Journal of Theoretical and Applied Information Technology
10th May 2015. Vol.75. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

X-TRACT RECODING ALGORITHM FOR MINIMAL HAMMING WEIGHT DIGIT SET CONVERSION

[1]MIZA MUMTAZ AHMAD, [2]SHARIFAH MD YASIN, [3]RAMLAN MAHMOD, [4]MOHAMAD AFENDEE MOHAMED

[1]Lecturer, School of Mathematical Sciences, Universiti Kebangsaan Malaysia

[2]Senior Lecturer, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia

[3]Profesor., Faculty of Computer Science and Information Technology, Universiti Putra Malaysia

[4]Senior Lecturer, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia

E-mail: [1]mmumtaz@ukm.my, [2]ifah@upm.edu.my, [3]ramlan@fsktm.upm.edu.my, [4]afendee@upm.edu.my

ABSTRACT

Scalar multiplication is the most computational intensive operation in elliptic curve cryptosystem (ECC). Improving the performance of this operation can be achieved by using recoding technique which aims at minimizing the density of nonzero digits in the scalar, also known as its Hamming weight. We proposed X-Tract recoding technique that alters the representation of scalar from binary digit to {-1,0,1,3} digit set in non-adjacent form (NAF). The recoding algorithm can be expressed both mathematically and logically and reduces the Hamming weight of the scalar to 41% in average for 163-bit scalar with balanced bit. X-Tract Recoding algorithm reads every three adjacent bits overlapped by one bit from left-to-right in the input binary scalar to produce each new output digit. We use ANOVA analysis to show that the new recoding technique produces significantly better output with less Hamming weight compared to its counterparts.

Keywords: *Elliptic Curve Cryptosystem, Scalar Multiplication, Recoding, Hamming Weight, Non-Adjacent Form*

1. INTRODUCTION

Elliptic curve cryptosystem (ECC) application was introduced independently by Koblitz [11] and Miller [12] in mid 1980s. The main operation in ECC is the scalar multiplication $Q = kP$ where k is a scalar, P and Q are points on an elliptic curve. The scalar multiplication is defined as the addition of point P on the elliptic curve with itself for k times such that:

$$kP = \underbrace{P + P + \ldots + P}_{\text{k times}} \qquad (1)$$

Technically, this process is the most time consuming and expensive in elliptic curve arithmetic [2], hence the motivation behind many researches to improve its performance. The scalars used for arithmetic calculation in computing systems are originally represented as binary strings. Sometimes it is more advantageous to represent a binary scalar in other digit representation such as in the calculation of scalar multiplication in elliptic curve cryptosystem (ECC). There are two main point operations in elliptic curve scalar multiplication that are directly related to the representation of k. These operations are called point doubling and point addition. The quantities of point doubling and point addition operations are directly proportional to the length and the density of non-zero digits in the scalar string respectively. Thus, changing the representation of k will greatly affect the performance of scalar multiplication algorithm. The number of non-zero digits in the scalar is also known as the Hamming weight of the scalar. The lesser is the Hamming weight, the lesser algebraic operations will it take in the scalar multiplication [1].

Distinctly, the best scalar representation of k for most efficient scalar multiplication would be the one that has the shortest string and least Hamming weight. It is impossible to attain a scalar representation that has both of the mentioned qualities thus priority should be given to the better quality over the other. In the case of elliptic curve arithmetic, since point addition is much more expensive than point doubling, more emphasis is

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

given to lessen the number of non-zero digits compared to shortening the scalar string.

To reduce the Hamming weight in the string, recoding techniques are introduced. The performance of recoding techniques is related to the appearance of the scalar string. For instance, Reitwiesner's method [6] works best with binary strings that contains consecutive 1s while {0,1,3}-NAF recoding [7] works best with string that contains balanced bits. Since the two types of scalar strings are the most used in ECC implementation, we combined the attributes of both of the aforementioned recoding techniques into our X-Tract recoding technique that scans every 3 bits from left to right to produce every new bit in the new string. In the average case of 163-bit string with balanced bit, our technique reduces the number of non-zero bits to 41%. The algorithm can be expressed both mathematically and logically. We use ANOVA analysis to show that the new recoding technique produces significantly lesser Hamming weight scalars compared to the implementation of [6] and [7].

The rest of the paper is organized as follows. In Section 2, we briefly describe the basics of digit set conversion with some examples. Then we present some of the existing scalar recoding techniques in Section 3. In Section 4, we explain our proposed X-Tract Recoding Technique in term of its algorithm and characteristics. We show that our algorithm is more efficient compared to its counterparts via ANOVA analysis in Section 5. Finally, we conclude this paper in Section 6.

2. PRELIMINARIES

2.1 Definitions

Following [1], a binary scalar, k can be written as:

$$k = k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \cdots + k_0 = \sum_{i=0}^{n-1} k_i 2^i \quad (2)$$

With $k_{n-1} = 1$ and $k_i \in \{0,1\}$. Also, the digit set conversion is the mapping $F: X^n \rightarrow Y^m$, such that if Y=F(X), then $\|Y\| = \|X\|$. While a fixed radix conversion is a digit set conversion $F: X^n \rightarrow Y^m$ with $r_x = r_y = r$.

The number of non-zero digits in the scalar string is the Hamming weight of that string. For instance, if X = 100011010, then the Hamming Weight of X is $Hw_x = 4$.

2.2 Examples

Consider converting a decimal based scalar into a 4-digit representation (n = 4) with digit set X = {-1, 0, 1} and radix, $r_x = 2$. Then a scalar from 0 to 15 can be represented as X = (a, b, c, d) where the value of the scalar is calculated as below:

$$\|X\| = (a \times 2^3) + (b \times 2^2) + (c \times 2^1) + (d \times 2^0) \quad (3)$$

Thus, 7 can be represented as $X = (1,0,0,-1)$, where:

$$\|X\| = (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + ((-1) \times 2^0)$$
$$\|X\| = 8 + 0 + 0 + (-1) = 7$$

Below we provide two examples of scalar multiplication kP with $k = 42\,527$. In the first example, k is represented as a binary digit such that $k = 1010011000011111$. Therefore the scalar multiplication is:

kP = (2 (2 (2 (2 (2^5 (2 (2^3 (2^{2}P + P) +P) +P) +P) +P) +P) +P) +P)
= 42 527 P

In the second example, k is represented as {-1, 0, 1}-digit set, such that k = 101010-100010000-1. Therefore the scalar multiplication of kP is:

kP = (2^5 (2^4 (2^2 (2^2 (2^{2}P + P) + P) – P) + P) – P)
= 42,527 P

3. RELATED WORK

To the best of our knowledge, the earliest idea to recode binary scalar into other digit representation was published in 1951 [4]. This recoding technique converts scalar from binary representation to {-1,0,1} digit representation by reading every two adjacent bits overlapped by one, from left to right. The recoding can be expressed by a look up table as described in Table 1. Booth recoding may or may not reduce the number of non-zero digits in the scalar string. To illustrate this, we provide an example for the case of X = 110110101 that converts to Y = 10-110-11-1. As in the example given, since $Hw_x < Hw_y$, the recoding is not minimal.

Table 1: Booth Recoding

x_{i+1}	x_i	y_i
0	0	0
0	1	1
1	0	-1
1	1	0

Journal of Theoretical and Applied Information Technology
10th May 2015. Vol.75. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

More attempts to modify Booth recoding were published afterwards such as in [5] that recodes binary scalars to {-2,-1,0,1,2} digit representation strings. However, none of the modifications is successful in producing neither minimal nor unique scalar recoding.

In 1960, Reitwiesner [6] introduced minimal binary recoding algorithm that produces NAF output in {-1,0,1} representation. There are two versions of recoding introduced by Reitwiesner. The first is a right-to-left conversion that reads 2 adjacent bits from overlapped by one. This version can be expressed both by algorithm and look-up table as in Table 2. Since the bits are read from right-to-left, it requires extra memory for output storage.

Table 2: Minimal Binary Recoding

x_i	x_{i+1}	x_i	c_{i+1}	y_i
0	0	-	0	0
0	1	0	0	1
0	1	1	1	-1
1	0	0	0	1
1	0	1	1	-1
1	1	-	1	0

Hence, [13] introduced the left-to-right Reitwiesner recoding algorithm but without the look-up table. This is a minimal recoding that is especially effective for scalar that contains long string of 1s such as X = 11111111 that can be converted into Y = 10000000-1. However, the recoding is least effective for cases where the 1s appears in pairs in the scalar string. To illustrate this, take X as X = 01100110. After conversion, Y will become Y = 10-1010-10, thus the number of non-zero digits remain the same such that Hw_x = Hw_y.

In 2011, [7] proposed converting scalar from binary digit set into {0,1,3} digit set via {0,1,3}-NAF recoding technique. This recoding reads every three adjacent binary bits from left-to-right overlapped by one bit. Contrary to Reitwiesner's method, {0,1,3}-NAF recoding is most effective with the type of scalar that contains lots '11' pairs in the string. This is because the Hamming weight of the pairs is reduced to half via conversion of '11' into '03'. For instance, $X = 11100110$ will be converted to $Y = 10300030$. This usually happens a lot in scalars that contain almost identical number of 1s and 0s, known as the balanced bit scalars. In contrast, [7] is not as effective as Reitwiesner's method in reducing the Hamming weight of string of 1s, such that using {0,1,3}-NAF method, X = 11111111111 will become Y = 030303030303 after conversion.

There are also other types of recoding techniques such as the Double Base Number System (DBNS) recoding. As the name implies, the scalar is represented as a sum of base 2 and 3 number [8], such that:

$$k = \sum_{i=1}^{m} s_i 2^{b} 3^{t},\ s_i \in \{-1,1\},\ b_i, t_i \geq 0 \quad (3)$$

The DBNS technique is applied together with scalar multiplication and the conversion rate is relatively slow and requires extra memory. Window method [10] also requires extra memory in its execution. In this method, w consecutive binary bit representation of k is scanned and replaced by pre-computed table value.

4. PROPOSED X-TRACT SCALAR RECODING

There are two outstanding recoding techniques based on the different form of the scalar string. While Reitwiener's recoding works best with strings that contain lengthy consecutive 1s, {0,1,3}-NAF recoding works best in string with balanced bit distribution. In X-Tract recoding technique, we combined these two sought after qualities into a simple algorithm that can also be expressed in 8 × 7 look-up table.

Algorithm 4.1: X-Tract Recoding Algorithm

Input: $X = x_{n-1} \dots x_1 x_0 \in \{0,1\}$
Output: $Y = y_n \dots y_1 y_0 \in \{-1,0,1,3\}$
Process:

1. Let $c_0 = 0$
2. Pad X with 000 such that $000X$
3. For $i = 0,1,\dots,n$:
   a) $\alpha = 4(x_{i+2})(x_{i+1})(x_i)$,
   $y_i = x_i + 2(x_{i+1})(x_i) - \alpha +$
   b) $c_i^2[1 - (2x_i + x_{i+1}) + \alpha]$

Journal of Theoretical and Applied Information Technology
10th May 2015. Vol.75. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

c) $c_{i+1} = \frac{x_i - y_i + c}{2}$

The algorithm of X-Tract recoding technique is as in Algorithm 4.1 while the look-up table is as described in Table 3. X-Tract recoding reads every 3 adjacent bits from right-to-left overlapped by one bit. Initially, the original scalar bit need to be padded with 3 zeros at the front before the scalar recoding takes place. For example, before converting $X = 11100110$ into {-1,0,1,3}-NAF using X-Tract recoding, we pad X such that X = 00011100110. Then using either the algorithm or look-up table, the output after the conversion will be Y = 100-100030.

Table 3: X-Tract Recoding

c_i	b_{i+2}	b_{i+1}	b_i	b'_i	c_{i+1}
0	-	-	0	0	0
0	-	0	1	1	0
0	0	1	1	3	-1
0	1	1	1	-1	1
1	-	-	1	0	1
1	-	0	0	1	0
1	-	1	0	-1	1
-1	-	0	1	0	0

We compare the performance between the two types of implementations of X-Tract recoding technique, i.e, the algorithm version and the look-up table version. The result is as displayed in Table 4.

Table 4 : Memory consumption and running speed of X-Tract recoding for all 16-bit scalars

X-Tract recoding version	Look-up Table version	Algorithm version
Commit	2928	1880
Working Set	2976	2496
Sharable	1024	592
Private	1952	1904
Running Time	366 µsec	409 µsec

5. ANOVA ANALYSIS ON PERFORMANCE

To prove that X-Tract recoding is better than Reitwiesner's recoding and {0,1,3}-NAF recoding, we compared the performance among the recoding techniques. The summary of performance for 16-bit scalar is as displayed in Table 5.

Based on Table 5, we see that {0,1,3}-NAF recoding gives better result for balance bit scalars while Reitwiesner is better for high density scalars. However, X-Tract recoding gives the best result for both cases. We further proceed with ANOVA analysis to confirm that the great performance of X-Tract scalar recoding is just not mere coincidence.

Table 5 Hamming weight difference for different recoding techniques

Average Hw Reduction Rate	[6]	[7]	X-Tract
Balance bit scalars (50% of input string contain 1s)	18.99%	29.59%	30.88%
High density scalars (> 80% of input string contain 1s)	66.18%	43.25%	66.60%

5.1 Hypothesis Testing

We want to test if there is significance difference in the Hamming weight of recoded scalars from X-Tract scalar recoding algorithm compared to the other two scalar recoding algorithms, i.e. Reitwiesner's algorithm and {0, 1, 3}-NAF algorithm. The mean of these three algorithms are represented as μ_X, μ_R, and μ_N respectively. The null hypothesis H_0 and the alternative hypothesis H_1 are formulated as below:

$$H_0: \mu_X = \mu_R = \mu_N$$
$$H_1: \mu_X \neq \mu_R \neq \mu_N$$

Rejecting the null hypothesis indicates that there is a significant difference in the Hamming Weight of our X-Tract Scalar Recoding Algorithm as compared its counterparts.

5.2 Variables & Measurement Scale

In our experiment, the independent variable is the original scalar before recoding. While the dependent variable is the recoded scalar's Hamming weight. The measurement scale used for this experiment is ordinal.

5.3 Subject Selection

A total of 300 16-bit binary scalars with Hamming weight of 8 are used as sample for this hypothesis testing. We choose all scalars with balanced bit since this is the type of scalar that is commonly used in cryptographic practice.

5.4 Experimental Design

The experiment involves one factor with three treatments. The factor in this experiment is the scalar recoding algorithm with X-Tract algorithm,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Reitwiesner's algorithm and {0, 1, 3}-NAF algorithm as treatments. The 300 samples are systematically selected and divided into three sets of 100 replications with each set tested on distinct algorithm.

5.5 Experimental Instrument

The tests is run using MatLab programming language on an Intel® Core™ i7-3537U CPU @ 2.00GHz. The inputs are pre-recorded beforehand. Output from each test will be directly written and saved to Excel file for further analysis.

5.6 Handling Validity Threats

The large number of focused sample of 300 inputs reduces internal validity. External validity is negligible since data and experiments are done digitally. We have ensured the validity of all scalar recoding algorithms being tested by checking the correctness of outputs from the inputs. In addition, X-Tract Scalar Recoding Algorithm has been mathematically proven to be correct.

5.7 Result

We obtain the following ANOVA table from the experiment result.

Source	DF	Sum of Squares	Mean Square	F
Treatments	2	50.1667	25.0833	3.03
Error	297	199.5	0.6717	
Total	299	249.667		

Since $f^* = 37.3421 > F_{(2;\ 297;\ 0.05)}$, H_0 is rejected. We conclude that there is a significant different between the three types of scalar recoding algorithms.

6. DISCUSSION

In software implementation, the look-up table version is faster but uses more memory than the algorithm version as presented in Table 4. Thus depending on the type of application, either software or hardware, one version may be more advantageous than the other. In fact, the look-up table version may be more suitable for hardware implementation while the algorithm version is more suitable for software implementation.

From experiment with all 16-bit scalars, we found that the mathematical form perform faster, making it more suitable for software implementation while the logical version requires less memory storage, making it more suitable for hardware implementation.

7. CONCLUSION

The recoding of binary scalar has many implications towards improving the performance of arithmetic in computing system in term of execution time and memory usage.

In this paper we showed that X-Tract recoding greatly reduces the Hamming weight of input string both in extreme cases and average cases. This recoding can be represented both logically and mathematically. For the logical or look-up table version, less memory is used thus making it more suitable for hardware implementation. While the mathematical or formula version is faster, making it more suitable for software implementation.

For future work, we will optimize the look-up table version so that its execution will be faster and require less memory usage compared to the formula version. Also, come up with the left-to-right recoding version.

REFERENCES:

[1] B. Phillips, and N. Burgess, "Minimal Weight Digit Set Conversion", *IEEE Transactions on Computers,* Vol. 53, No. 6, 2004, pp. 666-677.

[2] F.A. Diego, F.-H. Armando, L. Julio, and R.-H. Francisco, "Faster Implementation of Scalar Multiplication on Koblitz Curves", 2012, Retrieved December 2014. *http://eprint.iacr.org/2012/519.pdf*

[3] B. Elaine, B. William, P. William, and S. Miles, "Recommendation for Key Management – Part 1: General (Revision 3)", *NIST Special Publication,* Vol. 800, No. 57, 2012, pp. 64.

[4] A.D. Booth, "A Signed Binary Multiplication Technique", *Quarterly J. Mechanics and Applied Math.* Vol. 4, 1951, pp. 236-240.

[5] O.L. MacSorley, "High-Speed Arithmetic in Binary Computers", *Proc. IRE.,* Vol. 49, 1961, pp. 91-103.

[6] G.W. Reitwiesner, "Binary Arithmetic", *Advances in Computers,* Academic Press, Vol. 1, 1960, pp. 231-308.

[7] S.M. Yasin "New Signed-Digit {0,1,3}-NAF Scalar Multiplication Algorithm for Elliptic Curve over Binary Field." *PhD Thesis, Universiti Putra Malaysia*, 2011.

[8] V. Dimitrov, L. Imbert, and P. Mishra, "Efficient and Secure Elliptic Curve Point

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Multiplication using Double - Base Chains." *Advances in Cryptology – ASIACRYPT'05*, LNCS, Vol. 3788, 2005, pp. 59–78.

[9] M. Subhashis, and S. Amitabha, "A New Algorithm for Computing Triple-Base Number System", *ACM SIGARCH Computer Architecture News*. Vol. 40, No. 4, 2012, pp. 3-9.

[10] P. Longa, "Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields". *Master Thesis, University of Ottawa, Canada*, 2007.

[11] N. Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation*, Vol. 48, No.177, 1987, pp. 203–209.

[12] V. Miller, "Use of Elliptic Curves in Cryptography", CRYPTO, *Lecture Notes in Computer Science*, Vol. 85, 1985, pp. 417–426.

[13] M. Joye, and S. Yen, "Optimal Left-To-Right Signed-Digit Recoding", *IEEE Transactions on Computers,* Vol. 49, No. 7, 2000, pp. 740-748.