, in the new ICS, the effect of CanMAX on the total efficiency becomes much smaller and we can find more efficient value of CanMAX. In the original ICS, CanMAX for curves over $GF(q)$ with 155 bit size q was set as 10^8, but it can be increased to 10^9 in the new ICS. Of course, the function SDS has actual effect in the case where the number of candidates becomes close to CanMAX.

Examples 2 *Here we demonstrate the detail of actual computation by examples on a PC with Pentium II 400MHz CPU. We select examples for which the function* SDS *works well. In the below, series of triples represent the trace of computation, where each triple consists of the selected method, the selected prime, and the actual cost (in seconds). For simplicity, we write* **a**,**e** *and* **i** *for Atkin's method, Elkies' method and the isogeny cycles method, respectively. (For the definition of* E_β*, see Section 4.1.)*

(1) E_β*, where* $\beta = \alpha^5 + \alpha^3$*, over* $GF(2^{155}) = GF(2)(\alpha)$*. It took 27.5 seconds by the ICS without* SDS*, however, it took 18.9 second by the ICS with* SDS*. We list the final steps in the computations.*

no SDS: [**e**,41,2.6],[**a**,43,2.3] → virtual method → Stage (II)
SDS: [**e**,41,2.6],[**a**,43,2.3],[**e**,47,3.1] → virtual method → Stage (II)

(2) E_β*, where* $\beta = \alpha^6 + \alpha^5 + \alpha^2 + 1$*, over* $GF(2^{196}) = GF(2)(\alpha)$*. It took 119 seconds by the ICS without* SDS*, however, it took 95.7 seconds by the ICS with* SDS*. We list the final steps in the computations below.*

no SDS: [**a**,67,11.0],[**e**,71,14.1],[**a**,73,14.6] → virtual method → Stage (II)
SDS: [**a**,67,10.9],[**e**,71,14.1],[**i**,11,6.8] → virtual method → Stage (II)

4 Implementations of Subprocedures

In order to make the ICS work efficient, subprocedures must be efficiently computable. Here we pick up subprocedures which are important and completely different from subprocedures for odd characteristic fields, and give some details on them. (Cf. [7].) From now on, let $K = GF(2^n)$.

4.1 Two-Cycles Method for Curves over $GF(2^n)$

The two-cycles method utilizes the special characters of the characteristic 2 fields. Here, we consider elliptic curves defined as follows:

$$E_a : \quad y^2 + yx = x^3 + a, \qquad a \in K \tag{3}$$

We note that each elliptic curve defined over K is K-isomorphic to some E_a or its quadratic twist.

The two-cycles method uses the fact that $E_a[2^k] \cong \mathbf{Z}/2^k\mathbf{Z}$ is the eigenspace of the Frobenius endomorphism and the 2^k-division polynomial can be computed efficiently. Let $g_0 = 0$, $g_1 = x + \sqrt[4]{a}$, and for each $i > 1$

$$g_i = g_{i-1}^2 + \sqrt[2^{i+1}]{a}\, x \prod_{j=1}^{i-2} g_j^2. \tag{4}$$

Then, for points $P = (x, y), Q = (x', y')$ on E with $Q = 2^k P$, we have

$$x' = \frac{g_k^{2^{k+1}}}{h_k^{2^k}}, \tag{5}$$

where $h_k = x \prod_{j=1}^{k-1} g_j^2$. (See [14].) Since the point of order 2 on E_a is $(0, \sqrt{a})$, the x-coordinate of each point of order 2^k is a root of $g_{k-1}(x)$. By generalizing this fact, we have the following. (We note that the x-coordinate of a point P of order 2^k is K-rational, then P is K-rational.)

Lemma 3 *Suppose that a point of order 2^k is K-rational and its x-coordinate, say α, is already computed. Then, the x-coordinate of each point of order 2^{k+s} is a root of the following polynomial of degree 2^s.*

$$g_{k,s} = g_s^2 + \sqrt[2^s]{\alpha_k}\, h_s = 0 \tag{6}$$

Each point of order 4 is K-rational and so $t \equiv 1 \pmod 4$. Then, the x-coordinate of each point of order 8 is a root of the polynomial g_2 of degree 2. Factoring g_2, we get the x-coordinate of a point of order 8, if g_2 has a rational root, or we get $t \equiv 5 \pmod 8$, otherwise. Repeating this procedure, we can compute $t \equiv 1 \pmod{2^k}$ efficiently if each point of order 2^k is K-rational.

Examples 4 Let $K = GF(2^{155}) = GF(2)(\alpha)$, where $\alpha^{155} + \alpha^{62} + 1 = 0$. On the curve $E_\alpha : y^2 + yx = x^3 + \alpha$, points of order 2^{20} are K-rational. In this case, the two-cycles method works efficiently and the order $\#E_\alpha(K)$ was computed within 7.6 seconds on a PC with Pentium II 400MHz CPU.

When we compute $t \bmod 2^{k+s}$ by using $g_{k,s}$, we can use a technique similar to the match-and-sort technique in the isogeny cycles method ([6]). Since the value $t' = t \bmod 2^{k+s-1}$ is already computed, either $t \equiv t' \pmod{2^{k+s}}$ or $t \equiv t' + 2^{k+s-1} \pmod{2^{k+s}}$ holds. Thus, finding a pair (u, v) of integers such that (i) either $ut' = v$ or $u(t' + 2^{k+s-1}) = v$ holds, and (ii) $u^2 + v^2$ is as small as possible, and testing if

$$u\phi(P) = \pm vP, \tag{7}$$

we can determine the value $t \bmod 2^{k+s}$.

Estimate of Complexity: In the new ICS, we can use the following estimate on the complexity. Suppose that we have already computed $t \bmod 2^{k+s-1}$. Then we consider that the computation of $t \bmod 2^{k+s}$, requires

$$(n + c \max\{|u|, |v|\}) \times U^2, \tag{8}$$

where $U = \deg(g_{k,s})$, n is the extension degree of K and c is a constant.

4.2 Isogeny Map Computation by Lercier's Method

Here, we will discuss on the efficiency of the isogeny map computation for curves over $GF(2^n)$. In the implementation, we tested Lercier's method [11], Couveignes's second method [2] and a simple modification of Couveignes's second method based on interpolation and we found that Lercier's method is more efficient than others, even in the case suited for Couveignes's second method, i.e. a given curve has rational points of a large order 2^k. To give a practical implementation of Lercier's method, we utilize Gröbner basis computation facility of Risa/Asir computer algebra system [17]. Then, the ratio of the computation of isogeny maps in the whole computation is quite small. For example, the ratio is around 7 % for curves over $GF(2^{155})$ and $GF(2^{196})$.

Implementation with Gröbner Basis Computation In Lercier's method [11], one computes the isogeny map by solving a certain system $\mathcal{S}$ of algebraic equations over K and those algebraic equations are derived from the commutativity of the isogeny map and the translation of the point of order 2. In a practical point of view, the key of Lercier's method is that $\mathcal{S}$ has the unique solution belonging to $GF(2)^k$, where k is the number of indeterminates in $\mathcal{S}$. Since $\mathcal{S}$ is defined over K, we can transform $\mathcal{S}$ to another system $\mathcal{S}_0$ of algebraic equations over $GF(2)$ with the same indeterminates. Then $\mathcal{S}_0$ becomes so-called an "over-determined" system. To solve such a system, the Gröbner basis computation seems very useful and efficient. (See [1] for the notion and algorithms of Gröbner basis.)

Now, suppose that $\pi_1, \ldots, \pi_k$ are indeterminates in the system $\mathcal{S}_0$ and $I(\mathcal{S}_0)$ is the ideal in $GF(2)[\pi_1, \ldots, \pi_k]$ generated by all polynomials in $\mathcal{S}_0$. Since $\mathcal{S}_0$ has the unique solution, with respect to any term order, the Gröbner basis of $I(\mathcal{S}_0)$ is $(\pi_1 - e_1, \ldots, \pi_k - e_k)$, where $e_i \in GF(2)$ and $(e_1, \ldots, e_k)$ is the solution of $\mathcal{S}_0$.

By several experiments, we observe that many of equations in $\mathcal{S}_0$ are linear. Moreover, there seems a certain randomness in the distribution of coefficients of equations, which implies that for each π_i, the ratio of equations having a term containing π_1 is close to $1/2$. From these observation and the property of the "sugar" strategy [5] in the computation of the Gröbner basis, the computation of the Gröbner basis with sugar strategy becomes close to the computation of solving of a system of sparse linear equations. Hence, the Gröbner basis can be computed quite efficiently in this case. As a result, the cost of generating algebraic equations is much larger than that of solving the system in the experiments.

Of course, the Gröbner basis computation is a general method which can handle non-linear equations. So, it can also compute "exceptional" cases, where the system $\mathcal{S}_0$ is not sparse or has a large degree, precisely.

5 Experimental Results

We have implemented the new ICS using Risa/Asir computer algebra system [17] and examined its ability and efficiency by experiments on a number of examples

on a PC with Pentium II 400 MHz CPU. We used the same settings on the complexity estimate function `complex` for Atkin/Elkies methods and the isogeny cycles method as in [6], and made a special table for the two-cycles method based on data of actual computations. Same as in [6], we did not use the original Schoof's method. For the experiment, we pre-computed modular polynomials for curves over $GF(2^n)$. The timings of the experiment and the detailed analysis on the timings certainly show the efficiency/progress of the new ICS and the implementations of subprocedures.

5.1 Order Counting

We selected four fields $GF(2^{155})$, $GF(2^{160})$, $GF(2^{196})$, $GF(2^{239})$, chose 200 curves over each field as

$$y^2 + yx = x^3 + a, \quad 2 \leq n(a) \leq 201,$$

and measured the average time needed to compute the order of one curve. (For $GF(2^{239})$, we chose 150 curves.) Here, $n(a)$ denotes the binary expression of an element a in K, that is, $n(a) = \sum a_i 2^i$ if $a = \sum a_i \alpha^i$, where $a_i \in \{0, 1\}$ and α is the primitive element. Moreover, in order to analyze the effect of the ICS, we tried several combinations of functions of the ICS and applied them for 200 curves over $GF(2^{160})$.

Table 1 shows the timings of counting the orders of curves over each field. We also put the best and the worst time in the table. To know the effect of the function `SDS`, we tested the ICS with `SDS` and that without `SDS`. The speed up (the progress) by the `SDS` is almost 3 % and the expected "practical complexity" seems around $O(n^5)$ which also shows the quality of the implementation. In smaller field, as the two-cycles method works very well, the effect of the `SDS` is rather small. Table 2 shows the detailed analysis on the effect of each function in the ICS, where `SDS` means the use of `SDS`, `re-ordering` means the use of Atkin index (see Section 2.3) and `isogeny` means the use of `complex` to choose the isogeny method. So, the number (3) corresponds to the original ICS applied to curves over $GF(2^n)$ and the number (8) corresponds to the ordinary SEA algorithm with isogeny cycles method. Table 2 suggests that the new ICS attained nearly twice faster than the ordinary SEA with isogeny cycles method. Moreover, it also suggests that our further enhancement on the ICS improved the efficiency nearly 40 % up compared with the original ICS.

5.2 Finding Secure Elliptic Curves

For secure ECC, it is strongly recommended to use a curve whose order is prime or almost prime. For this purpose, we can use *early abort* strategy [12]. In this strategy, we check if the order has a factor in each step of the computation of $t \bmod \ell$. If we find that the order is not almost prime, we can abandon the curve and try the next one. The effect of the strategy is supported by mathematical

Table 1. Timings for counting orders (seconds):

degree	SDS	average time	best time	worst time	CanMAX
155	USE	20.3	7.6	42.7	10^9
	NO	21.0	8.6	44.5	10^8
160	USE	22.2	14.2	50.3	10^9
	NO	22.8	15.5	50.2	10^8
196	USE	79.8	40.4	155.4	10^{10}
	NO	82.2	40.7	149.7	5×10^8
239	USE	212.0	107.4	578.2	5×10^{10}
	NO	216.0	113.7	548.7	10^9

Table 2. Effects of functions of the ICS (seconds):

No.	SDS	two-cycle	isogeny	re-ordering	virtual	average	best	worst
(1)	YES	YES	YES	YES	YES	22.2	14.2	50.3
(2)	NO	YES	YES	YES	YES	22.8	15.5	50.2
(3)	NO	NO	YES	YES	YES	31.0	19.1	77.5
(4)	NO	YES	NO	YES	YES	25.9	16.6	53.3
(5)	NO	YES	YES	NO	YES	27.1	15.2	88.8
(6)	NO	YES	YES	YES	NO	29.6	20.1	50.5
(7)	NO	NO	YES	YES	NO	38.1	23.3	77.5
(8)	NO	NO	NO	NO	NO	42.6	37.3	78.4

analysis [9,12]. Here, we incorporated the strategy to our implementation with the ICS and searched a number of secure curves.

For $GF(2^{160})$ and $GF(2^{239})$, we generated curves E_a randomly and tested if the order of the quadratic twist of E_a is written as $2 \times p$ for some prime number p. So, we searched "secure curves" among the quadratic twists of randomly generated curves. We generated 20 "secure" curves over each field and list the average time to find one curve in Table 3. From the timings, we are convinced that one can generate secure curves quite efficiently.

Table 3. Secure Curve Generation (seconds):

finite field	$GF(2^{160})$	$GF(2^{239})$
average time	266	3783

Remark 5 *We note that even if the order $\#E_a$ is divisible by larger power of 2, its quadratic twist is not divisible by 4. Thus, the two-cycles method never does harm in finding a secure curve.*

6 Concluding Remarks

We have introduced the new ICS and could speed up the order counting process 40 %. Comparing with the process without the ICS, the new ICS process is more than 90 % faster. We also proposed several practical implementations that can be used in the characteristic 2 case. As a result, incorporating with the early-abort strategy, we can choose secure curves from randomly selected elliptic curves in a short time. For example, our experiment shows that we can generate one secure curve over $GF(2^{160})$ within 266 seconds. We note that for order counting of curves over large prime fields our current implementation with `SDS` is about 20 % faster for curves over $GF(2^{240}+115)$ than that in [6], where we do not use fast arithmetics technique.

Our future work will be speeding up the process and calculation of curves defined over bigger fields. The important factors are (1) finding the better settings in the ICS, (2) improvements on the match-and-sort algorithm, (3) improvements on isogeny calculation, and (4) speeding up the calculation of eigenvalues.

The new ICS can handle any efficient subprocedures and it can be applied to any elliptic curves over any ground fields (e.g. OEF [23]) in the SEA algorithm.

References

1. Becker, T., Weispfenning, V., Gröbner Bases. Graduate Texts in Mathematics 141, Springer-Verlag (1993). 218
2. Couveignes, J-M., Computing ℓ-isogenies using the p-torsion, *ANT-II*, Lecture Notes in Compute Science, 1122, Springer (1996) 59–65. 218
3. Couveignes, J.-M., Dewaghe, L., Morain, F., Isogeny cycles and the Schoof-Elkies-Atkin algorithm, LIX/RR/96/03 (1996). 210, 212
4. Dewaghe, L., Remarks on the Schoof-Elkies-Atkin algorithm, *Mathematics of Computation* **67** (1998) 1247–1252. 215
5. Giovini, A., Mora, T., Niesi, G., Robbiano, L., Traverso, C., One sugar cube, please, *Proceedings of ISSAC '91*, ACM Press (1991) 49–54. 218
6. Izu, T., Kogure, J., Noro, M., Yokoyama, K., Efficient implementation of Schoof's algorithm, *ASIACRYPT'98*, Lecture Notes in Computer Science, 1514, Springer (1998) 66–79. 210, 211, 213, 217, 219, 221
7. Izu, T., Kogure, J., Noro, M., Yokoyama, K., Order counting of elliptic curves defined over finite fields of characteristic 2, *Transaction of IEICE Series A*, **82** (1999) 1253–1260. (in Japanese) 211, 216
8. Koblitz, N., Elliptic curve cryptosystems, *Mathematics of Computation* **48** (1987) 203–209. 210
9. Lenstra Jr., H.W., Factoring integers with elliptic curves, *Annals of Mathematics* **126** (1987) 649–673. 220
10. Lercier, R., Algorithmique des courbes elliptiques dansles corps finis, Doctoral Thesis, L'École Polytechnique (1997). 210, 212
11. Lercier, R., Computing isogenies in F_{2^n}, *ANTS-II*, Lecture Notes in Computer Science, 1122, Springer (1996) 197–212. 210, 211, 218
12. Lercier, R., Finding good random elliptic curves for cryptosystems defined over F_{2^n}, *EUROCRYPT '97*, Lecture Notes in Computer Science, 1233, Springer (1997) 379–392. 211, 219, 220

13. Lercier, R., Morain, F., Counting the number of points on elliptic curves over finite fields: strategy and performances, *EUROCRYPT '95*, Lecture Notes in Computer Science, 921, Springer (1995) 79–94. 210
14. Menezes, A., Elliptic curve public key cryptosystems, Kluwer Academic Publishers, Boston (1993). 217
15. Menezes, A., Okamoto, T., Vanstone, S.E., Reducing elliptic curves logarithms to logarithms in a finite field, *STOC '91*, ACM Press (1991) 80–89. 210
16. Miller, V., Uses of elliptic curves in cryptography, In *CRYPTO '85*, Lecture Notes in Computer Science, 218 (1986) 417–426. 210
17. Noro, M., Takeshima,T., Risa/Asir – a computer algebra system, *Proceedings of ISSAC '92*, ACM Press, New York (1992) 387–396. 218
18. Satoh, T., Araki, K., Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves, *Commentarii Math, Univ. Sancti Pauli* **47** (1997) 865–874. 210
19. Schoof, R., Elliptic curves over finite fields and the computation of square roots mod p, *Mathematics of Computation* **44** (1985) 483–494. 211
20. Schoof, R., Counting points on elliptic curves over finite fields, *J. Théor. Nombres Bordeaux* **7** (1995) 219–254. 210, 212
21. Semaev, I.A., Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curves in characteristic p, *Mathematics of Computation* **67** (1998) 353-356. 210
22. Smart, N.P. The discrete logarithm problem on elliptic curves of trace one, preprint (1997). (to appear in J. Cryptology) 210
23. Vailey, D.V., Paar, C., Optimal extension fields for fast arithmetic in public-key algorirthms, *CRYPTO'98*, Lecture Notes in Computer Science, 1462, Springer (1998) 472-485. 221