Energy-efficient multiplication method for ECC in WSNs

Ravi Kishore Kodali
Department of E and C E
National Institute of Technology,
Warangal, India,
<kishore@nitw.ac.in>

Srikrishna Karanam
Department of E and C E
National Institute of Technology,
Warangal, India

***Abstract*—Wireless sensor networks (WSN's)are highly resource constrained and require energy efficient and strong cryptographic techniques to ensure secure connectivity. Elliptic Curve Cryptography (ECC) is a very popular public key cryptography technique due to the extremely complex and difficult Elliptic Curve Discrete Logarithm Problem (ECDLP), it relies upon. In this paper, a technique for speeding up point multiplication, a fundamental operation involved in ECC, is presented. Specifically, the computation cost of the width−w Non-Adjacent Form (NAF) method of point multiplication is drastically reduced by employing a *mixed* coordinate system to perform the basic point doubling and addition operations. This paper demonstrates that this technique results in a reduction of about 62% in the number of field multiplications required when compared to the traditional techniques for elliptic curve point multiplication.**

***Index Terms*—NAF, ECC, Point Multiplicaton, WSN**

I. Introduction

The advent of the world wide web in the 1990s started bringing the world closer than ever before, creating new avenues for information sharing. The Internet, since then, has been on an expanding spree, and has resulted in the emergence and ubiquitous spread of devices like the Personal Digital Assistants (PDAs). With information sharing hitting unprecedented levels, security of the information being shared becomes paramount. The risks associated with sharing information through the internet, or any local area network, can be mitigated using easy-to-use and inexpensive cryptographic techniques.

With rapid strides being made in the field of networking and information sharing, wireless sensor networks became prominent for their potential applications include earthquake monitoring and intrusion detection among others. These sensor devices interact with each other in a distributed fashion, collecting and processing information from their physical surroundings. However, wireless sensor networks are unlike any traditional networks, or the Internet for that matter and traditional cryptographic algorithms cannot be applied. A major limiting factor of WSNs is that they are extremely energy-constrained and this motivates the need to have energy-efficient cryptographic techniques. The cryptographic standard prevelant today is RSA. However, Elliptic Curve Cryptography (ECC), introduced independently by Koblitz in [9] and Miller in [12] is emerging as a viable alternative to RSA, for ECC provides same level of security as RSA at much lower sizes of the key. ECC, in recent years, has also been standardized by organizations such as the ISO [1], IEEE [14] and NIST [13]. However, ECC is compute-intensive, as will be demonstrated in the subsequent sections. This work focuses focuses on reducing the computational cost of ECC from the viewpoint of wireless sensor networks. Specifically, the computational demands of the point multiplication operation, a basic step involved in encryption and decryption using ECC, is addressed.

II. Elliptic Curve Cryptography

ECC makes use of elliptic curves in which the variables and coefficients are restricted to the elements of a finite field, Z_p, where p is a prime number. For example, in case of coefficients and variables limited to Z_p, the elliptic curve equation used is

$$y^2 = (x^3 + ax + b)$$

The set of all points satisfying this equation for given values of p, a and b is denoted by $E_p(a,b)$. Given two points P and Q in $E_p(a,b)$, it would be useful to calculate $R = P+Q$, whose coordinates are: $x_R = (\lambda^2 - x_P - x_Q)$ and $y_R = (\lambda(x_P - x_R) - y_P)$, where $\lambda = \frac{y_Q - y_P}{x_Q - x_P}$. If $P = Q$, i.e, $R = 2P$, λ is given by: $\lambda = \frac{3x_P^2 + a}{2y_P}$.

A fundamental operation in ECC is the point multiplication. Given an integer k and a point P $\in E_p(a,b)$, the point multiplication operation is defined as $Q = kP$, $Q \in E_p(a,b)$. kP is nothing but repeated addition, *i.e.*, $kP = P+P+P\cdots+P$ (k times). Therefore, point multiplication involves repeated use of point addition and point doubling equations given above. The most popular algorithm for point multiplication, double and add algorithm as given by the algorithm 1.

III. Related Work and our contribution

A number of improved versions of the traditional double and add algorithm have been proposed in the past to mitigate the computational complexity of point multiplication. In the case of an elliptic curve over a prime field, for a given point $P(x,y)$, $-P = (x,(p-y))$. Hence, subtraction of points over a prime field is as efficient as addition. This has motivated the

978-1-4673-5090-7/13/$31.00 ©2013 IEEE
Authorized licensed use limited to: NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL. Downloaded on January 24,2025 at 09:26:58 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 The Double and Add algorithm

Input: k(in its binary form), P
Output Q = kP
A. Q = ϕ (infinity point).
B. for $i = t - 1$ downto 0
Q = 2Q.
if $k_i = 1$, $Q = Q + P$.
C. Return Q

development of a signed digit representation for k, and details can be found in [7]. Solinas [16] proposes a sliding window approach to point multiplication and achieves reasonable efficiency. Huang [8] introduces the idea of fuzzy optimization of the trade-off between window size, as defined in [16], and computational complexity. There have also been instances wherein different radices have been used to represent the scalars. Ciet *et al* [4] introduced the binary/ternary method of scalar representation using radices 2 and 3. Longa *et al* [11] introduced the multi-base non-adjacent form (mbNAF) method, which efficiently represents integers using multiple bases.

As will be seen in the subsequent sections, the width-w NAF representation offers a lower computation cost when compared to the double and add algorithm. However, width-w NAF multiplication requires pre-computatation of $P_i = iP$ for $i = 3, \cdots, 2^w - 1$, and their storage. The pre-computation of these points is performed using the conventional double and add algorithm. Hence, it is essential that efficiency is achieved with these pre-computations in order to speed up point multiplication. In this work, the primary focus is on speeding up the width$-w$ NAF method of point multiplication. This is achieved through the use of different coordinate systems for the point addition and doubling field operations. In addition, the cost requirements of the odd point pre-computation operation are also addressed. The contributions of this work is enumerated below:

1) a mixed coordinate system [5] is used to reduce the computation cost of point multiplication, and
2) a primary problem with using the double and add algorithm for pre-computations is field inversion. This problem is overcome by using the Montgomery trick described in [2], which trades inversions with multiplications. To carry out subsequent point multiplications, the points are represented using the mixed coordinate system to achieve greater efficiency.

IV. Limitations of the Double and Add Algorithm and the NAF representation

The double and add algorithm utilizes the binary representation of k. Assuming a $t-$bit binary representation, the expected number of 1's in the binary representation of k would be $t/2$. This implies that the cost of the double and add algorithm would be t point doublings and $t/2$ point additions. For a large value of k, this complexity is huge.

Algorithm 2 Computing width-w NAF

Input: k
Output: width-w NAF of k
A. $i = 0$.
B. while k ≥ 1 do
if k is odd
$k_i = u$, where $u \equiv k (mod 2^w)$
$k = k - k_i$
else $k_i = 0$.
C. $k = k/2$
D. $i = i + 1$
Return $\{k_{i-1}, k_{i-2}, \cdots, k_1, k_0\}$

This complexity motivates the development of different decimal number representation schemes. The core idea behind these schemes is to minimize the number of 1's in the representation, thereby leading to a reduced number of point addition operations. One such scheme is the signed digit representation, according to which $k = \sum_0^{t-1} k_i 2^i$, where $k_i \in \{0, \pm 1\}$. A particularly useful signed-digit represenation is the Non-Adjacent Form (NAF), which has the property that no two adjacent coefficients k_i are non-zero.

In this case, the average density of non-zero coefficients is $1/3$, thereby reducing the number of point additions to be performed. This computation cost can be further reduced by using a variant of the NAF, called the *width-w* NAF. In a width-w NAF representation, each coefficient k_i is odd and satisfies the following:

- $k_i \leq (2^w - 1)$.
- at most one of any w consequtive coefficients is non-zero.

In this case, the average density of non-zero coefficients is approximately $\frac{1}{w+1}$.

The algorithm for width-w NAF point multiplication is given in algorithm 3 [16].

Algorithm 3 Width-w NAF Point Multiplication

Input: w, width-w NAF of k, P
Output: Q = kP
A. Compute $P_i = iP$ for $i = 1, 3, \cdots, 2^w - 1$.
B. $Q = \phi$.
C. for i = t-1 downto 0 do
$Q = 2Q$.
if $k_i \neq 0$ then
if $k_i > 0$ then $Q = Q + P_{k_i}$.
else $Q = Q - P_{k_i}$.
Return Q.

A. Illustrations

Let us compare the computational cost of the point multiplication operation kP, where k is represented in binary

Authorized licensed use limited to: NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL. Downloaded on January 24,2025 at 09:26:58 UTC from IEEE Xplore. Restrictions apply.

form and in width-w NAF. Let k be equal to 1234567.

*1) **Binary Representation**:* The binary representation of 1234567 is 10010110101101000111. It contains 21 bits, out of which 11 are 1's and the remaing are 0's. Therefore, the computation cost of kP in this case would be 20 point doublings (D) and 10 point additions (A), or $20D + 10A$.

*2) **Width-w NAF**:* The width-4 NAF representation of 1234567 is 9 0 0 0 0 13 0 0 0 0 13 0 0 0 0 0 0 7. Notice the increased number of 0's when compared to its binary representation. In this case, the computational cost would be $18D+4A$, provided $9P, 13P$ and $7P$ are pre-computed. If computational efficiency can be achieved with these pre-computations, a reduction in the computational cost of point multiplication can be obtained over the binary representation.

V. Coordinate Systems and Computation Cost

The coordinate system in which points on elliptic curves are represented also dictates the computational cost of point multiplication. A few coordinate systems and the corresponding computation costs incurred are described in the following subsections.

A. *Affine Coordinates*

In the normal affine coordinates [15], the point addition and doubling ($R = P + Q$) formulae are as given in section II. In this case, the computation cost of one point addition would be: 1 inversion (I), 2 multiplication (M) and 1 squaring (S), or $I + 2M + S$. Note that addition, subtraction and multiplication by a constant are neglected because they are usually faster than point multiplication and inversion. Similarly, the computation cost of one point doubling would be $I + 2M + 2S$.

B. *Projective Coordinates*

In projective coordinates [10], the following substitutions are made: $x = X/Z$ and $y = Y/Z$. The point addition formulae would be: $X_R = vA$, $Y_R = u(v^2X_PZ_Q - A) - v^3Y_PZ_Q$ and $Z_R = v^3Z_PZ_Q$, where $u = Y_QZ_P - Y_PZ_Q$, $v = X_QZ_P - X_PZ_Q$ and $A = u^2Z_PZ_Q - v^3 - 2v^2X_PZ_Q$.

The point doubling formulae would be: $X_R = 2hs$, $Y_R = w(4B - h) - 8Y_P^2s^2$ and $Z_R = 8s^3$, where $w = aZ_P^2 + 3X_P^2$, $s = Y_PZ_P$, $B = X_PY_Ps$ and $h = w^2 - 8B$.

C. *Jacobian Coordinates*

In the Jacobian coordinate system [3], following substitutions are made: $x = X/Z^2$ and $y = Y/Z^2$. The point addition formulae would be: $X_R = -H^3 - 2U_1H^2 + r^2$, $Y_R = -S_1H^3 + r(U_1H^2 - X_R)$ and $Z_R = Z_PZ_QH$, where $U_1 = X_PZ_Q^2 - X_QZ_P^2$, $S_1 = Y_PZ_Q^3$, $S_2 = Y_QZ_P^3$, $H = U_2 - U_1$ and $r = S_2 - S_1$.

The point doubling formulae would be: $X_R = T$, $Y_R = -8Y_1^4 + M(S - T)$ and $Z_R = 2Y_PZ_P$ where $S = 4X_PY_P^2$, $M = 3X_P^2 + aZ_P^4$ and $T = -2S + M^2$.

The computation costs incurred in all these operations are tabulated in Table I.

TABLE I: Point doubling and addition costs in various coordinate systems

Operation	Affine	Projective	Jacobian
Point Addition	$I + 2M + S$	$12M + 2S$	$12M + 4S$
Point Doubling	$I + 2M + 2S$	$7M + 5S$	$4M + 6S$

TABLE II: Computation cost for the illustration of section IV-A

Format	Affine	Projective	Jacobian
Binary	$645M + 50S$	$260M + 120S$	$200M + 160S$
Width-w NAF	$473M + 40S$	$174M + 98S$	$120M + 124S$

D. *Discussion*

On an average, the cost of the inversion operation I in elliptic curve arithmetic would relate to the cost of multiplication operation M as [5]:

$$9M \leq I \leq 30M$$

for prime p larger than 100 bits. Therefore, it would be straightforward to see than projective coordinates offer a drastic reduction in the computation cost of point addition and doubling.

Let us once again consider the illustration given in section IV-A. For $k = 1234567$ in binary form, the cost is $20D + 10A$ which would translate into $30I + 60M + 50S$ in affine coordinates. Substituting $I = 19.5M$ (average), this cost becomes $645M + 50S$. Similarly the cost of width-w NAF is computed in both coordinate systems, and all these results are tabulated in Table II.

As can be observed from Table I, an interesting aspect of the Jacobian coordinate system is that it offers faster doubling than the projective coordinate system. However, the projective coordinate system offers faster addition. Hence, using different coordinate systems for addition and doubling is something which might turn out to be very cost-effective. Therefore, in all our subsequent analysis, different coordinate systems are used for the point addition and doubling operations. This is noted in Table III.

TABLE III: The mixed coordinate system

Operation	Coordinate system
Point addition	Projective
Point doubling	Jacobian

VI. Cost analysis of width-w NAF point multiplication using mixed coordinates

In this section, an analysis of the computational requirements of the width-w NAF point multiplication algorithm

Authorized licensed use limited to: NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL. Downloaded on January 24,2025 at 09:26:58 UTC from IEEE Xplore. Restrictions apply.

using the mixed coordinate system is performed. Assume a t bit width-w NAF representation of the scalar k. As already noted, the average density of non-zero coefficients in the NAF representation of k is $\frac{1}{w+1}$. Therefore, the average number of non-zero coefficients in a $t-$bit NAF representation of k would be $\frac{t}{w+1}$. The computational cost of algorithm 3 would hence be equivalent to t point doublings and $\frac{t}{w+1}$ point additions. However, the use of the mixed coordinate system would result in additional overhead of converting points from one coordinate system to another. The entire cost analysis of algorithm 3 using mixed coordinates is enumerated below:

1. Assuming the original point is input in affine coordinates, it would have to be converted to Jacobian coordinates, for the first step in algorithm 3 is doubling, and according to the mixed coordinate system described above, the Jacobian coordinate system is used for doubling and Projective coordinate system for addition. The cost of this conversion from Affine to Jacobian would be $S + 2M$.
2. Since t point doublings are performed, the cost would be t times the cost of point doubling in Jacobian coordinates, *i.e.*, $t(6S + 4M)$.
3. The points would have to be converted from Jacobian to Projective system $\frac{t}{w+1}$ times, for point addition is performed those many times, as can be observed from algorithm 3. The cost of this conversion would be $\frac{t}{w+1}(I + 2M)$.
4. Next, the point addition operation is performed in the projective coordinate system $\frac{t}{w+1}$ times. In this step, the pre-computed points P_{k_i} would also have to be convereted from affine to projective system. The net cost of this step would be $\frac{t}{w+1}(12M + 2S + 2M)$.
5. Since the iteration from addition to doubling happens $\frac{t}{w+1}$ times, the result of addition, which is in Projective system, would have to be converted into the Jacobian system. The cost of this step would be $\frac{t}{w+1}(2M)$.
6. Finally, the result is converted from Projective system to affine system.

Hence, the total cost C_{mnaf} would add upto

$$C_{mnaf} = (4M + S) + t(6S + 4M) + \frac{t}{w+1}(I + 2S + 18M) \quad (1)$$

Rewriting this equation,

$$C_{mnaf} = I(\frac{t}{w+1}) + M(4 + 4t + \frac{18t}{w+1}) + S(6t + 1 + \frac{2t}{w+1}) \quad (2)$$

A. Evaluations

In this section, the cost of the mixed coordinate based witdh-w NAF point multiplication method is compared with the traditional double and add algorithm. The average cost C_{da} of the traditional double and add algorithm, based on affine coordinates, would be

$$C_{da} = t(I + 2M + 2S) + \frac{t}{2}(I + 2M + S) \quad (3)$$

Rewriting,

$$C_{da} = I(\frac{3t}{2}) + M(3t) + S(\frac{5t}{2}) \quad (4)$$

1) The case of a 160-bit and a 192-bit scalar: Consider a 160-bit and a 192-bit scalar k. It has been pointed out that the NAF representation of k is atmost one longer than the binary representation [7]. Therefore, assuming t to be almost same across both binary and NAF representation, and considering the average case of $I = 19.5M$ as noted earlier, a cost comparison of double and add method and the mixed coordinate based width-w NAF method is performed. The comparison is given in Table IV.

TABLE IV: Computation cost comparison

Bit Width	C_{da}	w	C_{mnaf}
160	$5160M + 400S$	4	$1844M + 1025S$
		5	$1664M + 1014.3S$
		6	$1501.14M + 1006.7S$
		7	$1394M + 1001S$
		8	$1310.67M + 996.5S$
192	$6192M + 480S$	4	$2209M + 1229.8S$
		5	$1972M + 1217S$
		6	$1800.57M + 1207.85S$
		7	$1672M + 1201S$
		8	$1572M + 1195.67S$

VII. Trading Field Inversions with Multiplications - The Montgomery Trick

In the previous sections, it has been seen that a mixed coordinate system can provide best computational efficieny with the point multiplication operation. However, this is just half the problem solved. As established previously, a key step in the performance of the width-w NAF multiplication algorithm is the precomputation of the points $P_i = iP$ for $i = 1, 3, \cdots, 2^w - 1$. In our work, the technique introduced by Dahmen *et al* in [6] is used. This technique determines $(2i - 1)P$ as

$$(2i - 1)P = 2P + (2i - 3)P$$

The formulae for determining these points in affine coordinates are listed below:

$$\begin{aligned} 2P(x_2, y_2): \quad & 3P(x_3, y_3): \\ x_2 = \lambda_1^2 - 2x_1 \quad & x_3 = (\lambda_2^2 - x_2 - x_1) \\ y_2 = \lambda_1(x_1 - x_2) - y_1 \quad & y_3 = (\lambda_2(x_2 - x_3) - y_2) \\ \lambda_1 = \frac{3x_1^2 + a}{2y_1} \quad & \lambda_2 = \frac{y_2 - y_1}{x_2 - x_1} \end{aligned}$$

$$\begin{aligned} (2i - 1)P(x_{i+1}, y_{i+1}): x_{i+1} &= (\lambda_i^2 - x_2 - x_i) \\ y_{i+1} &= (\lambda_i(x_2 - x_{i+1}) - y_2) \\ \lambda_i &= \frac{y_i - y_2}{x_i - x_2} \end{aligned}$$

As can be seen in the formulae listed above, the most critical component of these pre-computations is field inversion, *i.e.*, determining the inverses of the denominators $den_1 = 2y_1$, $den_2 = (x_2 - x_1)$ and in general, $den_i = (x_i - x_2)$. It is, however,

Authorized licensed use limited to: NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL. Downloaded on January 24,2025 at 09:26:58 UTC from IEEE Xplore. Restrictions apply.

possible to compute all these den_i simultaneously using the Montgomery trick [2].

The cost $C_{precomp}$ of precomputing the points $P_i = iP$ for $i = 3, \cdots, 2^w - 1$ by using Dahmen's method in conjunction with the Montgomery's trick is given by

$$C_{precomp} = (10 * 2^{w-1} - 11)M + 2^{w+1}S + I \quad (5)$$

An extensive mathematical proof of equation 5 can be found in [6]. Table V shows the cost ($I = 19.5M$ as before) for various values of w.

TABLE V: $C_{precomp}$ versus w

w	$C_{precomp}$
4	$98.5M + 32S$
5	$168.5M + 64S$
6	$328.5M + 128S$
7	$648.5M + 256S$
8	$1288.5M + 512S$

A. Total Cost Analysis

This $C_{precomp}$ has to be added to C_{mnaf} to determine the total cost of the mixed coordinate based width-w NAF method of point multiplication. A total cost comparison of this method with the traditional double and add algorithm is given in Table VI.

TABLE VI: Total computation cost comparison - 192-bit k

C_{da}	w	C_{mnaf}
$6192M + 480S$	4	$2307.5M + 1261.8S$
	5	$2140.5M + 1281S$
	6	$2129.07M + 1335.85S$
	7	$2320.5M + 1457S$
	8	$2860.5M + 1707.67S$

However, it is to be noted that this pre-computation has to be performed only once for a given w, and hence the total cost given in Table VI is only one time cost. For any subsequent calculations, the cost would be as per table IV, assuming that w remains the same.

VIII. Results and Conclusions

As can be seen from Table II, the width-w NAF method of point multiplication offers superior performance when compared to the traditional double and add method. In addition, as can be seen from Table IV, Table V and Table VI, the use of a mixed coordinate system in the width-w NAF greatly enhances the efficiency of the point multiplication operation. Specifically, let us focus on the number of field multiplication operations, since the amount of resources one field multiplication operation requires is generally much higher when compared to a field squaring operation. Referring to Table VI, in the case of $w = 6$, the number of field multiplications required by the mixed-coordinate based width$-w$ NAF algorithm is approximately 2129 whereas the double and add algorithm requires 6192 field multiplications. This represents a reduction of about 66% in the number of field multiplication operations. A Table representing this cost reduction for various values of the window parameter w is given below.

TABLE VII: Table showing the reduction in the number of field multiplication operations required

w	Double and add	Mixed Coordinate width-w NAF	Reduction
4	6192	2307.5	62%
5	6192	2140.5	65%
6	6192	2129.07	66%
7	6192	2320.5	62%
8	6192	2860.5	54%

As can be seen from Table VII, an average reduction of about 62% in the number of field multiplications required has been achieved by employing the mixed coordinate system in the width$-w$ NAF method of point multiplication.

References

[1] ISO/IEC 14888-3. Information Technology - Security Techniques - Digital Signatures with Appendix - Part 3: Certificate Based-Mechanisms, 1998.

[2] R.M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography*. Chapman and Hall/CRC, 2005.

[3] D.V. Chudnovsky and G.V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Advances in Applied Math.*, 7:385–434, 1986.

[4] M. Ciet, M. Joye, K. Lauter, and P.L. Montgomery. Trading inversions for multiplications in elliptic curve cryptography. *Designs, Codes and Cryptography*, 39:189–206, 2006.

[5] H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In *Advances in Cryptology âĂŤ ASIACRYPTâĂŹ98*, volume 1514 of *Lecture Notes in Computer Science*, pages 51–65. Springer Berlin / Heidelberg, 1998.

[6] E. Dahmen, K. Okeya, and D. Schepers. Affine precomputation with sole inversion in elliptic curve cryptography. In *Proceedings of the 12th Australasian conference on Information security and privacy*, ACISP'07, pages 245–258. Springer-Verlag, 2007.

[7] D. Hankerson, J.L. Hernandez, and A. Menezes. Software implementation of elliptic curve cryptography over binary fields. In *CHES, LNCS 1965*, pages 1–24, 2000.

[8] X. Huang, D. Sharma, and H. Cui. Fuzzy controlling window for elliptic curve cryptography in wireless sensor networks. In *ICOIN*, 2012.

[9] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[10] K. Koyama and Y. Tsuruoka. Speeding up elliptic cryptosystems by using a signed binary window method. In *Advances in Cryptology-Proceedings of CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 345–357. Springer-Verlag, 1993.

[11] P. Longa and A. Miri. New multibase non-adjacent form scalar multiplication and its application to elliptic curve cryptosystems. In *âĂć*, number 052 in Cryptology e-print Archive. 2008.

[12] V.S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology*, volume 218, pages 417–426, 1986.

[13] National Institute of Standards and Technology. Digital signature standard.

[14] IEEE P1363. Standard Specifications for Public-Key Cryptography, 2000.

[15] J.H. Silvermann. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer, ii edition, 2009.

[16] J. Solinas. Efficient arithmetic on koblitz curves. *Designs, Codes and Cryptography*, 19:195–249, 2000.

Authorized licensed use limited to: NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL. Downloaded on January 24,2025 at 09:26:58 UTC from IEEE Xplore. Restrictions apply.