Contemporary Engineering Sciences, Vol. 9, 2016, no. 18, 897 - 904
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2016.6690

Reduced Precomputed Scalar Multiplication Cost for ECC

Gautam Kumar

Computer Science & Engineering
Jaypee University of Information Technology
Waknaghat, Solan, H.P-173234 India

Hemraj Saini

Computer Science & Engineering
Jaypee University of Information Technology
Waknaghat, Solan, H.P-173234 India

Abstract

Elliptic curve cryptography (ECC) is one of the most fascinating areas in cryptography that provides a higher level of security on smaller key sizes. The paper proposes the optimal scalar multiplication cost without pre-computed operations on Radix-16. It is one of the most optimized methods that best suits to low memory devices on the reduced complexity. Further, in relation to the previous proposed work, it is computing 6.25 percent faster on the software performance perspective and 8.33 percent faster on the hardware performance perspective.

Keywords: Precomputation, ECC, Radix-16, DLP, Non-fifteen Encoding

1 Introduction

Cryptography is a based on mathematical conceptualization from the discipline of computer science, which is responsible for security strength on all applications. Security algorithms are the strength of any protocol which is based

on the (initial) precomputation cost for all algorithms. The idea of public key cryptography (PKC) was first proposed in 1976 by Diffie and Hellman [1]. In PKC's various algorithms have been proposed, but in all of them Elliptic Curve Cryptography (ECC) has been attracted the most attention due to higher computational speedup on the comparatively very shorter key sizes. From the recent research trends and which has discovered, its complexity is considering still in excess and/or improvements in the same is the motivational issue for any problem [2]. In research, various provisions with exceptionally enhanced system security services need for its performances.

A heart of cryptography is Discrete Logarithmic Problem (DLP), which is acting as a central role in information security. Information security is an attractive area that is based on high computational speed at a lower cost for efficient algorithms. The fast running algorithms are leading high performance and high speed. When DLP applies on the ECC is known to ECC-DLP. Its computation is based on elliptic points P and Q, to find the value of k (generally secret key), on $\mathrm{Q} = \mathrm{kP}$, which is the core building block in PKC [3].

The basic algorithm of ECC is based on the scalar multiplication which consists of repeated point addition (ADD) and point doubling (DBL) operations. For one point ADDs and for one point DBLs operation the required pre-computed operations are $13{,}617$ and $14{,}000$ clock cycles [4].

Reduction in the precomputed operation is a research gap. Here is a brief idea about the existing algorithms has presented for input secret key for the working environment. This works on a secret key (scalar) k, represented in the form of m bit binary field, the algorithm of Most Significant Bit (MSB) first [5], requires m bits doubling (DBLs) and on average $\mathrm{m}/2$ bits of addition (ADDs) operations. Similarly, for Least Significant Bit (LSB) first [5]-[6] requires on average $\mathrm{m}/2$ bits of ADDs and same bits of DBLs. A Montgomery Ladder method keeps k DBLs & k ADDs, which is higher precomputed operations than the previous two but additional advantages is in favor of Side Channel Attack (SCA) [7]-[8]. A non-adjacent form (NAF) [9], [10] is another variation of algorithm representation in $\{-1, 0, 1\}$, on average $\mathrm{m}/2$ bits of ADDS and $\mathrm{m}/2$ bits of DBLs, but in addition to this, it is resistant to the side-channel attacks [10]. The complexity of w-NAF [11] keeps $\mathrm{m}/(\mathrm{w}+1)$ in point ADDs only. A variation of w-NAF [12] does in sliding w-NAF, also known by Frobenious operations, that escapes the series of zeros during the scalar multiplication, which counts the enhancement in scalar multiplication. Abdulrahman and Masoleh in [13] have been given a methodology for resistance to the SSCAs, which have been proposed for Radix-8 scalar multiplications is the most optimized algorithm on complexity $log_8(k+1)$ without precomputed operation, reported in 2015. Table 1 consist the basic ECC algorithm that exists and its related complexities have presented which is showing the reduction in the precomputed operation and/or security strengths.

Table 1: Existing approaches and its complexities

Algorithm	**Complexity**
Most Significant Bit	k DBLs & k/2 ADDs
Least Significant Bit	k/2 DBLs & k/2 ADDs
Montgomery Method	k DBLs & k ADDs + Side Channel Attack
Non-Adjacent Form (NAF)	k/2 DBLs & k/2 ADDs + Side Channel Attack
Window methods	k/(w+1) ADDs
Sliding window method	Escaping series of zero's on k/(w+1) ADD
Radix-8 without Pre-Computations	$\log_8(k+1)$ without (ADDs & DBLs)

Our manuscript organization is as follows. Section 2 presents the proposed Radix-16 Scalar Multiplication through the basic assumption for Radix generalization principles, methodologies on non-fifteen encoding representation algorithm and algorithms for Radix-16. Section 3 presents a validation approach through numerical example. Section 4 presents the prospective benefit of this approach on software and hardware performance enhancements. Finally, our manuscript has summarized in section 5.

2 Proposed Radix-16 Algorithm

2.1 Basic Assumption for Radix- Generalization Technique

We have just considered the Lemma 1 [13] proposed for the radix-expansion technique that has applied recently on Radix-8 scalar multiplication. Now we are extending the same on reduced computation cost, where its complexity is better than all the existing approaches. For the reduced instruction set computing (RISC) architecture this is one of the appropriate radix. The various reasons are available to set up this problem as a novel contributions such as Hexadecimal converts easily to binary which uses the computer at the fundamental level, offers faster computation and requirements of less memory, code size and memory size execution are smaller, feasibility of parallelism much lower in arithmetic computation [14], storage of more information in the same space, has reported to reach a better performance used in cryptosystem and the most important to implement the same with the short memory devices [15].

The realization is based on the three initial registers such as $P_{kP} = 0$, $P_1 = P$, $P_{ACC} = P$ where P is an initial point on the curve, P_{kP} is initial assumption scalar multiplication assumed to be zero and two additional registers have considered as P_1 and P_{ACC} assigned as elliptic point P from the curve. The scalar multiplication is based on encoding representation according on chosen basis\radix. The Lemma 1, has been presented below for any key lengths with its place value on r basis for P_{kP}, P_1 and P_{ACC}.

Lemma 1. The range $[1,\ l-1]$ is a key length as $k_j^{'}$ in between $0 \leq k_j^{'} \leq r-1$, then scalar multiplication P_{kP}^{j}, defines in:

$$P_{kP}^{j} = \begin{cases} P_{kP}^{j-1} + k_j' P_{ACC}^{j} \\ rP_{ACC}^{j} - P_1^{j} \end{cases} \quad (1)$$

Whereas P_1^j defines as follows:

$$P_1^{j} = \begin{cases} rP_{ACC}^{j} - P_{kP}^{j} \\ (r-1-k_j')P_{ACC}^{j} + P_1^{j-1}, \text{ where } P_{kP}^{j} = r^j P = rP_{ACC}^{j}. \end{cases} \quad (2)$$

The $P_{ACC} = rP$ is always calls for each radix set values.

2.2 Non-fifteen Encoding Representation

The pre-assumption for secret key k assumes in decimal, first this one converted into hexadecimal. After obtaining the hexadecimal add '0' as signed value next to the most significant position. Further, read the hexadecimal including '0' from least significant bit first to most significant bit and converted the same in the form of Non-fifteen encoding representation. This says if any sets values contain 15 or 'F', it replaces the same by -1 and add '1' as a carry to the next position value. The algorithm 1 is a representing the basic idea for Non-fifteen encoding method, which has presented below.

Algorithm 1: Non-Fifteen Encoding Method

Input: A $k-1$ digit Radix-16 of the scalar k, $k = (k'_{t-2}, \ldots, k'_1, k'_0)_{16}, k'_j \in \{0,1, \ldots, 15\}$
Output: $k = (k_{t-1}, \ldots, k_1, k_0)_{16}, k_j \in \{-1,0,1, \ldots, 14\}$
Initialize: $k = (0, k'_{t-2}, \ldots, k'_1, k'_0)_{16}$;
Step 1: For j=0 to t-1 do
Step 1.1: If $k'_j \in \{15\}$ then
Step 1.1.1: $k_j = k'_1 - 15, k'_{j+1} = k'_{j+1} + 1$;
Step 1.2: Else Leave the digit as it is, i.e., $k_j = k'_j$;
Step 2: End For
Step 3: Return $k = (k_{t-1}, \ldots, k_1, k_0)_{16}$;

2.3 Radix-16 Scalar Multiplication Algorithm

The output from the Non-fifteen encoding of length t applies for Elliptic Curve scalar multiplication $Q = kP$. The Non-fifteen encoding contains a range of $k_j \in \{-1,0,1, \ldots, 14\}$. The initial assumptions is based on three registers, such

as $P_{kP} = 0$, $P_1 = P$, $P_{ACC} = P$, as the following steps progress through the algorithm 2.

The discrete logarithmic problem is almost negligible to revert back on the original key in computation.

Algorithm 2: Radix-16 Algorithm

Input: Point $P \in E(F_q)$, At digit of integer k, i.e., $k = (k_{t-1}, \dots, k_1, k_0)_{16}, k_j \in \{-1,0,1, \dots ,14\}$
Output: Point $Q = kP$.
Initialize: $P_{kP} = 0, P_1 = P$, $P_{Acc} = P$;
Step 1: For j=0 to t-1 do
  Step 1.1: If $k_j \in \{-1,0,1,2,4,7,8,9,10,12\}$ then
    Step 1.1.1: $P_{kP} = P_{kP} + k_j P_{ACC}$;
    Step 1.1.2: $P_{ACC} = 16P_{ACC}$;
    Step 1.1.3: $P_1 = P_{ACC} - P_{kP}$;
  Step 1.2: If $k_j \in \{3,5,6,11,13,14\}$ then
    Step 1.2.1: $P_1 = P_1 + (15 - k_j)P_{ACC}$
    Step 1.2.2: $P_{ACC} = 16P_{ACC}$;
    Step 1.2.3: $P_{kP} = P_{ACC} - P_1$;
Step 2: End For
Step 3: Return P_{kP};

3 Validation of Proposed Problem through Numerical Example

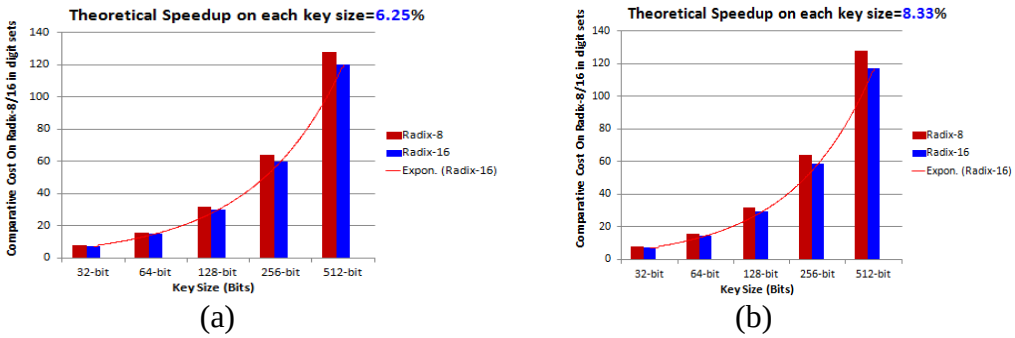
The input secret scalar k assumes in decimal, converted into the form of hexadecimal and in addition one extra bit 0 adjoins as signed value. The non-fifteen encoding algorithm says if any digit contain F or 15 it replaces the same by -1, and add one as a carry to the next, otherwise its values only records as it passes as described in algorithm 1. Further, the computation starts with an initial assumption on three registers such as $P_{kP} = 0$, $P_1 = P$, $P_{ACC} = P$, as the steps progress through the algorithm 2. The final result of scalar multiplication P_{kP} returns as output.

We clarify the same by an example, where secret scalar $k = 89982$ and its hexadecimal is $(15F7E)_{16}$. It is represented in the form of non-fifteen encoding representation as $(016\bar{1}7E)$. Table 2 demonstrates the computing process of proposed signed radix-16 scalar multiplication.

Table 2: Radix-16 Numerical Example for Scalar Multiplication

Digit Set		$a[k_j] \in \{-1,0,1,2,4, 7,8,9,10,12\}$	$a[k_j] \in \{3,5,6,11, 13,14\}$
Initialization		$P_{kP} = 0;\ P_1 = P;\ P_{Acc} = P$	
(Iteration from Right-to-Left)	$k_j = E$		$P_1 = 2P$ $P_{Acc} = 16P$ $P_{kP} = 14P$
	$k_j = 7$	$P_{kP} = 126P$ $P_{Acc} = 256P$ $P_1 = 130P$	
	$k_j = \bar{\bar{1}}$	$P_{kP} = -130P$ $P_{Acc} = 4096P$ $P_1 = 4226P$	
	$k_j = 6$		$P_1 = 41090P$ $P_{Acc} = 65536P$ $P_{kP} = 24446P$
	$k_j = 1$	$P_{kP} = 89982P$ $P_{Acc} = 1048576P$ $P_1 = 958594P$	
	$k_j = 0$	$P_{kP} = 89973P$ $P_{Acc} = 16777216P$ $P_1 = 16687234P$	

It computes scalar multiplication for any elliptic point P on the curve. From the computational point of view, the computation cost t at radix-16 for scalar k, we can define its cost $t = \lceil log_{16}k \rceil + 1$, whereas for radix-8 its cost is $t = \lceil log_8 k \rceil + 1$.



(a) (b)

Figure 1. The performance optimization has shown on behalf of computational complexity of the proposed algorithm for each key size from the two perspectives. These are listed as: (**a**) Software performance enhancement; (**b**) Hardware performance enhancement

4 Performance Comparisons on Software and Hardware

From the theoretical point of view on each key size, uses by ECC for scalar multiplication, gets a bigger marginal enhancement. From the software perspective

the proposed algorithm at Radix-16 has accelerated by 6.25 percent compared to the previous proposed radix-8 scalar multiplication on the complexity basis, as represented in Figure 1 (a). Further, from a hardware performance perspective, scalar multiplication has upgraded on computational enhancement from radix-8 to radix-16 is leading by 8.33 percent improvement, as represented in Figure 1 (b).

5 Conclusion

The manuscript consist the importance of precomputed operations with the existing approaches for Elliptic Curve Cryptography. On the same precomputed technique, we optimized the existing approach of Radix-8 scalar multiplication for better performance on the reduced computation cost on Radix-16 scalar multiplication. This is showing the theoretical computational enhancement on software and hardware performance enhancements with the respective differences are 6.25 and 8.33 from an existing Radix-8 scalar multiplication technique. The reason to covering this problem is to obtain a lot of special benefits regarding the performance optimization and future implementable applications for short-memory device on reduced cost.

Acknowledgements. This work is supported by Jaypee University of Information Technology, H.P. (INDIA)

References

[1] W. Diffie and M.E. Hellman, New Directions in Cryptography, *IEEE Transaction on Information Theory*, **22** (1976), 644-654. http://dx.doi.org/10.1109/tit.1976.1055638

[2] K. Jarvinen and J. Skytta, Parallelization of High-Speed processor for Elliptic Curve Cryptography, *IEEE Transaction on VLSI*, **16** (2008), 1162-1175. http://dx.doi.org/10.1109/tvlsi.2008.2000728

[3] N. Koblitz, Elliptic Curve Cryptosystems, *Math Computation*, **48** (1987), 203-209. http://dx.doi.org/10.1090/s0025-5718-1987-0866109-5

[4] C.H. Gebotys, *Security in Embedded Devices*, Springer, London, 2010. http://dx.doi.org/10.1007/978-1-4419-1530-6

[5] V.S. Miller, Use of Elliptic Curves in Cryptography, Chapter in *Advances in Cryptology*, 1986, 417-426. http://dx.doi.org/10.1007/3-540-39799-x_31

[6] D.R. Hankerson, A Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, New York, 2004. http://dx.doi.org/10.1007/b97644

[7] K. Okeya, H. Kurumatani, and K. Sakurai, Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications, Chapter in *Public Key Cryptography*, 2000, 238-257. http://dx.doi.org/10.1007/978-3-540-46588-1_17

[8] M. Joye and S. M. Yen, The Montgomery Powering Ladder, Chapter in *Cryptographic Hardware Embedded Systems*, Springer-Verlag, 2003, 291-302. http://dx.doi.org/10.1007/3-540-36400-5_22

[9] G. Avoine, J. Monnerat and T. Peyrin. Advances in Alternative Non-adjacent Form Representations, Chapter in *Progress in Cryptology-INDOCRYPT*, 2004, 260-274. http://dx.doi.org/10.1007/978-3-540-30556-9_21

[10] I.F. Blake, V. Kumar Murty and G. Xu, A note on window τ-NAF algorithm, *Information Processing Letters*, **95** (2005), 496-502. http://dx.doi.org/10.1016/j.ipl.2005.05.013

[11] C. Heuberger and H. Pondinger, Analysis of Alternatives Digits Sets for Non-Adjacent Representation, *SIAM Journal on Discrete Mathematics*, **19** (2006), 165–191.

[12] R M Avanzi, C Heuberger, H Podinger, On Redundant τ-adic Expansions and Non-Adjacent Digit Sets, Chapter in *Selected Areas in Cryptography*, Springer-Verlag, 2007, 285-301. http://dx.doi.org/10.1137/s0895480103437651

[13] E.A.H. Abdulrahman and A. R-Masoleh, New Regular Radix-8 Scheme for Elliptic Curve Scalar Multiplication without Pre-Computation, *IEEE Transaction on Computers*, **64** (2015), 438-451. http://dx.doi.org/10.1109/tc.2013.213

[14] P. K. Mishra, Pipelined Computation of Scalar Multiplication in Elliptic Curve Cryptosystems (Extended Version), *IEEE Transaction on Computers*, **55** (2006), 1000–1010. http://dx.doi.org/10.1109/tc.2006.129

[15] C. Vuillaume, K. Okeya and T. Takagi, Short-Memory Scalar Multiplication for Koblitz Curve, *IEEE Transaction on Computers*, **57** (2008), 481-489. http://dx.doi.org/10.1109/tc.2007.70824

Received: June 24, 2016; Published: August 12, 2016