Elliptic curve point counting over finite fields with Gaussian normal basis

By Je Hong Park, Jung Youl Park, and Sang Geun Hahn

Department of Mathematics, Korea Advanced Institute of Science and Technology(KAIST)
373-1, Kusung-dong, Yusung-ku, Taejon 305-701, South Korea
(Communicated by Heisuke Hironaka, m. j. a., Jan. 14, 2003)

Abstract: In this paper, we present the GNB-aided MSST algorithm for the curves over finite fields that have a Gaussian normal basis of type $t \leq 2$. It is based on the MSST algorithm proposed by P. Gaudry [3] at ASIACRYPT 2002. For those fields, we combine the lifting phase of the MSST algorithm and the norm computation algorithm in [6]. So the time complexity of the MSST is reduced from $O(N^{2\mu+0.5})$ to $O(N^{2\mu+1/(\mu+1)})$ and it runs faster than any other algorithms in our case.

Key words: Elliptic curve; order counting; Gaussian normal basis; finite field; cryptography.

1. Introduction. Since Satoh had proposed an efficient elliptic curve point counting algorithm for the curves defined over finite fields of small characteristic $p \geq 5$ [7], some works have been done on this subject to extend it to the characteristics $p = 2, 3$ [2, 10] and to use less memory [11]. Last year, Satoh-Skjernaa-Taguchi [9], and Harley-Mestre-Gaudry [5] separately proposed different efficient elliptic curve point counting algorithms, which are called the SST and AGM algorithms, respectively. Recently, Gaudry put them together into more efficient algorithm to announce the modified SST algorithm (MSST). It modifies only the lifting phase of the SST by adapting the idea of the AGM, and so it has the same complexity as that of the SST; but it removes the intermediate step between the lifting and the norm computation phase in the SST, and simplifies the implementation codes. As a result, it runs faster by a constant factor than the original one although its time complexity remains the same.

In this paper, we present the GNB-aided MSST algorithm for the curves over finite fields that have a Gaussian normal basis of type $t \leq 2$. In our case, it has lower time complexity than that of the original MSST so it runs faster.

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 introduce the notion of a Gaussian normal basis. We briefly review the MSST algorithm and describe how our idea can be applied to point counting in Section 3. In Section 4, we exhibit our practical results and notes for implementation. Finally, this paper ends up with some comments in Section 5.

Notations. Throughout the rest of this paper, let $p = 2$ and $q = p^N$, where N be a positive integer. We denote the unramified extension of degree N of $\mathbf{Q}_p$ by $\mathbf{Q}_q$ and its valuation ring by $\mathbf{Z}_q$. In fact we consider $\mathbf{Z}_q \bmod p^M$ instead of $\mathbf{Z}_q$ for a sufficiently large integer M which is called a precision. Also, an operation is said to be done with a precision M if it is performed modulo p^M. We let σ stand for the Frobenius substitution in $\mathrm{Gal}(\mathbf{Q}_q/\mathbf{Q}_p)$, and π be the reduction map by p from $\mathbf{Q}_q$ to $\mathbf{F}_q$. We assume that E is a non-supersingular elliptic curve over $\mathbf{F}_q$ whose equation is $y^2 + xy = x^3 + a_6$, and that $j(E) = a_6^{-1}$ is its j-invariant where $j(E) \in \mathbf{F}_q \backslash \mathbf{F}_{p^2}$. It is well known that $|E(\mathbf{F}_q)| = q + 1 - T$, where T is a trace of the Frobenius endomorphism of E.

2. Gaussian normal basis. When K is a field and L/K is 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*. In this section, we concentrate our interest on a normal basis, especially which is generated by a Gauss period that is defined below.

Definition 1 [6]**.** Let N and t be positive integers such that $Nt+1$ is a prime not dividing p. Let τ be any primitive t-th root of unity in $\mathbf{Z}/(Nt+1)\mathbf{Z}$.

2000 Mathematics Subject Classification. Primary 11G20; Secondary 11G07, 11T71.

Let γ be a primitive $(Nt+1)$-th root of unity in some extension field of $\mathbf{F}_p$. A *Gauss period* of type (N,t) over $\mathbf{F}_p$ is defined as

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

Let us call a normal basis induced by the Gauss period of type (N,t) the *Gaussian normal basis of type t* and denote it by GNB of type t. It is easy to see that the Gauss period of type (N,t) belongs to $\mathbf{F}_q$. 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 [6]. *Let N and t be positive integers in Definition* 1. *Let e be the order of p modulo $Nt+1$. Then* $\gcd(Nt/e, N) = 1$ *if and only if the Gauss period of type (N,t) over $\mathbf{F}_p$ generates the normal basis for $\mathbf{F}_q$ over $\mathbf{F}_p$.*

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.* [1] showed that this idea could be extended to a GNB of type 2 by using a symmetric polynomial of double length.

In [6, Section 4], it is showed that if $\mathbf{F}_q/\mathbf{F}_p$ has a GNB, then it can be lifted to $\mathbf{Z}_q$ in a natural way and so efficient multiplication and the Frobenius substitution are available because the defining polynomial of the finite field has a sparse structure. Moreover, a fast norm computation algorithm is derived, so we focus on finite fields with a GNB of type $t \leq 2$ considering the practical reason. Refer [6] for more details about GNB of type 1 and 2 over finite fields and its lifting to p-adic fields.

3. Application to point counting.

3.1. Modified SST algorithm. The p-adic method for elliptic curve point counting which was firstly proposed by Satoh attempts to construct a p-adic lift of the Frobenius endomorphism to characteristic zero. The main strategy is to lift E given over $\mathbf{F}_q$ to an elliptic curve over a certain p-adic ring $\mathbf{Z}_q$ above $\mathbf{F}_q$. By a result of Lubin-Serre-Tate, there is a canonical way to lift the curve which is called the canonical lift of E, by lifting its j-invariant j from $\mathbf{F}_q$ to $\mathbf{Z}_q$ using the modular polynomial Φ_p. Since the canonical lift $E^\uparrow$ satisfies $\pi(E^\uparrow) = E$ and $\mathrm{End}(E) \cong \mathrm{End}(E^\uparrow)$, the Frobenius endomorphism also is lifted to an endomorphism of the lifted curve. Satoh showed that once one obtains the lifted j-invariant $j^\uparrow$ and the dual of the Frobenius endomorphism of $E^\uparrow$, he can calculate T [7].

The SST algorithm proposed by Satoh-Skjernaa-Taguchi, improves the lifting phase of previous p-adic method by using the Frobenius substitution of $\mathbf{Z}_q$. Independently, the AGM computes the canonical lift by iterating the AGM sequence (Refer [3, 8] for more details about the AGM.). Then, a norm computation gives the trace of the initial curve up to some precision. Because it does not use the modular equation Φ_p and removes the intermediate phase between the lifting and norm computation phases, more fast implementation results are obtained.

In the MSST algorithm, the idea of the AGM algorithm is used to reveal the polynomial which can be applicable in place of the modular equation Φ_p in the SST algorithm; that is

$$\tilde{E}(X,Y) = (X + 2Y + 8XY)^2 + Y + 4XY = 0.$$

In detail, we know by the AGM that the lifting of the AGM sequence is more efficient than that of j-invariant because it induces fewer operations and more simple procedure of algorithm. So in the MSST, $\tilde{E}$ plays the same role of Φ_p in the SST with reduced number of operations [3]. Furthermore, if λ is a solution of $\tilde{E}(X, X^\sigma)$ and $\lambda \equiv a_6 \bmod p$, then the following holds:

$$T \equiv \frac{1}{\mathrm{Norm}(1+4\lambda)} \mod q.$$

The description of the MSST is as follows:

Algorithm 1. MSST algorithm

Require: a_6 in $\mathbf{F}_{2^n}^*$
Ensure: $T(E)$
1: $y \leftarrow a_6$ (arbitrary lift to 2^W)
2: **for** i from 1 to $W-1$ **do**
3: $x \leftarrow \sigma^{-1}(y) \mod 2^{i+1}$
4: $y \leftarrow y - \tilde{E}(x,y) \mod 2^{i+1}$
5: **end for**
6: $x \leftarrow \sigma^{-1}(y) \mod 2^W$
7: $D_X \leftarrow \partial_X \tilde{E}(x,y) \mod 2^W$
8: $D_Y \leftarrow \partial_Y \tilde{E}(x,y) \mod 2^W$
9: **for** m from 1 to $\lfloor \frac{N}{2W} \rfloor$ **do**
10: Lift arbitrarily y modulo $2^{(m+1)W}$
11: $x \leftarrow \sigma^{-1}(y) \mod 2^{(m+1)W}$

12: $V \leftarrow \tilde{E}(x,y) \mod 2^{(m+1)W}$
13: **for** i from 1 to $W-1$ **do**
14: // break if $i + mW \geq \lceil \frac{N}{2} \rceil$
15: $\delta_Y \leftarrow -2^{-mW} V \mod 2^W$
16: $\delta_X \leftarrow \sigma^{-1}(\delta_Y) \mod 2^W$
17: $y \leftarrow y + 2^{mW}\delta_Y \mod 2^{(m+1)W}$
18: $V \leftarrow V + 2^{mW}(D_X\delta_X + D_Y\delta_Y) \mod 2^{(m+1)W}$
19: **end for**
20: **end for**
21: Return $\frac{1}{\mathrm{Norm}(1+4\lambda)} \mod 2^{\lceil \frac{N}{2} \rceil + 2}$ in $[-2\sqrt{2^N}, 2\sqrt{2^N}]$

3.2. GNB-aided MSST algorithm. The SST and MSST algorithms require much computations of σ. In order to speed up the computation, the MSST algorithm uses a dense defining polynomial for $\mathbf{Z}_q$, but a reduction modulo the defining polynomial subsequent to a multiplication or a square becomes costly. To avoid this harassment, we use a GNB to represent elements of $\mathbf{Z}_q$. A benefit of a GNB is to make an arithmetic in the field just as fast as that with a sparse defining polynomial without precomputation and have an easy formula for σ. Furthermore, we can use the norm computation algorithm proposed in [6]. By using the 2-adic expansion of N, this algorithm requires fewer multiplications and more Frobenius substitutions. Let us denote the binary expansion of $N = \sum_{i=0}^{l} n_i 2^i$ by $[n_0, n_1, \ldots, n_l]_2$, where $n_l = 1$. Since $\mathrm{Gal}(\mathbf{Q}_q/\mathbf{Q}_p)$ is generated by σ, we obtain that

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

where $M_i = (\sigma^{2^{i-1}}M_{i-1})\cdot M_{i-1}$ and $M_0 = A$. From this equation, we can derive a norm computation algorithm [6]. With precision M, it runs in $O((NM)^\mu \log N)$ time with $O(NM)$ space, while that of the MSST algorithm runs in $O((NM)^\mu M^{0.5})$ time with $O(NM)$ space, where μ is a constant such that the multiplication of two n bit integers can be carried out with $O(\mu)$ bit operations. Note that $M = N/2 + O(1)$ in both cases. Because the time complexity of the lifting phase of the MSST algorithm is $O(N^{2\mu+1/(\mu+1)})$, the complexity of whole algorithm can be reduced from $O(N^{2\mu+0.5})$ to $O(N^{2\mu+1/(\mu+1)})$ in our case.

Algorithm 2. Norm computation algorithm
Require: $A \in \mathbf{Z}_q,\ N = [n_0, n_1, \ldots, n_l]_2,\ n_l = 1$
Ensure: $\mathrm{Norm}_{\mathbf{Q}_q/\mathbf{Q}_p}(A)$
1: $M \leftarrow A$
2: **if** $n_0 = 1$ **then**
3: $T \leftarrow \sigma^{N-1}A$
4: **else**
5: $T \leftarrow 1$
6: **end if**
7: **for** $i = 1$ to $l-1$ **do**
8: $M \leftarrow (\sigma^{2^{i-1}}M)\cdot M$
9: **if** $n_i = 1$ **then**
10: $T \leftarrow T\cdot(\sigma^{n_{i+1}2^{i+1}+\cdots+n_l 2^l}M)$
11: **end if**
12: **end for**
13: $M \leftarrow (\sigma^{2^{l-1}}M)\cdot M$
14: $M \leftarrow M\cdot T$
15: Return M

4. Implementation results. Using this GNB-aided MSST algorithm, we can compute the order of a given elliptic curve. Our algorithm has been implemented in C programming language for the most part, and some assembly for the most basic operations of multi-precision integers. Our computational circumstance is on a Pentium III-800MHz processor with 128MB main memory, running Linux and all codes are compiled using `gcc` version 2.96.

For efficiency we use a fixed value $W = 32$, a machine word size of 32-bit processors like P-III; hence in many steps operations are performed within a constant precision. It allows us to eliminate much of the loop overhead. For $t = 2$, we use a palindromicity to store only halves of the polynomials representing elements of $\mathbf{Z}_q$, while elements of $\mathbf{F}_q$ are always represented as full-size polynomials [6]. Multiplication of two elements in $\mathbf{Z}_q$ is implemented using Karatsuba's method. The following Table I shows our implementation results.

For the comparison, we present the recent results of the MSST algorithm [3] in Table II. Gaudry obtained the result on Pentium III 700MHz running Linux. He compiled his codes using `gcc` version 2.96 and the GNU MP library to handle multi-precision integers.

For a researching interest, we also show out results for large N for GNBs of type $t \leq 2$ in the Table III, varying W.

Table I. Timings for computations of order counting (unit:sec)

N	t	Total	N	t	Total
162	1	0.0370	233	2	0.1955
173	2	0.0885	239	2	0.2190
194	2	0.1245	268	1	0.1705
196	1	0.0665	292	1	0.1990
209	2	0.1520	293	2	0.4080
210	1	0.0800	299	2	0.4345

Table II. Comparison MSST with GNB-aided MSST (unit:sec)

N	t	Lifting	Norm	Total	Note
163		0.08	0.05	0.13	[3]
163	4	0.143	0.126	0.269	
239		0.26	0.14	0.40	[3]
239	2	0.099	0.120	0.219	

Table III. Timings for computations of norm and order counting for large N (unit:sec)

N	t	W	Lifting	Norm	Total
3010	1	96	334.21	173.0	507.24
3005	2	96	651.60	405.13	1056.75
6010	1	128	3324.62	2245.59	5570.22
6005	2	128	6730.50	4503.88	11234.40
12010	1	192	38206	26907	65113

5. Conclusion and remark. In this paper, we reduced the time complexity of the MSST from $O(N^{2\mu+0.5})$ to $O(N^{2\mu+1/(\mu+1)})$ on finite fields that have a GNB of type $t \leq 2$ by combining the lifting phase of the MSST and the norm computation algorithm from [6]. As a result, very fast point counting is available on finite fields with a GNB of type $t \leq 2$ and our method works well for a very large N because of the less complexity. Our implementation result shows this fact.

Similar work has been appeared at number theory archive [4]. This work is done independently.

Acknowledgement. The authors of this paper would like to thank Pierrick Gaudry, Robert Harley, Jung Hee Cheon and Dong Hoon Lee for valuable comments. This work was supported by International Research Center for Information Security (IRIS), Information and Communications University. (ICU), Korea.

References

[1] Blake, L. F., Roth, R. M., and Seroussi, G.: Efficient Arithmetic in GF(2^n) through Palindromic Representation. Tech. Rep. HPL-98-134, Hewlett Packard (http://www.hpl.hp.com/techreports/). (1998).

[2] Fouquet, M., Gaudry, P., and Harley, R.: An extension of Satoh's algorithm and its implementation. J. Ramanujan Math. Soc., **15**, 281–318 (2000).

[3] Gaudry, P.: A Comparison and a Combination of SST and AGM Algorithms for Counting Points of Elliptic Curves in Characteristic 2. Proc. of ASIACRYPT 2002, Lecture Notes in Comput. Sci., 2501, Springer-Verlag, Berlin, pp. 311–327 (2002).

[4] Harley, R.: Elliptic Curve Point Counting: 32003 bits. Series of e-mails to the NUMBERTHRY mailing list, Nov. 14th (2002).

[5] Harley, R., Mestre, J.F., and Gaudry, P.: Counting points with the arithmetic-geometry mean. EUROCRYPT 2001, Rump session (2001).

[6] Kim, H. Y., Park, J. Y., Cheon, J. H., Park, J. H., Kim, J. H., and Hahn, S. G.: Fast Elliptic Curve Point Counting Using Gaussian Normal Basis. Proc. of ANTS V, Lecture Notes in Comput. Sci., 2369, Springer-Verlag, Berlin, pp. 292–307 (2002).

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

[8] Satoh, T.: On p-adic Point Counting Algorithms for Elliptic Curves over Finite Fields. Proc. of ANTS-V, Lecture Notes in Comput. Sci., 2369, Springer-Verlag, Berlin, pp. 43–66 (2002).

[9] Satoh, T., Skjernaa, B., and Taguchi, Y.: Fast computation of canonical lifts of elliptic curves and its application to point counting. Finite Fields Appl., **9**, 89–101 (2003).

[10] Skjernaa, B.: Satoh point counting in characteristic 2. Math. Comp., **72**, 477-487 (2003).

[11] Vercauteren, F., Preneel, B., and Vandewalle, J.: A memory efficient version of Satoh's algorithm. Proc. of EUROCRYPT 2001, Lecture Notes in Comput. Sci., 2045, Springer-Verlag, Berlin, pp. 1–13 (2001).