Fast Elliptic Curve Point Counting Using Gaussian Normal Basis

Hae Young Kim[1], Jung Youl Park[1], Jung Hee Cheon[2], Je Hong Park[1], Jae Heon Kim[3], and Sang Geun Hahn[1]

[1] Department of Mathematics
Korea Advanced Institute of Science and Technology(KAIST)
Daejon, Republic of Korea
`{hykim,jungyoul,arttex,sghahn}@mathx.kaist.ac.kr`
`http://crypt.kaist.ac.kr`

[2] International Research Center for Information Security (IRIS)
Information and Communications University (ICU)
Daejon, Republic of Korea
`jhcheon@icu.ac.kr`
`http://vega.icu.ac.kr/~jhcheon`

[3] National Security Research Institute (NSRI)
Daejon, Republic of Korea
`jaeheon@etri.re.kr`

Abstract. In this paper we present an improved algorithm for counting points on elliptic curves over finite fields. It is mainly based on Satoh-Skjernaa-Taguchi algorithm [SST01], and uses a Gaussian Normal Basis (GNB) of small type $t \leq 4$. In practice, about 42% (36% for prime N) of fields in cryptographic context (i.e., for $p = 2$ and $160 < N < 600$) have such bases. They can be lifted from $\mathbb{F}_{p^N}$ to $\mathbb{Z}_{p^N}$ in a natural way. From the specific properties of GNBs, efficient multiplication and the Frobenius substitution are available. Thus a fast norm computation algorithm is derived, which runs in $O(N^{2\mu} \log N)$ with $O(N^2)$ space, where the time complexity of multiplying two n-bit objects is $O(n^\mu)$. As a result, for all small characteristic p, we reduced the time complexity of the SST-algorithm from $O(N^{2\mu+0.5})$ to $O(N^{2\mu+\frac{1}{\mu+1}})$ and the space complexity still fits in $O(N^2)$. Our approach is expected to be applicable to the AGM since the exhibited improvement is not restricted to only [SST01].
Keywords: elliptic curve, Gaussian normal basis, order counting

1 Introduction

Elliptic curve cryptography was independently proposed by Koblitz [Kob87] and Miller [Mil87] in 1985. Because it runs with a smaller key size than an RSA-type cryptosystem, it is possible to implement a fast and compact cryptosystem. As a result a vast amount of research has been done on its secure and efficient implementations. One of the important issues on studying elliptic curve cryptosystems is to count the number of points on an elliptic curve E over a finite field $\mathbb{F}_q$ with $q = p^n$. In 1985, Schoof [Sch85] gave the first polynomial-time algorithm whose complexity is $O(\log^{3\mu+2} q)$. Later, Elkies [Elk98] and Atkin [Art92] improved this to so-called Schoof-Elkies-Atkin (SEA) algorithm with running time $O(\log^{2\mu+2} q)$ for a large characteristic. SEA-algorithm was extended to small characteristics by Couveignes [Cou96]. In 2000, Satoh [Sat00] proposed an algorithm running in $O(N^{2\mu+1})$ time and $O(N^3)$ space for the small characteristic $p \geq 5$. Fouquet, Gaudry and Harley [FGH00] extended Satoh's algorithm for the cases $p = 2, 3$. Skjernaa [Skj00] independently extended it for the case $p = 2$. In 2001, Vercauteren, Preneel and Vandewalle [VPV01] presented a modified memory-efficient version of the algorithm whose space complexity fell to $O(N^2)$. The most recent counting algorithm, suggested by Satoh, Skjernaa and Taguchi [SST01], uses the Frobenius substitution to reduce the number of arithmetic operations over a p-adic number field with full precision. This algorithm runs in $O(N^{2\mu+0.5})$ time with $O(N^2)$ memory for $p = 2$, and $O(\max\{N^{\mu+2}, N^{2\mu+0.5}\} \log N)$ time with $O(N^{2.5})$ memory for $p \geq 3$. Harley, Mestre and Gaudry [HMG01] announced a totally different algorithm, based on the AGM (arithmetic geometric mean) iteration with a fast norm algorithm, which, as far as the authors know, has not been published yet.

Our contribution is the improvement of Satoh-Skjernaa-Taguchi (SST) algorithm. The time complexity of our algorithm is $O(N^{2\mu+\frac{1}{\mu+1}})$. We focus on a finite field with the GNB of type t. For the practical reason, we restrict $t \leq 4$. In spite of such a restriction, our cases cover about 42% (36% for prime N) of the fields in a cryptographical contexts, i.e., for $p = 2$ and $160 < N < 600$. It is known that multiplication is performed efficiently in the finite field with the GNB of small types [Sil99], [VPV01]. So we lift the GNB from the finite field to the p-adic number field in a natural way to utilize the benefits of GNBs, for the SST-algorithm mainly works over a p-adic number field. Thus a fast norm computation algorithm for the p-adic number field is derived. It runs in $O((NM)^\mu \log N)$ time with $O(NM)$ space to get precision M, while that of the SST-algorithm runs in $O((NM)^\mu M^{0.5})$ time with $O(NM)$ space. Additionally, M is about $N/2$ in point counting algorithm. As a result, for all small characteristic p, we reduced the time complexity of the SST-algorithm from $O(N^{2\mu+0.5})$ to $O(N^{2\mu+\frac{1}{\mu+1}})$ and the space complexity still fits in $O(N^2)$. As to the large-scale computation with the smallest type, our algorithm takes only about 1 day and 10 hours to count the number of points on the elliptic curve defined on $\mathbb{F}_{2^{12010}}$. Since the AGM method uses multiplication and the norm computation over p-adic field, we also expect that our methods speed up the AGM algorithm.

This paper is organized as follows: First, we set up the notation and terminology at the end of section 1, then in section 2 we briefly review Satoh-Skjernaa-Taguchi [SST01] algorithm. We introduce the notion of a Gauss period and a normal basis representation in section 3, which leads us to compute multiplication and the Frobenius substitution efficiently as described in section 4. In section 5 we present an algorithm to compute the norm with fewer operations. Followed by section 6 we describe how our algorithm can be applied to point counting. We exhibit our practical results and notes for implementation in section 7. Finally this paper ends up with some comments in section 8.

Notation Throughout this paper, p is assumed to be a small prime. We put that q is a power of p and N is a positive integer. We denote the unramified extension of degree N of $\mathbb{Q}_p$ by $\mathbb{Q}_{p^N}$, and its valuation ring by $\mathbb{Z}_{p^N}$. In general, σ stands for the Frobenius substitution in $\mathrm{Gal}(\mathbb{Q}_{p^N}/\mathbb{Q}_p)$, and π is the reduction map by p from $\mathbb{Q}_{p^N}$ to $\mathbb{F}_{p^N}$. Given a positive integer M, an operation is said to be with precision M if it is done modulo p^M. For the rest of this paper, E is a non-supersingular elliptic curve over $\mathbb{F}_{p^N}$ and $j(E)$ denotes its j-invariant. We assume that $j(E) \in \mathbb{F}_{p^N} - \mathbb{F}_{p^2}$.

2 Satoh-Skjernaa-Taguchi algorithm

In this section, we briefly review the SST-algorithm. We assume that $j(E) \notin \mathbb{F}_{p^2}$. Furthermore, the case $j(E) \in \mathbb{F}_{p^2}$ can be easily handled by counting points over a tiny subfield. It is well known that for T, the trace of the Frobenius endomorphism, $\#E(\mathbb{F}_{p^N}) = p^N + 1 - T$.

The canonical lift $E^{\uparrow}$ of a non-supersingular elliptic curve E from $\mathbb{F}_{p^N}$ to $\mathbb{Z}_{p^N}$ is an elliptic curve over $\mathbb{Q}_{p^N}$ which satisfies $\pi(E^{\uparrow}) = E$ and $\mathrm{End}(E) \cong \mathrm{End}(E^{\uparrow})$. Moreover, the canonical lift is unique up to isomorphism [Deu41]. Satoh [Sat00] showed that once we obtain the lifted j-invariant $j^{\uparrow}$ and the dual of the Frobenius endomorphism (Verschiebung) of $E^{\uparrow}$, we can calculate T, the trace of the Frobenius endomorphism, from the lifted data. By Hasse's theorem, we have $|T| \leq 2\sqrt{p^N}$. Therefore, it suffices to lift all the data with precision $M = N/2 + O(1)$. The SST-algorithm [SST01] is outlined as follows.

SST-Algorithm

(1) Compute the j-invariant of the canonical lift of E modulo p^M.
(2) Calculate the square of the leading coefficient, c_1, of the homomorphism induced by the lifted p-th Verschiebung on the formal group of $E^{\uparrow}$.
(3) Find an integer T satisfying $T^2 \equiv \mathrm{Norm}_{\mathbb{Q}_{p^N}/\mathbb{Q}_p}(c_1^2) \mod p^M$ and $|T| \leq 2\sqrt{p^N}$, and determine the sign of T.

2.1 Computing the canonical lift

To compute the $j^{\uparrow}$, the p-th modular polynomial $\Phi_p(X, Y)$ plays an important role. By a result of Lubin-Serre-Tate [LST64], the canonical lift is characterized as follows: let $j \in \mathbb{F}_{p^N} - \mathbb{F}_{p^2}$, then the solution J of $\Phi_p(\sigma^{-1}(J), J) = 0$ with $J \equiv j \mod p$ is unique in $\mathbb{Z}_{p^N}$, and $J = j^{\uparrow}$. To calculate the j-invariant of the canonical lift of E, Satoh's original algorithm [Sat00] lifts all conjugates of j simultaneously, which requires $O(N^3)$ memory. Later Vercauteren *et al.* [VPV01] improved this algorithm to reduce the space complexity to $O(N^2)$ by the direct computation of $j^{\uparrow}$. However, it still takes many evaluations of the modular polynomial $\Phi_p(X, Y)$ and inversions of elements in $\mathbb{Z}_{p^N}$. For efficiency, the SST-algorithm [SST01] used the following lemma which is a slight modification of the above result of Lubin *et al.*

Lemma 1 *For $j \in \mathbb{F}_{p^N} - \mathbb{F}_{p^2}$, let $y \in \mathbb{Z}_{p^N}$ with $y \equiv j^{\uparrow} \mod p^i$ for some $i \geq 1$, and let $\eta \in \mathbb{Z}_{p^N}$ be the element with $\Phi_p(\sigma^{-1}(y), \eta) \equiv 0 \mod p^{i+1}$ and $\eta \equiv y \mod p$, then $\eta \equiv j^{\uparrow} \mod p^{i+1}$.*

From the above Lemma, we see that for given $j^{\uparrow}$ with precision $i \geq 1$, we can raise the precision one by one, by updating Φ_p for every bit. Suppose that we have y satisfying $\Phi_p(\sigma^{-1}(y), y) \equiv 0 \mod p^i$ for some $i \geq 1$. Then it suffices to find $\delta_y \in \mathbb{Z}_{p^N}$ such that $\Phi_p(\sigma^{-1}(y), y+\delta_y) \equiv 0 \mod p^{i+1}$. Since $\Phi_p(\sigma^{-1}(y), y+\delta_y) = \Phi_p(\sigma^{-1}(y), y) + \delta_y \partial_Y \Phi_p(\sigma^{-1}(y), y) + O(\delta_y^2)$, we take

$$\delta_y \equiv -\Phi_p(\sigma^{-1}(y), y)(1/\partial_Y \Phi_p(\sigma^{-1}(y), y)) \mod p^{i+1}.$$

Moreover, it is enough to obtain $(1/\partial_Y \Phi_p(\sigma^{-1}(y), y))$ with precision 1 by the condition of y. The SST-algorithm uses a more refined technique; let $W := O(M^{\frac{\mu}{\mu+1}})$[1]. After obtaining $j^{\uparrow}$ with precision W by the above method, one can raise the precision by a similar computation based on the following observation:

$$\begin{aligned}&\Phi_p(x + p^{mW+i}\Delta_X,\, y + p^{mW+i}\Delta_Y)\\ &\qquad \equiv \Phi_p(x, y) + p^{mW+i}(\partial_X \Phi_p(x, y)\Delta_X + \partial_Y \Phi_p(x, y)\Delta_Y) \mod p^{(m+1)W}\end{aligned}$$

for $i \geq 0$ and $m \geq 1$. One can easily find that all of the operations between parentheses can be done within precision W. Furthermore, the use of σ^{-1} reduced many redundant evaluation of Φ_p, while [VPV01] did not. Computing the j-invariant of the canonical lift of E takes $O\big(N^{2\mu+\frac{1}{\mu+1}}\big)$ bit operations using $O(N^2)$ memory, where W subjects to $O\big(N^{\frac{\mu}{\mu+1}}\big)$.

2.2 Computing the leading coefficient associated with *p*-th Verschiebung

We determine the kernel of the lifted p-th Verschiebung, and then compute the square of leading coefficient of the homomorphism induced by the lifted p-th Verschiebung on the formal group. It can be performed by the algorithms described in [Sat00] for $p \geq 5$, [FGH00] for $p = 2, 3$, and [VPV01] or [Skj00] for $p = 2$.

2.3 Norm computation over the *p*-adic number fields

For $p = 2$, Satoh *et al.* presented a new algorithm to compute the norm of an element in $1 + 2^2\mathbb{Z}_2$ modulo 2^M, which is suitable for point counting of elliptic curves over $\mathbb{F}_{2^N}$. It is an analytic method using $\mathrm{Norm}_{\mathbb{Q}_{2^N}/\mathbb{Q}_2}(A) =$

[1] For cryptographic application, a word size of the CPU is recommended for W.

$\exp(\mathrm{Tr}_{\mathbb{Q}_{2^N}/\mathbb{Q}_2}(\log A))$, for $A \in 1 + 2^2\mathbb{Z}_2$. They computes the norm with precision $N/2 + O(1)$ in $O(N^{2\mu+0.5})$ time and $O(N^2)$ space by developing a fast method to obtain $\mathrm{Tr}_{\mathbb{Q}_{2^N}/\mathbb{Q}_2}$.

For $p > 2$, they use Kedlaya [Ked01] together with the Paterson-Stockmeyer algorithm [PS73]. It runs in $O(\max\{N^{2+\mu}, N^{2\mu+1/2}\} \log N)$ time with $O(N^{2.5})$ memory.

3 Gauss periods and normal bases in finite fields

Let us recall that there are two most-common bases of an extension field : a normal basis (NB) and a polynomial basis (PB). When L/K be a finite Galois extension of degree N, a basis of L over K is called a normal basis if it is of the form $(\lambda\alpha)_{\lambda \in \mathrm{Gal}(L/K)}$ for some $\alpha \in L$. Any such α is called a *normal element.* A basis is called a polynomial basis if it is of the form $(\omega^i)_{0 \le i < N}$ for some $\omega \in L$. In this section, we concentrate our interest on a normal basis, especially which is generated by a Gauss period which is defined below.

Definition 1. [Men2] *Let q be a prime or prime power, and let N, t be positive integers such that $Nt + 1$ is a prime not dividing q. Let τ be any primitive t-th root of unity in $\mathbb{Z}/(Nt+1)\mathbb{Z}$. Let γ be a primitive $(Nt+1)$-th root of unity in some extension field of $\mathbb{F}_q$. A Gauss period of type (N, t) over $\mathbb{F}_q$ is defined as*

$$\alpha = \sum_{i=0}^{t-1} \gamma^{\tau^i}.$$

Let us call a NB induced by the Gauss period of type (N, t) as the *Gaussian normal basis of type t* and denote GNB of type t. It is easy to see that the Gauss period of type (N, t) belongs to $\mathbb{F}_{q^N}$. GNBs are very practical for the cryptographic application because their representations have the computational advantage that both squaring and multiplication can be done very simply. There is a simple criterion for a Gauss period to be a normal element.

Theorem 1 [Men2] *Let q, N and t be positive integers in Definition 1. Let e be the order of q modulo $Nt + 1$. Then $\gcd(Nt/e, N) = 1$ if and only if the Gauss period of type (N, t) over $\mathbb{F}_q$ generates the normal basis for $\mathbb{F}_{q^N}$ over $\mathbb{F}_q$.*

We need to know the following lemma to develop the next section.

Lemma 2 [Men2] *Let $Nt+1$ be a prime, and $\gcd(Nt/e, N) = 1$ where e denotes the order of q modulo $Nt+1$. Let τ be a primitive t-th root of unity in $\mathbb{Z}/(Nt+1)\mathbb{Z}$. Then every non-zero element k in $\mathbb{Z}/(Nt+1)\mathbb{Z}$ can be written uniquely in the form*

$$k = q^i \tau^j, \quad 0 \le i \le N-1,\ 0 \le j \le t-1.$$

It is known that a representation with respect to a GNB of type 1 can be considered as an ordinary polynomial by a suitable change of indices, and Blake *et al.*[BRS98] showed that this idea could be extended to a GNB of type 2 by using a symmetric polynomial of double length. Refer [Sil99], [BRS98] and [BSS00] for more details about GNB of type 1 and 2 over finite fields. Moreover, their idea is extendable to GNBs of all types after a slight modification. We will deal with this extension over p-adic number fields in details in Section 4.

There is a famous conjecture of Artin that for each square-free integer $g \neq -1$, there exist infinitely many primes which have g as a primitive root. Hooley proved that this conjecture is true under the Extended Riemann Hypothesis [Hoo67], [Mur88]. Therefore, assuming the Extended Riemann Hypothesis or Artin's conjecture, it is expected that there are infinitely many finite fields with GNBs of type $t \leq 2$.

Remark 1. Note that for N prime, the type t has to be even. Furthermore, there are 26 values of prime N between 160 and 600 which there is an GNB of type 2 or 4 of $\mathbb{F}_{2^N}$ over $\mathbb{F}_2$. It covers 36% primes between 160 and 600 (For type 2, N= 173, 179, 191, 233, 239, 251, 281, 293, 359, 419, 431, 443, 491, 509, 593, For type 4, N= 163, 193, 199, 277, 307, 373, 409, 433, 487, 499, 577.)

4 p-adic lift of Gauss periods over finite fields

In this section, we consider p-adic fields taking advantage of both polynomial and normal basis representations. In this family of fields, both of multiplication and the Frobenius substitution can be done efficiently.

Theorem 2 *Let (K, v) be a complete discrete valuation field, and let L/K be a finite unramified extension of degree N. Let R_L (resp. R_K) denote the valuation ring of L (resp. K) and let $p \in K$ be a prime element of K which also prime in L. We also denote the residue class field of K and L by k_K and k_L, respectively. If $\mathcal{B}$ is a k_K-basis of k_L, then for any lift $\tilde{\mathcal{B}}$ of $\mathcal{B}$ in R_L, $\tilde{\mathcal{B}}$ is a K-basis of L. Furthermore, $\tilde{\mathcal{B}}$ is a R_K-basis of R_L.*

Proof. Let $\mathcal{B} = \{b_1, b_2, \ldots, b_N\}$ and let $\tilde{\mathcal{B}} = \{r_1, r_2, \ldots, r_N\}$. Denote π be the reduction modulo p map. If we have a non-trivial K-linear relation $c_1r_1 + c_2r_2 + \cdots + c_Nr_N = 0$, then without loss of generality, we can assume $c_i \in R_K$ and for at least one i , $\pi(c_i) \neq 0$. So we obtain $\pi(c_1)b_1 + \pi(c_2)b_2 + \cdots + \pi(c_N)b_N = 0$ in k_L, which is a contradiction. By comparing dimension, we see that $\tilde{\mathcal{B}}$ is a K-basis of L. For the second statement, it suffices to show that R_L is represented by R_K-linear sum of $\tilde{\mathcal{B}}$. Suppose that we have an element c in R_L which is not represented by R_K-linear sum of $\tilde{\mathcal{B}}$. Let $c := c_1r_1 + c_2r_2 + \cdots + c_Nr_N$, with $c_i \in K - R_K$ for some i. Put $z = \min_j(v(c_j))$; then $z < 0 \leq v(c)$ and $\pi(cp^{-z}) = 0$. By multiplying p^{-z} to c, we can write $\pi(c_1p^{-z})b_1 + \pi(c_2p^{-z})b_2 + \cdots + \pi(c_Np^{-z})b_N = 0$ in k_L which is a contradiction.

Corollary 1 *Let q be a prime or prime power, and let N, t be positive integers such that $Nt+1$ is a prime not dividing q. Let γ be a primitive $(Nt+1)$-th root of unity in some extension field of $\mathbb{Q}_q$. If $\gcd(Nt/e, N) = 1$ where e denotes* *the order of q modulo $Nt+1$, then for any primitive t-th root of unity τ in $\mathbb{Z}/(Nt+1)\mathbb{Z}$,*

$$\alpha := \sum_{i=0}^{t-1} \gamma^{\tau^i}$$

generates a normal basis over $\mathbb{Q}_q$. Furthermore, $[\mathbb{Q}_q(\alpha) : \mathbb{Q}_q] = N$.

Proof. From $\gcd(Nt+1, q) = 1$, it follows that $\pi(X^{Nt+1} - 1)$ is square free polynomial in $\mathbb{F}_q[X]$. Then $\mathbb{Q}_q(\gamma)$ is the unramified extension of $\mathbb{Q}_q$ [[Lan94], Prop. II.4.7], and so is $\mathbb{Q}_q(\alpha)$. Therefore, $[\mathbb{Q}_q(\alpha) : \mathbb{Q}_q] = [\mathbb{F}_q(\pi(\alpha)) : \mathbb{F}_q]$. Clearly, $\pi(\alpha)$ is a Gauss period of type (N, t) over $\mathbb{F}_q$. Thus $[\mathbb{F}_q(\pi(\alpha)) : \mathbb{F}_q] = N$ from the Theorem 1. By the Theorem 2, the desired conclusion can be shown.

From Corollary 1, the Gauss period can be lifted from the finite field $\mathbb{F}_{q^N}$ to $\mathbb{Q}_{q^N}$ and so we will use the family of fields that have the residue class field with the Gauss period. Let us extend the notion of a GNB from a finite field to a p-adic number field naturally, still denoting it by a GNB over a p-adic field. Note that γ defined Corollary 1 satisfies $\sigma(\gamma) = \gamma^q$ for the Frobenius substitution $\sigma \in \mathrm{Gal}(\mathbb{Q}_q(\gamma)/\mathbb{Q}_q)$ because $\gamma^{Nt+1} = 1$ and $q^{Nt} \equiv 1 \mod Nt+1$.

For convenience, we consider only the case of $p = q$. Furthermore, the following arguments hold for any other q where $q = p^l$ for some l. In the remainder of this section, we will describe how elements in $\mathbb{Z}_{p^N}$ represented with respect to the GNB of type t are expanded to elements in $\mathbb{Z}_{p^{Nt}}$ with respect to the PB, and how we can easily multiply two elements together and get the Frobenius substitution $\sigma \in \mathrm{Gal}(\mathbb{Q}_{p^N}/\mathbb{Q}_p)$.

4.1 *p*-adic number fields with GNBs of type 1

We assume the condition in Corollary 1 with $t = 1$. In this case, $\alpha \in \mathbb{Q}_{p^N}$ is equal to γ and a normal element. Furthermore, $\pi(\alpha)$ is a normal element generating the GNB of type 1 over the field $\mathbb{F}_p$.

To get a type 1 GNB over p-adic number fields, consider the minimal polynomial $F(X) = X^N + X^{N-1} + \cdots + X + 1 \in \mathbb{Z}_p[X]$ of α. Since $\sigma(\alpha) = \alpha^p$ and p is primitive in $\mathbb{Z}/(N+1)\mathbb{Z}$, we have the NB,

$$\tilde{\mathcal{B}} = \{\alpha, \sigma(\alpha), \ldots, \sigma^{N-1}(\alpha)\} = \{\alpha, \alpha^p, \ldots, \alpha^{p^{N-1}}\} = \{\alpha, \alpha^2, \ldots, \alpha^N\}.$$

Similar to extension fields with GNBs of type 1 over finite fields [Sil99], multiplication can be handled through polynomial arithmetic. Moreover, the Frobenius substitution can be done by applying simple permutation.

Multiplication For multiplication in $\mathbb{Z}_{p^N} := \mathbb{Z}_p[X]/\langle F(X)\rangle$, we will use the ring $R' = \mathbb{Z}_p[X]/\langle X^{N+1} - 1\rangle$. A lift of elements in $\mathbb{Z}_p[X]/\langle F(X)\rangle$ into R' is given as follows;

$$\sum_{i=0}^{N-1} a_i X^i \mapsto \sum_{i=0}^{N-1} a_i X^i + 0X^N.$$

Conversely, a projection from R' to $\mathbb{Z}_p[X]/\langle F(X)\rangle$ is given by the reduction modulo F. It implies that for arbitrary elements A, B in $\mathbb{Z}_p[X]/\langle F(X)\rangle$,

$$A \cdot B \equiv (A \cdot B \mod X^{N+1} - 1) \mod F.$$

Specifically, multiply A by B modulo $X^{N+1} - 1$, and then take the remainder modulo F to obtain $A \cdot B \in \mathbb{Z}_p[X]/\langle F(X)\rangle$. This multiplication is just that of two polynomials with degrees less than or equal to N; hence the complexity is clearly $O((NM)^\mu)$ to get precision M.

Frobenius substitution Let $A(X) = a_0 + a_1 X + \cdots + a_N X^N$ be an element of R'. By substituting $X = \alpha$, we obtain the Frobenius substitution of $A(\alpha)$ by

$$\sigma(\sum_{i=0}^{N} a_i \alpha^i) = \sum_{i=0}^{N} a_i \alpha^{pi} = a_0 + \sum_{j=1}^{N} a_{j/p} \alpha^j, \quad \text{where } j/p \in (\mathbb{Z}/(N+1)\mathbb{Z})^*.$$

Similarly, for all $k \in \mathbb{Z}$

$$\sigma^k(\sum_{i=0}^{N} a_i \alpha^i) = \sum_{i=0}^{N} a_i \alpha^{p^k i} = a_0 + \sum_{j=1}^{N} a_{j/p^k} \alpha^j, \quad \text{where } j/p^k \in (\mathbb{Z}/(N+1)\mathbb{Z})^*.$$

So we can compute $\sigma^k(A)$ by simple permutation on the set $(\mathbb{Z}/(N+1)\mathbb{Z})^*$, which needs $O(N)$ bit operations in a naive implementation and $O(1)$ bit operations with some elaborate implementation.

4.2 *p*-adic number fields with GNBs of type 2

We assume the condition in Corollary 1 with $t = 2$. Then $\alpha(= \gamma + \gamma^{-1})$ is the Gauss period of type $(N, 2)$ in $\mathbb{Z}_{p^N}$, and normal in $\mathbb{Q}_{p^N}$. Since $\sigma'(\gamma) = \gamma^p$ for the Frobenius substitution σ' in $\mathrm{Gal}(\mathbb{Q}_p(\gamma)/\mathbb{Q}_p)$ and $\sigma'|_{\mathbb{Q}(\alpha)} = \sigma$, we obtain the normal basis

$$\tilde{\mathcal{B}} = \{\alpha, \sigma(\alpha), \ldots, \sigma^{N-1}(\alpha)\} = \{\gamma + \gamma^{-1}, \gamma^p + \gamma^{-p}, \ldots, \gamma^{p^{N-1}} + \gamma^{-p^{N-1}}\}.$$

For every $0 \le i \le N-1$, exactly one element in the pair $(p^i, -p^i)$ can be written as $j \mod 2N+1$, for some $1 \le j \le N$ (see Lemma 2). Thus we have

$$\tilde{\mathcal{B}} = \{\gamma + \gamma^{-1}, \gamma^2 + \gamma^{-2}, \ldots, \gamma^N + \gamma^N\} = \{\gamma + \gamma^{2N}, \gamma^2 + \gamma^{2N-1}, \ldots, \gamma^N + \gamma^N\}.$$

From the last equality, the PB representation is naturally induced.

Multiplication First, we consider a representation of an element in $\mathbb{Z}_{p^N} := \mathbb{Z}_p[X]/\langle F(X)\rangle$, where $F(X)$ is a minimal polynomial of α. We denote $\|x\| := \min\{|y| \mid y \equiv x \mod 2N+1\}$, where x, $y \in \mathbb{Z}$. Then we can define a bijection $f : \{0, 1, \ldots, N-1\} \to \{1, 2, \ldots, N\}$ by $f(i) = \|p^i\|$.

For an element $A = \sum_{i=0}^{N-1} a_i \sigma^i(\alpha)$ in $\mathbb{Z}_{p^N}$, we can rewrite A with respect to γ as follows;

$$\begin{aligned} A &= \sum_{i=0}^{N-1} a_i(\gamma^{p^i} + \gamma^{-p^i}) = \sum_{j=1}^{N} a_{f^{-1}(j)}(\gamma^j + \gamma^{-j}) \\ &= \sum_{j=1}^{N} a_{f^{-1}(j)}\gamma^j + \sum_{j=1}^{N} a_{f^{-1}(j)}\gamma^{2N+1-j}. \end{aligned}$$

By replacing γ by X, we obtain the polynomial

$$A(X) = \sum_{j=1}^{N} a_{f^{-1}(j)}X^j + \sum_{j=1}^{N} a_{f^{-1}(j)}X^{2N+1-j}.$$

Thus, an element in $\mathbb{Z}_{p^N}$ can be uniquely represented as a polynomial in the ring $\mathbb{Z}_p[X]/\langle X^{2N+1} - 1\rangle$ uniquely determined modulo $X^{2N} + X^{2N-1} + \cdots + 1$. Note that this polynomial has a special property, $A(1/X)X^{2N+1} = A(X)$.

Definition 2. *For a given ring R, we call a polynomial $A(X) \in R[X]$ semi-palindromic if $A(X)$ is of the form*

$$A(X) = a_0 + \sum_{i=1}^{N} a_i(X^i + X^{2N+1-i}), \quad \textit{where } a_i \in R \textit{ for } 0 \le i \le N.$$

A semi-palindromic polynomial with $a_0 = 0$ is called palindromic.

Let S be a set of all semi-palindromic polynomials over $\mathbb{Z}_p$, then it is actually the set of all polynomials modulo $X^{2N+1} - 1$ representing elements in $\mathbb{Z}_{p^N}$. The addition is defined as the ordinary polynomial addition of elements in S, and the product of two polynomials $A(X)$, $B(X) \in S$ is the unique polynomial $C(X) \in S$ such that

$$C(X) \equiv A(X) \cdot B(X) \mod X^{2N+1} - 1. \tag{4.1}$$

Equation (4.1) yields that multiplication can be implemented using the standard polynomial multiplication with modular reduction. Indeed, if we substitute $X = \gamma$, then we see that $C(\gamma) \in \mathbb{Z}_{p^N}$ and so $C(X) \in S$.

Let us consider more efficient method to multiply two elements. For an element $A(X) \in S$, we can easily eliminate the constant term using the equality

$$A(\gamma) = \sum_{i=0}^{2N} a_i\gamma^i = \sum_{i=1}^{2N} (a_i - a_0)\gamma^i.$$

Thus, it is enough to consider multiplication of two palindromic polynomials in S. Given two palindromic polynomials $A(X)$ and $B(X)$, we can write them as

$$A(X) = A_1(X)X + A_2(X)X^{N+1} \text{ and } B(X) = B_1(X)X + B_2(X)X^{N+1},$$

where both $\deg(A_i)$ and $\deg(B_i)$ are less than N. Additionally, A_1 is of the symmetric form of A_2, that is $X^{N-1}A_1(1/X) = A_2(X)$ and the same holds for B_1 and B_2. We can easily show that a similar relation holds for pairs $(A_1B_1,\, A_2B_2)$ and $(A_1B_2,\, A_2B_1)$ (i.e. $X^{2N-2}A_1(1/X)B_1(1/X) = A_2(X)B_2(X)$). Since $A(X)\cdot B(X)$ is given by

$$A(X)\cdot B(X) \equiv A_1B_1X^2 + (A_1B_2 + A_2B_1)X^{N+2} + A_2B_2X \mod X^{2N+1}-1,$$

the multiplication in $\mathbb{Z}_{p^N}$ with respect to palindromic representation can be done by two multiplications, A_1B_1 and A_1B_2, of polynomials of degree less than N. Hence the complexity is $O(2(NM)^\mu)$. Note that it is two times slower than the case of $t = 1$.

Frobenius substitution Let $A(X) = a_0 + \sum_{i=1}^N a_iX^i + \sum_{i=1}^N a_iX^{2N+1-i}$ be an element in S. When we substitute $X = \gamma$, we obtain $A(\gamma) = a_0 + \sum_{i=1}^N a_i(\gamma^i + \gamma^{-i})$ and so the k-th Frobenius substitution σ^k of $A(\gamma)$ by

$$\sigma^k(A(\gamma)) = a_0 + \sum_{i=1}^N a_i(\gamma^{p^k i} + \gamma^{-p^k i}) = a_0 + \sum_{j=1}^N a_{\|j/p^k\|}(\gamma^j + \gamma^{-j}) \quad \text{for all } k \in \mathbb{Z}.$$

Thus we can get the k-th Frobenius substitution by simple permutation on the set $(\mathbb{Z}/(N+1)\mathbb{Z})^*$.

4.3 p-adic number fields with GNBs of type $t > 2$

We generalize the p-adic lift of GNBs of type 1 and 2. We assume that $q\,(=p)$, N and t satisfies the condition of Corollary 1.

When t is even, for an element A in $\mathbb{Z}_{p^N}$, we can express A with respect to γ by

$$\begin{aligned} A &= \sum_{i=0}^{N-1} a_i\sigma^i(\alpha) = \sum_{i=0}^{N-1} a_i(\sum_{j=0}^{t-1}\gamma^{p^i\tau^j}) \\ &= \sum_{i=0}^{N-1} a_i(\sum_{j=0}^{t/2-1}\gamma^{p^i\tau^j} + \gamma^{-p^i\tau^j}) = \sum_{i=1}^{Nt/2} c_i\gamma^i + \sum_{i=1}^{Nt/2} c_i\gamma^{tN+1-i}, \end{aligned}$$

where $\{c_i \,|\, 1 \le i \le N\}$ is a bijective image of $\{a_j \,|\, 0 \le j \le N-1\}$. If we replace γ by X, then elements of $\mathbb{Z}_{p^N}$ can be represented as palindromic polynomials in the polynomial ring modulo $X^{Nt+1}-1$. To get multiplication with this representation, we need two multiplications of polynomials of degree less than or equal to $(Nt/2)-1$ as done in the case of type 2. Therefore, the complexity is $O(2(tNM/2)^\mu)$. The Frobenius substitution can be done by a suitable permutation in the same manner as the case of $t = 2$.

When t is odd, an element A in $\mathbb{Z}_{p^N}$ can be represented by the polynomial modulo $X^{Nt+1}-1$. The multiplication in this representation can be done by multiplication of polynomials with degree less than or equal to Nt. Therefore the time complexity of the multiplication is $O((tNM)^{\mu})$. The Frobenius substitution can be done in the same manner as the case of $t = 1$.

Remark 2. If t is even, then multiplication with a GNB of type t is slower than that of type 1 by a constant factor of $2(t/2)^{\mu}$. Similarly, if t is odd, then it is slower by a constant factor of t^{μ}. Thus for $t \geq 5$, it is slower by at least 10 times. With this practical reason, hereafter we restrict the choice of t so that $t \leq 4$.

5 Norm Computation Algorithm

In this section, we develop an algorithm to compute $\mathrm{Norm}_{\mathbb{Q}_{p^N}/\mathbb{Q}_p}(A) \mod p^M$ for $A \in \mathbb{Z}_{p^N}$, where $\mathbb{F}_{p^N}$ has the GNB of type $t \leq 4$. We will use the representation for elements in $\mathbb{Z}_{p^N}$ as described in previous section. By using the 2-adic expansion of N, our algorithm requires fewer multiplications and more Frobenius substitutions. Let $N = \sum_{i=0}^{l} n_i 2^i$ with $n_i \in \{0, 1\}$ and $n_l = 1$. Denote it by $[n_0, n_1, \ldots, n_l]_2$. Since $\mathrm{Gal}(\mathbb{Q}_{p^N}/\mathbb{Q}_p)$ is generated by σ, we obtain that

$$\mathrm{Norm}_{\mathbb{Q}_{p^N}/\mathbb{Q}_p}(A) = A(\sigma A)\cdots(\sigma^{N-1}A) = M_{l-1} \cdot \prod_{i=0}^{l-2} (\sigma^{N-[n_0,n_1,\ldots,n_i]_2} M_i)^{n_i},$$

where $M_i = (\sigma^{2^{i-1}} M_{i-1}) M_{i-1},\ M_0 = A$. The following norm computation algorithm is derived from the above expression.

Algorithm ComputeNorm

```
Input : $A \in \mathbb{Z}_{p^N},\ N = [n_0, n_1, \ldots, n_l]_2, n_l = 1$
Output : $\mathrm{Norm}_{\mathbb{Q}_{p^N}/\mathbb{Q}_p}(A)$.
Begin
    1. $M \leftarrow A$;
    2. If $n_0 = 1$ then Temp $\leftarrow \sigma^{N-1} A$;
       Else Temp $\leftarrow 1$;
    3. For $i = 1$ to $l - 1$ do
       (a) $M \leftarrow (\sigma^{2^{i-1}} M) M$;
       (b) If $n_i = 1$ then Temp $\leftarrow$ (Temp)$\cdot(\sigma^{n_{i+1}2^{i+1}+\cdots+n_l 2^l} M)$;
    4. $M \leftarrow (\sigma^{2^{l-1}} M)(M)$;
    5. $M \leftarrow M\cdot$Temp;
    6. Return $M$;
End
```

ComputeNorm requires at most $2\lfloor \log_2 N \rfloor$ times multiplications over $\mathbb{Z}_{p^N}$ and at most $2\lfloor \log_2 N \rfloor$ times σ^i substitutions. Since the Frobenius substitution requires at most $O(N)$ bit operations for the field with a GNB of type $t \leq 4$ as described in section 4, it requires $O((\log N)(NM)^{\mu})$ time and $O(NM)$ space to get precision M.

Example 1. If $N = 10 = 2 + 2^3$, then

$$\mathrm{Norm}_{K/\mathbb{Q}_p}(A) = A(\sigma A)\cdots(\sigma^9 A) = (A(\sigma A)\cdots(\sigma^{2^3-1}A))(\sigma^{2^3}(A(\sigma A)))$$

1. $M_1 \leftarrow (\sigma A)A$
   (a) Temp$\leftarrow \sigma^{2^3} M_1$
2. $M_2 \leftarrow (\sigma^2 M_1)M_1$
3. $M_3 \leftarrow (\sigma^{2^2} M_2)M_2$
4. $\mathrm{Norm}_{K/\mathbb{Q}_p}(A) = M_3 \cdot$Temp

6 Application to Point counting

Now we describe how our algorithm can be applied to point counting based on SST. Before explaining the application, first we consider two basic operations: multiplication and the Frobenius substitution. Since the SST-algorithm uses a polynomial basis generated by ψ satisfying $\psi^{p^N-1} = 1$, so in general, the reduction polynomial $f(X)$ is dense. For the given polynomial $f(X)$ of degree N, $A(X) \mod f(X)$ is given by $A - (((A/X^N)Z)/X^{N-2})f(X)$ for $\deg A \leq 2N-2$ where Z is precomputed as $Z := X^{2N-2}/f$. Hence the multiplication in $\mathbb{Z}_p[X]/\langle f(X)\rangle$ is about three times slower than in $\mathbb{Z}_p/\langle X^{N+1}-1\rangle$. In the case of a type 1 GNB, our reduction polynomial is exactly $X^{N+1}-1$, so our multiplication is three times faster than that of the SST-algorithm. In the case of type 2 GNB, it is 1.5 times faster than that of the SST-algorithm for the similar reason. With type 3 or 4, our multiplication may be slower than that of the SST-algorithm, since the polynomial representation is lengthy. For the Frobenius substitution, our method requires almost nothing, while the SST-algorithm requires $p-1$ multiplications and $p-1$ additions over $\mathbb{Z}_{p^N}$ (see [SST01]).

We will show that our algorithm improves the complexity of the SST-algorithm to $O(N^{2\mu+\frac{1}{\mu+1}})$ in time and $O(N^2)$ in space, while the SST-algorithm runs in $O(N^{2\mu+0.5})$ time and at a minimum of $O(N^2)$ space. Recall that the SST-algorithm works with precision $M := N/2 + O(1)$. It was previously proved in [SST01] that it takes $O(N^{2\mu+\frac{1}{\mu+1}})$ time and $O(N^2)$ space in step (1) and (2) of the algorithm in Section 2. In step (3), applying algorithm 1 in section 5 to the norm computation, the time complexity dropped from $O(N^{2\mu+0.5})$ to $O(N^{2\mu}\log N)$, while the space complexity remains fixed to $O(N^2)$ for all small p. Hence the total complexities in time and space can be obtained.

For a detailed description, since all Frobenius substitution requires almost nothing, there is at least a 10% speed-up in Step (1) (See [SST01] for details). Moreover, as our multiplication is much faster in the case of $t = 1$ or 2, the total running time is roughly reduced by a constant factor of 3 for type 1, and by 1.5 for type 2 at least.

7 Implementation and Results

In this section, we show experimental running time of our version of the SST-algorithm for $p = 2$. For comparison, we also present the recent results of the

SST-algorithm in [SST01], which is from [SST01]. Both algorithms have been implemented in the C programming language for the most part, and some assembly for most basic operations on multi-precision integers. Satoh *et al.* obtained their result on a 32bit Pentium III-866 MHz processor, while ours was on a Pentium III-800MHz processor with a 128MB RAM of the main memory, running Linux Mandrake 2.2.17 and compiled using gcc compiler version 2.95.3 with options optimized to Pentium III processors including '-O3'. Since two platforms are different, an exact comparison between the two running results can be ambiguous. Therefore one has to regard this as a reference. Before providing our actual results, we will briefly comment on the implementation of our algorithm.

First, for efficiency we used a constant value of 32 for $162 \leq N \leq 302$, a word size of a Pentium III processor, for W in the algorithm described in Section 2.1, hence in many steps operations are performed within one-word precision. It allows us to eliminate much of the loop overhead by using an unrolled version of operations. All elements of $\mathbb{Z}_{2^N}$ are represented as polynomials as in Section 4. For GNBs of even types we used a palindromicity to store only half of the polynomial, while elements of $\mathbb{F}_{2^N}$ are always represented as full-size polynomials. Multiplication of two elements in $\mathbb{Z}_{2^N}$ is implemented using Karatsuba's method. We use naive multiplication, so called pencil-and-paper method, for the coefficients.

In the Table 1, we present the running time of both algorithms for finite fields $\mathbb{F}_{2^N}$ where N is between 160 and 600. For our results, we used finite fields with GNBs of type 1, 2, 3 or 4. It shows that our improvement largely enhances the speed as that of Satoh *et al.* in the case of type 1 and 2. We also present the result of AGM method for a rough comparison.

For a researching interest, we also show out results for large N for GNBs of type 1 and 2 in the Table 2, with varing W. These results are obtained on the same machine environment, but the compiler gcc version 3.0.3 is used instead of ver 2.95.3.

8 Conclusion

In this paper, we reduced the time complexity of the original the SST-algorithm from $O(N^{2\mu+0.5})$ to $O(N^{2\mu+\frac{1}{\mu+1}})$ with some restrictions on N, while the space complexity still remains fixed to $O(N^2)$ for any small p. We also developed a fast algorithm for the norm computation with $O((NM)^{\mu} \log N)$ time and $O(NM)$ space to get precision M. In addition, our algorithm refined the running time by a maximum constant factor of 3. In a cryptographic context (i.e., for $p = 2$ and $160 < N < 600$), about 42% (36% for prime N) of fields have such bases. Because of the reduced complexity, our method works well for a large N. As shown in Section 7, it takes merely 17.38 minutes to count the number of points on an arbitrary elliptic curve defined on the finite field $\mathbb{F}_{2^{3010}}$, and about 1 day and 10 hours on $\mathbb{F}_{2^{12010}}$ by Pentium III-800 MHz computer.

Furthermore, our improvement is not only restricted to the SST-algorithm. It can also work with all algorithms working on p-adic number fields, which

Table 1. Timings(in sec) for computations of j-invariant, Norm and Order counting. The time table of AGM method through Alpha 750 MHz is published on the homepage of Argo Tech(http://argote.ch).

N	Type	j-inv	Norm	Total	Note	N	Type	j-inv	Norm	Total	Note
162	1	0.076	0.018	**0.110**		233	-	1.72	0.29	2.24	[SST01]
163	-	-	-	0.07	AGM	**233**	2	0.433	0.142	**0.743**	
163	-	0.58	0.10	0.76	[SST01]	235	4	1.318	0.481	2.513	
163	4	0.390	0.124	0.766		236	3	1.264	0.520	2.365	
166	3	0.350	0.154	0.691		239	-	1.86	0.45	2.54	[SST01]
173	2	0.192	0.070	**0.336**		**239**	2	0.432	0.168	**0.771**	
193	-	0.98	0.19	1.31	[SST01]	239	-	-	-	0.24	AGM
193	4	0.680	0.206	1.281		244	3	1.354	0.560	2.539	
194	2	0.228	0.068	**0.383**		265	4	1.672	0.505	3.066	
196	1	0.121	0.035	**0.201**		**268**	1	0.284	0.083	**0.474**	
197	-	-	-	0.14	AGM	279	4	1.807	0.662	3.453	
197	-	1.04	0.20	1.38	[SST01]	283	-	2.97	0.73	4.13	[SST01]
199	4	0.749	0.270	1.445		286	3	1.767	0.829	3.442	
204	3	0.754	0.265	1.328		**292**	1	0.306	0.096	**0.523**	
209	2	0.325	0.083	**0.511**		**293**	2	0.598	0.210	**1.059**	
210	1	0.168	0.042	**0.262**		307	4	2.548	0.991	4.893	

Table 2. Timings for computations of Norm and Order counting for large N.

N	Type	Norm	Total	W
3010	1	2.63 min	17.38 min	96
3005	2	7.93 min	41.03 min	96
6010	1	34.33 min	2 hr 59.25 min	128
6005	2	1hr 31.68 min	7 hr 7.25 min	128
12010	1	6 hr 45 min	1 day 10 hr 24 min	192

multiplications and the Frobenius substitutions play dominant roles. It is known that the AGM method uses norm computation over a p-adic number field; hence we expect that our norm computation algorithm can be combined with the AGM method to give faster point counting.

Acknowledgement

The authors of this paper would like to thank Takakazu Satoh for his many insightful comments and interesting discussions on this work.

References

[Art92] A. O. L. Atkin, The number of points on an elliptic curve modulo a prime, Series of e-mails to the NUMBERTHRY mailing list, 1992.

[BRS98] L. F. Blake, R. M. Roth, and G. Seroussi, Efficient Arithmetic in GF(2^n) through Palindromic Representation, Tech. Rep. **HPL-98-134**, Hewlett Packard, 1998.

[BSS00] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge Univ. Press, 2000.

[Cou96] J. M. Couveignes, Computing l-isogenies using the p-Torsion, *Algorithmic number theory - ANTS-II*, LNCS **1122**, pp. 59–66, Springer-Verlag, 1996.

[Deu41] M. Deuring, Die Typen der Multiplikatorenringe elliptischer Funktionenkörper. *Abh. Math. Sem. Univ. Hamburg*, **14**, pp. 197–272, 1941.

[Elk98] N. D. Elkies, Elliptic and modular curves over finite fields and related computational issues, In D.A. Buell and eds. J.T. Teitelbaum, editors, *Computational perspective on number theory*, AMS/IP Stud. Adv. Math., **7**, pp. 21–78, Province, RI: AMS, 1998. Proceedings of a Conference in Honor of A.O.L. Atkin.

[FGH00] M. Fouquet, P.Gaudry, and R. Harley, On Satoh's algorithm and its implementation, *J. Ramanujan Math. Soc.*, **15**, pp. 281–318, 2000.

[HMG01] R. Harley, Counting points with the arithmetic-geometric mean(joint work with J. F. Mestre and P. Gaudry), Eurocrypt 2001, Rump session, 2001.

[Hoo67] C. Hooley, On Artin's conjecture, *J. Reine Angew Math.*, **225**, pp. 209–220, 1967.

[Ked01] K. Kedlaya, Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology, available at `http://arXiv.org/abs/math/0105031`.

[Kob87] N. Koblitz, Elliptic curve cyptosystem, *Math. Comp.*, **48**(177), pp. 203-209, 1998.

[Lan94] S. Lang, *Algebraic Number Theory*, Springer-Verlag, 1994.

[LST64] J. Lubin, J. P. Serre, and J. Tate. Elliptic curves and formal group. *Lecture notes in prepared in connection with the seminars held at the Summer institute on Algebraic Geometry, Whitney Estate, Woods Hole, Massachusetts*, 1964.

[Men1] A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.

[Men2] A. Menezes, *Application of Finite Fields*, Kluwer Academic Publisher, 1993.

[Mil87] V. Miller, Use of elliptic curves in cryptography. *Crypto'86*, LNCS **263**, pp. 417–426, 1987.

[Mur88] M. R. Murty, Artin's conjecture for primitive roots, *Math. Intelligencer*, **10** (4), pp. 59–67, 1988.

[PS73] M. S. Parterson and L. J. Stockmeyer, On the number of nonscalar multiplications necessary to evaluate polynomials. *SIMA J. Comput.*, **2**, pp. 60–67, 1973.

[Sat00] T. Satoh, The canonical lift of an ordinary elliptic curve over a finite field and its point counting, *J. Ramanujan Math. Soc.*, **15**, pp. 247–270, 2000.

[Sch85] R. Schoof, Elliptic curves over finite fields and the computation of square roots mod p, *Math. Comput.*, **44**, pp. 483–494, 1985.

[Sil99] J. H. Silverman, Fast Multiplication in Finite Fields GF(2^N), *Crytographic Hardware and Embedded Systems - CHES'99*, LNCS **1717**, pp. 122-134, Springer-Verlag, 1999.

[Skj00] B. Skjernaa, Satoh Point Counting in characteristic 2. To appear in *Math. Comp.*

[SST01] T. Satoh, B. Skjernaa, and Y. Taguchi, Fast Computation of Canonical Lifts of Elliptic curves and its Application to Point Counting, *Preprint*, 2001.

[VPV01] F. Vercauteren, B. Preneel, and J. Vandewalle, A Memory Efficient Version of Satoh's Algorithm. *Advances in Cryptology - Eurocrypt 2001*, LNCS **2045**, pp. 1–13, Springer-Verlag, 2001.