Space Efficient Computation of Scalar Multiplication with Applications to Hand-held Devices

Anonymous Submission

No Institute Given

Abstract. We present a new approach for efficient computation of scalar multiplication using less storage space. We know that ECC is more suitable for hand-held devices like cell phones, smart-cards, PDAs, etc., because of their shorter key sizes with significant level of security compared with other conventional cryptosystems. One of the major constraints of hand-held devices is memory. It is worthwhile to have some techniques to compute on less storage space so that we can apply such techniques on any hand-held devices where memory is a constraint. Our approach incorporates the technique called the fixed base comb method into the Koblitz's idea of using Frobenius Map for computing scalar multiplication. With our approach the storage space needed during the preprocessing stage of scalar multiplication is tremendously improved from $s \times 3^h + 3^r$ to $n \times 2^w$, where $n = s \times h + r$ and w is word size.

Keywords: ECC, Scalar multiplication, Hand-held devices, Frobenius map.

1 Introduction

Arithmetic in finite fields is an integral part of many public-key algorithms, including those based on the discrete logarithm problem in finite fields, elliptic curve based schemes, and emerging applications of hyperelliptic curves [6]. The ability to quickly perform arithmetic in the underlying finite field determines the performance of these schemes. Finite fields are identified with the notation $GF(p^m)$, where p is a prime and m is a positive integer. In elliptic [6, 7, 9] and hyperelliptic curve cryptosystems, main operations such as key agreement and signing/verifying involves scalar multiplication using a large integer k. The speed of scalar multiplication plays an important role in the efficiency of the whole system. In particular, fast multiplication is more crucial in some environments such as central servers, where large numbers of key agreements or signature generations occur, and in hand-held devices with low computational power.

There are several points that influence the speed of multiplication: the choice of base field, the choice of curve, the representation of a point, the representation of a scalar, and the multiplication algorithm. There are several multiplication methods used on general elliptic and hyperelliptic curves. The efficiency of an elliptic and hyperelliptic curve cryptosystems is crucially dependent on the speed of scalar multiplication. This has led to a tremendous research in algorithms for efficient scalar multiplication. These algorithms fall into two classes.

- General algorithms which work for any cyclic group.
- Algorithms which exploit the algebraic properties of elliptic and hyperelliptic curves.

One of the most important techniques of the second kind is the use of endomorphisms to speed up scalar multiplication. This was first proposed by Koblitz [11] and has also been studied by many researchers in [4, 5, 8, 10, 12, 18, 19]. The most natural endomorphism is the Frobenius automorphism

and was initially proposed by Koblitz [11]. A series of research papers have resulted in the applicability of the Frobenius map technique to elliptic and hyperelliptic curves over any finite field [12]. In our work, we apply the fixed base comb technique to the Koblitz's idea of using Frobenius map for computing scalar multiplication, which results in tremendous improvement in the storage space needed to store the points during the preprocessing stage of scalar multiplication operation. So that, our scheme is very applicable to any hand-held devices where memory is a constraint.

Let $\mathbb{F}_q$ be the underlying field. The Frobenius map technique applies when q is a prime power (rather than a prime). The case $p = 2$ has been explored extensively by the research community. Our algorithm applies to the case $p > 2$, for example Optimal Extension Fields [1, 2]. Optimal Extension Fields (OEFs) are finite fields of the form $GF(p^m),\ p > 2$, where p and m are chosen to match the underlying hardware. OEFs, optimally utilizing the underlying hardware offer considerable advantage in software implementations of Elliptic curve cryptosystems. OEFs are special cases of prime power fields. In this paper, we mainly focus on space efficient computation of scalar multiplication as an application to hand-held devices like cell phones, smart-cards, PDAs, etc. We know that one of the main constraints of hand-held devices is a limited memory. Our motivation is to come with techniques which can be used to compute scalar multiplication even on less storage space.

This paper is organized in the following manner. In Section 2, we present the basic material required. In Section 3, we give the detailed description of the Fixed base comb method which we incorporate appropriately into the idea of Frobenius automorphism. In Section 4, we present our approach of efficient computation of the scalar multiplication whereas in Section 5 we compare our approach with some of the already existing algorithms. Finally we give some conclusions and further directions of the work in Section 6.

2 Preliminaries

Here in this section we give some background material needed to understand the rest of the work. In our work, we use the term *point* repeatedly, which we will mean either a point on an elliptic curve or a reduced divisor of an hyperelliptic curve. The main operation for realizing elliptic and hyperelliptic curve cryptosystems is computing kP, where k is an integer and P is a point. This operation is called the *scalar multiplication*. Our focus, in this work, is to obtain space efficient algorithms for scalar multiplication for curves over $GF(p^n)$ using Frobenius map.

2.1 Prime power fields

Binary fields $GF(2^n)$ and prime fields $GF(p)$ were considered as most attractive for software implementation of cryptographic applications. A special class of finite fields known as Optimal Extension Fields(OEFs) were proposed [14], where p and n were chosen appropriately to exploit the underlying hardware optimally for performance gain. An OEF is a finite field of the form $GF(p^n)$, where

- p is a Pseudo-mersenne prime
- an irreducible binomial $P(x) = x^n - \omega$ exits over $GF(p)$.

The prime p is generally chosen to be very close to the word size of the processor, so that each machine word can accommodate one element of the subfield $GF(p)$ and each element of the OEF $GF(p^n)$, can be accommodated in n words, with minimum wastage of memory. Also, OEFs allow efficient modular reduction for arithmetic in the extension field. The algorithms proposed in this work are suitable for the prime power fields of type $GF(p^n)$, which contains the OEFs as a subclass of it.

2.2 Normal basis

A field $\mathbb{F}_{q^n}$ is said to have a normal basis if it has a basis (over $\mathbb{F}_q$) of the form $\{\alpha, \alpha^q, \cdots, \alpha^{q^{n-1}}\}$. Any element of the field can be represented as $x = \sum_{j=0}^{n-1} a_j \alpha^{q^j}$ or briefly as an ordered n-tuple $x = (a_0, a_1, \cdots, a_{n-1})$. In the field $\mathbb{F}_{q^n}$, we have

$$x^q = (\sum_{j=0}^{n-1} a_j \alpha^{q^j})^q = \sum_{j=0}^{n-1} a_j \alpha^{q^{j+1}}.$$

Thus if x is represented by the tuple $(a_0, a_1, \cdots, a_{n-1})$ then x^q is represented by $(a_{n-1}, a_0, \cdots, a_{n-2})$, as $\alpha^{q^n} = 1$. With a normal basis representation of elements, x^q can be computed from x by a circular shift operation only. For further details on normal basis, interested reader can refer [13].

2.3 Frobenius map

The Frobenius map [3, 20] $\phi : \mathbb{F}_{q^n} \to \mathbb{F}_{q^n}$ is an automorphism of $\mathbb{F}_{q^n}$ and is defined as

$$\phi(x) = x^q.$$

The map is extended to points of an elliptic or hyperelliptic curve over $\mathbb{F}_{q^n}$ in the following manner: A point of an elliptic curve is represented using a pair of elements of $\mathbb{F}_{q^n}$; similarly a reduced divisor of a hyperelliptic curve is represented using a tuple of elements of $\mathbb{F}_{q^n}$. An application of the Frobenius map to a point is to actually apply the map individually to the field elements which represent the point. We note that ϕ^n is the identity map of $\mathbb{F}_{q^n}$. If the field $\mathbb{F}_{q^n}$ is represented using a normal basis, then the computation of $\phi(x)$ is "for free". Further, as observed in [18, 19], in the case $q = 2$, the Frobenius map is $\phi(x) = x^2$ and hence can be computed using a field squaring which is a relatively cheap operation even if polynomial basis representation of elements is used.

2.4 Scalar Multiplication using Frobenius map

Koblitz in [11], very first time suggested the use of Frobenius map to speed up scalar multiplication algorithm. Later on, this idea has been developed by different authors [5, 8, 10, 15, 17, 19]. Let q be a prime power, $\mathbb{F}_q$ be the finite field of order q and $\mathbb{F}_{q^n}$ an extension field of $\mathbb{F}_q$. Let C be the curve of genus g to be used for the cryptosystem and we consider the $\mathbb{F}_{q^n}$-rational points of C. Let ϕ be the Frobenius map from $\mathbb{F}_{q^n}$ to $\mathbb{F}_{q^n}$. Let k be an integer, and P be a point on an elliptic curve or a reduced divisor of a hyperelliptic curve and we wish to compute kP. For hyperelliptic curves, it has been shown that the Frobenius map based method can be used over any field of finite characteristic.

The base ϕ-expansion of k is $\sum_{i=0}^{n-1} u_i \phi^i$, where under reasonable assumptions each u_i is an integer in the range $[-q^g, q^g]$. It is possible to obtain the base-ϕ expansion of k [14].

Next we will define some additional parameters which will be required for the rest of the work in the paper.

- $A = max\lfloor(|u_i|)\rfloor$.
- For $i \in \{0, 1, \cdots, n-1\}$ write $|u_i| = \sum_{j=0}^{A} u'_{i,j} 2^j$, where $u'_{i,j} \in \{0, 1\}$.
- $u_{i,j} = \mathbf{sgn}(u_i) u'_{i,j}$, where $\mathbf{sgn}(u_i)$ is the sign of u_i.
- For $0 \leq i \leq n-1$, define $X_0 = X$ and $X_i = \phi^i(X_0) = \phi^i(X)$.

The expression kP can be written as

$$\begin{aligned} kP &= u_0 P_0 + u_1 P_1 + \cdots + u_{n-1} P_{n-1} \\ &= (u_{0,0} 2^0 + u_{0,1} 2^1 + \cdots + u_{0,A} 2^A) P_0 \\ &\quad + (u_{1,0} 2^0 + u_{1,1} 2^1 + \cdots + u_{1,A} 2^A) P_1 \\ &\quad \cdots\cdots\cdots\cdots \\ &\quad + (u_{n-1,0} 2^0 + u_{n-1,1} 2^1 + \cdots + u_{n-1,A} 2^A) P_{n-1} \end{aligned}$$

We consider this expression as a matrix of order $n \times (A+1)$

Now we will have a look into the several cases depending on the nature of the underlying field $\mathbb{F}_p$, where $p = q^n$.

1. **When $n = 1$ and p is prime :** In this case, the above expression reduces to a single row. In this situation, the Frobenius map based technique does not really apply. Hence we will not consider this kind of fields in our work.

2. **When $n > 1$:** In this case, the above expression will have more than one row and the Frobenius map technique can be applied. It will be convenient to divide this into two sub-cases.

   - If $q = 2$, the field is $\mathbb{F}_{2^n}$ and the curves are the Binary Koblitz curves. In this case each $u_i \in \{0, \pm 1\}$ and hence the above expression is actually a single column. This is the other extreme to case 1 above. In [18, 19], the above expression is called the ϕ-adic expansion of m.

   - If $q > 2$, then the above expression has a more square shape and again our algorithm offers improvements over existing algorithms.

Algorithm I: The Basic Approach

INPUT: Given an integer $k = \sum_{i=0}^{n-1} u_i \phi^i$ and a point P
OUTPUT: Compute kP

1. *for* $0 \le i \le n-1$ *and* $0 \le j \le A$, *compute* X_i *and* $u_{i,j}$
2. *set* $Y = \sum_{i=0}^{n-1} u_{i,A} X_i$
3. *for* $j = A-1$ *downto* 0
4. $\quad Y = 2Y; Y = Y + \sum_{i=0}^{n-1} u_{i,j} X_i$
5. *return* Y

By using the above Algorithm I, we can compute the scalar multiplication kP by using $n(A+1)$ and A additions and doublings respectively. The expected running time of Algorithm I is $(((2^w - 1)/2^w)d - 1)A + (d-1)D$.

3 Fixed-base Comb Method

In the fixed-based comb method [9], the binary representation of k is first padded on the left with $dw - A$ 0s, and is then divided into w bit strings each of the same length $d = \lceil A/w \rceil$ so that

$$u_i = U^{w-1} \| \cdots \| U^1 \| U^0, \text{ for } 0 \le i \le n-1$$

The bit strings U^j are written as rows of an *exponent array*

$$\begin{bmatrix} U^0 \\ \vdots \\ U^{w'} \\ \vdots \\ U^{w-1} \end{bmatrix} = \begin{bmatrix} U^0_{d-1} & \cdots & U^0_0 \\ \vdots & \cdots & \vdots \\ U^{w'}_{d-1} & \cdots & U^{w'}_0 \\ \vdots & \cdots & \vdots \\ U^{w-1}_{d-1} & \cdots & U^{w-1}_0 \end{bmatrix}$$

$$= \begin{bmatrix} U_{d-1} & \cdots & U_0 \\ \vdots & \cdots & \vdots \\ U_{(w'+1)d-1} & \cdots & U_{w'd} \\ \vdots & \cdots & \vdots \\ U_{wd-1} & \cdots & U_{(w-1)d} \end{bmatrix}$$

where U_l is the l^{th} bit of u_i and whose columns are then processed one at a time. In order to accelerate the computation, the points

$$[a_{w-1}, \cdots, a_2, a_1, a_0]P = a_{w-1}2^{(w-1)d}P + \cdots + a_2 2^{2d}P + a_1 2^d P + a_0 P$$

are precomputed for all possible bit strings $(a_{w-1}, \cdots, a_1, a_0)$.

Algorithm II: Fixed-base comb method for point multiplication

INPUT: Given an integer k, *word size* $w, d = \lceil A/w \rceil, u_i = (u_{i,A}, \cdots, u_{i,1}, u_{i,0})_2, P \in E(\mathbb{F}_q)$
OUTPUT: Compute kP

1. **precomputation:** *compute* $[a_{w-1}, \cdots, a_1, a_0]$ *for all bit strings* $(a_{w-1}, \cdots, a_1, a_0)$ *of length* w
2. *Add* 0*s on the left of* k *if necessary, write* $k = U^{w-1} \| \cdots \| U^1 \| U^0$, *where each* U^j *is a bit string of length* d. *Let* U_i^j *denote the* i^{th} *bit of* U^j
3. $Q \leftarrow \infty$
4. *for* i *from* $d-1$ *downto* 0 *do*
   (a) $Q \leftarrow 2Q$
   (b) $Q \leftarrow Q + [U_i^{w-1}, \cdots, U_i^1, U_i^0]P$
5. *return* Q

We know that the expected running time of Algorithm I is

$$(((2^w - 1)/2^w)d - 1)A + (d-1)D$$

If we treat $(2^w - 1)/2^w \approx 1$, then we can compute the scalar multiplication kP with $d-1$ additions and $d-1$ doublings by using the Algorithm II. But, this approach will require precomputation and hence will occupy 2^w storage space, where w is the word size. Whereas in the Algorithm I, no precomputations are needed.

4 Our New Approach

Here in this section, we explain our new approach for computing scalar multiplication by applying the concept of fixed base comb to Koblitz's idea of using Frobenius map. Given an integer k and a point P, we have to compute kP.

We will define

$$kP = u_0P_0 + u_1P_1 + \cdots + u_{n-1}P_{n-1}$$

where $P_0 = P$ and $P_i = \phi^i(P)$, which require n Frobenius map computations. Each u_i is an integer in the interval $[-q^g, q^g]$. It is possible to obtain the base-ϕ expansion of k. We will compute each u_iP_i, for $0 \leq i \leq n-1$, separately. Finally with $n-1$ additions we can obtain the scalar multiplication kP.

Now we will give the algorithm to compute u_iP_i $\forall i$.

Algorithm III: Sub-Algorithm

INPUT: Given u_i *and* P_i {here subscript i is used for notational consistency}
OUTPUT: Compute u_iP_i

1. **precomputation:** *compute* $[a_{w-1}, \cdots, a_1, a_0]P_i$, *for all bit strings* $(a_{w-1}, \cdots, a_1, a_0)$ *of length* w
2. *by padding* u_i *on the left with 0s if necessary, write* $u_i = U^{w-1} \| \cdots \| U^1 \| U^0$, *where each* U^j *is a bit string of length* d. *Let* U_l^j *denote the* l^{th} *bit of* U^j
3. $Q \leftarrow \infty$
4. *for* l *from* $d-1$ *downto* 0 *do*
   (a) $Q \leftarrow 2Q$
   (b) $Q \leftarrow Q + [U_l^{w-1}, \cdots, U_l^1, U_l^0]P_i$
5. *return* Q

Here $d = \lceil A/w \rceil$, where A is the maximum number of bits used to represent any $u_i, 0 \leq i \leq n-1$. In Algorithm III, the precomputation step will require 2^w points to be stored. This algorithm will take $(((2^w - 1)/2^w)d - 1)$ additions and $d - 1$ doubles. Here in Algorithm III, we are computing only one component of the kP representation. This is to be repeated n times and finally with $n-1$ additions we will get the required scalar multiplication kP.

Here we will give the generalized algorithm for computing scalar multiplication.

Algorithm IV: Scalar Multiplication Algorithm

INPUT: Given k and P
OUTPUT: Compute kP

1. *precompute* P_i *for* $0 \leq i \leq n-1$ *using Frobenius map.*
2. *for i from 0 to* $n-1$ *do*
   (a) *compute* u_iP_i *using Algorithm III*
   (b) *sum = sum +* Q_i
3. $Q \leftarrow$ *sum*
4. *return Q*

We will consider $(2^w - 1)/2^w \approx 1$. Then the algorithm IV requires nd additions and $n(d-1)$ doublings. Total storage space required is $n \times 2^w$.

5 Comparisons with Existing Approaches

Recently in [14, 16], the authors have given a new table look-up method for computing the scalar multiplication using Frobenius map. They have computed the scalar multiplication using A additions and A doublings, but during the precomputation stage their approach requires the huge storage space to store 3^n points, which is not ideal to implement such schemes in hand-held devices where memory is a constraint. Even for moderate values of n, huge amount of storage space is needed to store 3^n points. To minimize this problem, the authors in [14, 16] have introduced the smaller table look-up methods. By this approach, the number of points to be stored has been reduced from 3^n points to $s \times 3^h + 3^r$, where $n = s \times h + r$, which requires significantly less storage space compared with the storage space needed to store 3^n points. But the number of additions increased from A to $(s+1)A$, whereas the number of doublings remain the same.

We argue that even with the smaller table look-ups, the amount of storage space needed to store the points during the preprocessing stage is significantly high. To alleviate this problem to maximum extent possible, we have come up with a new approach, by which it is sufficient to store only $n \times 2^w$ points. From this it is clear that our approach minimizes the storage space significantly during the preprocessing stage of scalar multiplication over elliptic curve, and so is applicable to any hand-held devices where memory is a constraint. In our approach, we use the fixed-base comb method to compute each u_iP_i for $0 \leq i \leq n-1$. Where

$$kP = u_0P_0 + u_1P_1 + \cdots + u_{n-1}P_{n-1}.$$

By doing so, we can compute the scalar multiplication using nd additions and $n(d-1)$ doublings. Whereas during the precomputation stage we need storage space to store $n \times 2^w$ points only, where w is the word size. In our approach, the amount of storage space needed during the preprocessing stage is significantly improved over the amount of storage needed during the preprocessing stage in the existing approaches [16].

Here we analyze and highlight the results of our approach to compute scalar multiplication with those of the existing schemes by means of graphs. In both the following graphs, x-axis represents word sizes and y-axis represents the number of points. Figures $1(a)$ and $2(a)$ are the results of our scheme whereas Figures $1(b)$ and $2(b)$ are the results of the existing approach.

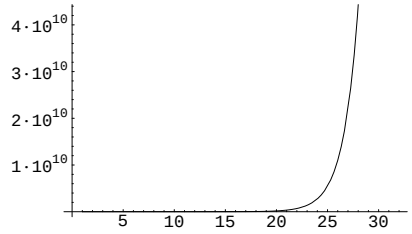


Figure : 1 (a)

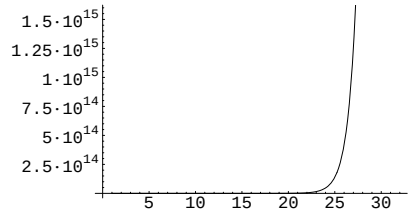


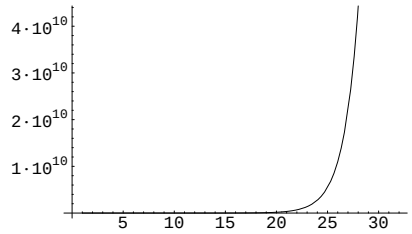
Figure : 1 (b)

Fig. 1. For the case $w = h$, where w is word size, the amount of storage space needed to store $n \times 2^w$ points is much lesser than the amount of storage space needed to store $s \times 3^h + 3^r$ points. In this case the number of additions remains the same whereas the number of doublings increased by n/w times, and we claim that this order of increase in the number of doublings is not so significant.

For the case $w > h$, our approach will take slightly more storage space, and we will get moderate improvement over the number of additions and doublings required. Its quite unlikely to occur this case as the word size w is much lesser than the size of h for the scalar k.

6 Conclusions

The efficient implementation of most of the elliptic and hyperelliptic curve cryptographic schemes is based on the efficient computation of kP for a given k, and a point P. We have improved the storage space required during the preprocessing stage of scalar multiplication operation significantly by applying the concept of fixed base comb to the Koblitz's idea of using Frobenius map for



Ffigure : 2 (a)

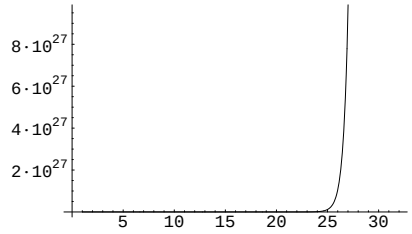


Figure : 2 (b)

Fig. 2. For the case $w < h$, where w is word size, the amount of storage space needed to store $n \times 2^w$ points is much smaller than the amount of storage space needed to store $s \times 3^h + 3^r$. In this case the number of additions slightly increased whereas the order of increase in the number of doubles remains the same to that of case 1.

computing scalar multiplication. By this approach, we obtain the improvement in storage space from $s \times 3^h + 3^r$ to $n \times 2^w$, where $n = s \times h + r$. Because of such a tremendous improvement in storage space, we can easily incorporate our scheme in any hand-held devices where memory is a constraint.

Bibliography

[1] Harald Baier, *Elliptic curves of prime order over optimal extension fields for use in cryptography*, Progress in cryptology - indocrypt 2001, 2001, pp. 99–107.

[2] Daniel V Bailey and Christof Paar, *Optimal extension fields for fast arithmetic in public key algorithms*, Proceedings of advances in cryptology - crypto '98, 1998, pp. 472–485.

[3] Ian Blake, Gadiel Seroussi, and Nigel Smart, *Elliptic curves in cryptography*, Cambridge University Press, 2004.

[4] Y.J. Choie and J.W. Lee, *Speeding up the scalar multiplication in the jacobian of hyperelliptic curves using frobenius map*, Proceedings of progress in cryptology - indocrypt '02, 2002, pp. 285–295.

[5] Mathieu Ciet, Tanja Lange, Francesco Sica, and Jean-Jacques Quisquater, *Improved algorithms for efficient arithmetic on elliptic curves using fast endomorphisms*, Proceedings of advances in cryptology - eurocrypt '03, August 2003May 4, pp. 388–400.

[6] Henri Cohen and Gerhard Frey, *Handbook of elliptic and hyperelliptic curve cryptography*, Chapman and Hall/CRC, 2005.

[7] Andreas Enge, *Elliptic curves and their applications to cryptography- an introduction*, Kluwer Academic Publishers, 2001.

[8] R.P. Gallant, R.J. Lambert, and S.A. Vanstone, *Faster point multiplication on elliptic curves using efficient endomorphisms*, Proceedings of advances in cryptology - crypto '01, 2001, pp. 190–200.

[9] Darrel Hankerson, Alfred Menezes, and Scott Vanstone, *Guide to elliptic curve cryptography*, Springer, 2004.

[10] T. Kobayashi, H. Morita, K. Kobayashi, and F. Hoshino, *Fast elliptic curve algorithm combining frobenius map and table reference to adapt to higher characteristic*, Proceedings of advances in cryptology - eurocrypt '99, 1999, pp. 176–189.

[11] N. Koblitz, *CM curves with good cryptographic properties*, Proceedings of advances in cryptology - crypto '91, 1991, pp. 279–288.

[12] T. Lange, *Efficient arithmetic on hyperelliptic curves*, Ph.D. Thesis, 2001.

[13] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications.*, Cambridge University Press, 1994.

[14] Pradeep Kumar Mishra, *Studies on efficient and secure implementation of elliptic and hyperelliptic curve cryptosystems*, Ph.D. Thesis, 2004.

[15] V. Muller, *Fast multiplication on elliptic curves over small fields of characteristic two.*, Journal of Cryptology **11** (1998), no. 4, 219–234.

[16] Palash Sarkar, Pradeep Kumar Mishra, and Rana Barua, *New table look-up methods for faster frobenius map based scalar multiplication over $GF(p^n)$*, Proceedings of acns, 2004, pp. 479–493.

[17] N.P. Smart, *Elliptic curve cryptosystems over small fields of odd characteristic.*, Journal of Cryptology **12** (1999), no. 2, 141–151.

[18] Jerome A. Solinas, *An improved algorithm on a family of elliptic curves.*, Proceedings of advances in cryptology - crypto '97, 1997, pp. 357–371.

[19] ———, *Efficient arithmetic on koblitz curves*, Designs Codes Cryptography **19** (2000), no. 2/3, 195–249.

[20] Lawrence C. Washington, *Elliptic curves - number theory and cryptography*, Chapman & HALL/CRC, 2003.