Original Research Article

Efficient Elliptic Curve Arithmetic for Lightweight Cryptographic schemes for IoT Applications

ABSTRACT

The Internet of Things' (IoT) market is expected to grow exponentially at the global level in the coming years, due to the proliferation of more reliable and faster networks resulting from the extensive rollout of 5 to 10 G mobile networks. By 2025, it is expected that worldwide projection of IoT connected devices will be pegged at 30.9 billion units. Despite the potential benefits of the new technology, security in IoT, which borders on protecting stored data and data in transit against eavesdropping, redirection or illegal altering, is a major threat. According to HP, 70% of IoT devices are vulnerable to sniffing attacks and reliable solution is yet to be found. The standard cryptographic algorithms such as RSA and AES provide good security but their utilization in IoT is questionably due to hardware and energy constraints for computationally expensive encryption schemes. However, elliptic curve-based cryptography, a recent paradigm in public key cryptography, achieves the same level of security with smaller key sizes. On the other hand, the total score of performance of an elliptic curve-based cryptosystem depends largely on the efficiency of the arithmetic operations performed in it. It is against this background that this paper proposes efficient elliptic curve arithmetic for implementing ECC based schemes which will fit into IoT systems implementations. Elliptic curve point arithmetic implementations in projective coordinate systems over binary extension fields introduce higher efficiencies in software. In this regard, this paper has proposed an improved López-Dahab point arithmetic methods on non-supersingular elliptic curves over $GF(2^m)$. The results show 69.20% improvement in Point Doubling, 44.68% in Point Addition and the scalar point multiplication execution time is decreased by 48.80%.

Keywords: ECC, Security, Galois Fields, Field arithmetic, Point arithmetic, ECSM, Projective coordinate

1. INTRODUCTION

The Internet of Things (IoT) is considered the most recent upgrade of the existing Internet and various Information and Communication Technology (ICT) tools. It envisions a very near future of total connectivity, interaction and ubiquity among objects of everyday life, industrial equipment, vehicles and practically all physical devices hence the term 'Things' [1].

Currently, security is a major threat to the IoT vision. IoT security borders on protecting stored data and data in transit against eavesdropping, redirection or illegal altering during devices' communications. According to HP, 70% of IoT devices are vulnerable to sniffing attacks and reliable solution is yet to be found [2].

Elliptic curve-based cryptographic schemes were first proposed as the foundation for security framework in Internet of Things and Cloud Computing [3]. In another study on lightweight cryptographic algorithms suitable for data security and authentication in IoT, [4] noted that the standard cryptographic algorithms such as RSA and AES provide good security but their utilization in IoT is questionably due to hardware and energy constraints for computationally expensive encryption schemes whilst ECC achieves the same level of security with smaller key sizes (Table 1).

Table 1 Comparison between RSA and ECC

Algorithm	**Key Size (bits)**	**Key Generation (s)**	**Signature Generation (s)**	**Signature Verification (s)**	**Merits**
RSA	1024	0.16	0.01	0.01	Increased security
	15360	679.06	9.20	0.03	
ECC	163	0.08	0.15	0.23	Increased speed, less memory requirement, optimum security
	571	1.44	3.07	4.53	

On the other hand, the total score of performance of an elliptic curve based cryptosystem depends largely on the efficiency of the arithmetic operations performed in it [5]. It is against this background that this paper proposes efficient elliptic curves arithmetic for implementing elliptic curve based schemes.

2. ARITHMETIC OPERATIONS IN ELLIPTIC CURVE CRYPTOGRAPHY

The building blocks of any public key cryptoscheme are its arithmetic operations. ECC arithmetics are put into three groups which include Scalar arithmetic, Point arithmetic and Field arithmetic. Point arithmetic operations include Point Addition and Point Doubling whilst addition, subtraction, multiplication, squaring and inversion in the underlying field constitute the field arithmetic operations.

The two major fields used in cryptography are prime fields $\mathbb{F}_p$ and binary extension fields, denoted $\mathbb{F}_{p^m}$ or $GF(2^m)$ for some integer $m > 1$. The latter introduce higher efficiencies as compared to the other fields in software implementation [6]. In this regard, efficient methods for the arithmetic involved in implementing elliptic curves over binary extension fields $GF(2^m)$ whereby the order of the curves can be up to m bits is of great importance.

2.1 Representation of elements in $GF(2^m)$

Elements of $GF(2^m)$ are represented in many basis. However the major ones used in cryptography are the Normal Basis (NB), Polynomial Basis (PB) and Redundant Basis (RB), according to [7]. A very remarkable observation is that the choice of the basis by which field elements are represented has a major effect on the implementation of the finite field operations. For example, in software implementations of cryptographic schemes, the use of Polynomial Basis yields better results than the Normal Basis [8]. However, when it comes to hardware implementations, for some $GF(2^m)$ operations, using the Normal Basis is more suitable [9].

In the Polynomial Basis, the elements of $GF(2^m)$ are binary polynomials and at most $m-1$ degree. Now let a basis element $\beta \in GF(2^m)$ be a root of an m-degree irreducible polynomial $P(x)$, the set of powers of the basis element $\beta = \{\beta^0, \beta^1, \cdots, \beta^{m-2}, \beta^{m-1}\}$ define a Polynomial Basis for $GF(2^m)$ [7]. Any field element $A \in GF(2^m)$ can therefore be uniquely expressed as

$$A = a_0 + a_1\beta + a_2\beta^2 + \cdots + a_{m-1}\beta^{m-1} \quad (1)$$

$$= \sum_{i=0}^{m-1} a_i \beta^i, a_i \in GF(2)$$

2.2 Addition (subtraction) of $GF(2^m)$ elements in Polynomial Basis

$$Let\ X = \sum_{i=0}^{m-1} x_i\beta^i, x_i \in GF(2)\ and\ Y = \sum_{i=0}^{m-1} y_i\beta^i, y_i \in GF(2) \quad (2)$$

The additions is given as

$$X + Y = \sum_{i=0}^{m-1} ((x_i + y_i) \bmod 2)\beta^i \quad (3)$$

A careful observation is that Equation (3) is reduced to simply XORing $X\ and\ Y$.

2.3 Multiplication of $GF(2^m)$ elements in Polynomial Basis

Let a, $b \in GF(2^n)$ be represented as polynomials

$$a(x) = \sum_{i=0}^{n-1} a_i x^i \text{ and}$$

$$b(x) = \sum_{i=0}^{n-1} b_i x^i \quad (4)$$

where $a_i, b_i \in GF(2)$ or equivalently represented as binary vectors

and let

$$p(x) = x^n + r(x)$$

be an irreducible polynomial of degree n over GF(2)
the extension field multiplication of $a(x).b(x)$ is given as

$$c(x) = a(x).b(x) = \left(\sum_{i=0}^{n-1} a_i x^i\right).\left(\sum_{j=0}^{n-1} b_j x^j\right) \quad (5)$$

$$\equiv \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left(a_i.b_j \ mod\, p(x)\right) .\ x^{(i+j)}$$

$$= \sum_{k=0}^{2n-2} c_k x^k$$

There are a number of ways of implementing Equation (5). A few of the most recommended approaches are presented here.

The standard Shift-and-Add method [10] computes $a(x)b(x)(mod\ (p(x))$. This is based on the fact that

$$a(x)b(x) = a_{n-1}x^{n-1}b(x) + \cdots + a_1 xb(x) + a_0 b(x) \quad (6)$$

and in that regard, the method computes (7)

$$x^i b(x) \text{ modulo } p(x) \ \forall\, 1 \leq i \leq n-1$$

and add all the results for which a_i = 1.

The Comba's *Right to left* and the *Left to Right* methods (Algorithm 1 and

Algorithm **2**) that can be interleaved with a polynomial reduction algorithm (Algorithm 3) are some improved versions of the Standard *Shift and Add*.

Algorithm 1: Right-Left Comba's Method for Polynomial Multiplication

---

INPUT: Binary polynomials a(x) and b(x) of degree at most $m - 1$
OUTPUT: c(x) = a(x). b(x)$of at\ most\ 2m - 2\ degree$
1. $C \leftarrow 0$
2 $for\ k$ = 0 $to\ w - 1\ do$
2.1 for j = 0 to t − 1 do
if the $k^{th} bit\ of\ A[j]$ = 1 $then$
$C\{j\} \leftarrow C\{j\} + B$
2.2 if k ≠ w − 1 then
B ← B. x
3. $Return\ (C)$

Algorithm 2: Left-to-Right Comba's Method for Polynomial Multiplication

INPUT: Binary polynomials a(x) and b(x) of degree at most $m-1$
OUTPUT: c(x) = a(x).b(x)$of at most\ 2m-2\ degree$
1. $C \leftarrow 0$
2 $for\ k = w-1\ \ downto\ 0\ do$
2.1 for j = 0 to t − 1 do
if the $k^{th} bit\ of\ A[j] = 1\ then$
$C\{j\} \leftarrow C\{j\} + B$
2.2 if k ≠ 0 then
C ← C.x
3. $Return\ (C)$

Algorithm 3: Bit by bit Polynomial Modular Reduction

INPUT: Binary polynomial c(x) of degree at most $2m-2$, p(x)
OUTPUT: c(x) (mod p(x))
1. $Precompute\ u_k(x) = x^k r(x)\ for\ all\ x^m r(x), 0 \le k \le w-1$
2. $for\ i = 2m-2\ \ downto\ m\ do$
2.1 $if\ c_i = 1\ then$
$j = \frac{i-m}{w}$
$k = (i-m) - wj$
$C\{j\} \leftarrow u_k(x)$
3. $Return\ \big((C[t-1], \dots C[1], C[0])\big)$

In [11], the polynomial representation was used to develop another method of multiplying two elements in GF(2^m). It was a variant of the original Montgomery's method for modular multiplication of integers.

2.4 Field Arithmetic Inversion

Inversion are the most expensive operations among the field arithmetics useful to point arithmetics in elliptic curves cryptography. The multiplicative inverse of an element $a \in GF(2^m)$ is that element a^{-1}such that

$$a.a^{-1} = 1 \tag{8}$$

A number of algorithms for both $\mathbb{F}_p$ and $GF(2^m)$ identified in the literature are based on the Euclidean algorithm and the Fermat's Little Theorem (FLT). A premier algorithm using Normal Basis found in [11] remains the generic. It was extended to Polynomial Basis by [12].

2.5 Point arithmetic on elliptic curves

Elliptic curves over binary extension fields are two types: Supersingular and Non-supersingular as defined in Equation (9) and Equation (10) respectively. However, for cryptographic applications, recommendations for the use of only non-supersingular curves have come up strongly [13, 14].

Let q = $GF(2^m)$,

A supersingular elliptic curve over q is

$$E(q): y^2 + xy = x^3 + ax + b \quad (9)$$

where $a, b \in q$ and $\Delta = -a^3$ *is the discriminant*.

A non-supersingular elliptic curve over q is

$$E(q): y^2 + xy = x^3 + ax^2 + b \quad (10)$$

where $a, b \in GF(2^m)$ and $b \neq 0$ *is the discriminant*

The set of points on E with coordinates in $GF(2^m)$ is the set

$$E(q) = \{(x, y): x, y \in GF(2^m)\ satisfying\ y^2 + xy = x^3 + ax + b \cup \{\mathcal{O}\}\ \} \text{ and} \quad (12)$$

$$E(q) = \{(x, y): x, y \in GF(2^m)\ satisfying\ y^2 + xy = x^3 + ax^2 + b \cup \{\mathcal{O}\}\ \} \quad (11)$$

for supersingular and non-supersingular curves respectively.

For a non-supersingular elliptic curve E defined over GF(2^m) in affine coordinates as in equation (10), Point Addition and Point Doubling operations are generally computed as follows [13]

i) Point Addition

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2) \in E, P \neq Q$ and Q $\neq$ −P,
then P + Q = (x_3, y_3) *where*

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \quad (13)$$

with

$$\lambda = \frac{y_1 + y_2}{x_1 + x_2}$$

ii) Point Doubling

Let $P = Q = (x_1, y_1)$ then $P + Q = P + P = 2P = (x_3, y_3)$ *where*

$$x_3 = \lambda^2 + \lambda + a = x_1^2 + \frac{b}{x_1^2} \quad (14)$$
$$y_3 = x_1^2 + \lambda x_3 + x_3$$
$$with\ \lambda = x_1 + \frac{y_1}{x_1}$$

A repeated addition is represented as multiplication of a point by an integer. For example, the addition of n times of P as illustrated bellow is considered n multiplied by P.

$$\underbrace{P + P + P + \cdots + P}_{\text{n copies}} = nP \quad (15)$$

In this regard, Point Doubling is a special addition instance where P = Q and as such P+Q = P+P, thus reducing the Point Doubling operation to addition of a point to itself. P+P is also denoted as 2P.

2.5.1 Projective coordinates

The formulas for both Point Addition and Point Doubling require one field inversion each and field multiplications in either case. The cost of field addition and squaring can be ignored according to the works of [14]. Inversion in GF(2^m) is very expensive as compared relatively to multiplication. As a result, alternative point representations, called projective cordinates, in which the point arithmetics are performed without inversions are the most prefered. Several types of projective coordinates for the non-supersingular elliptic curves exist. The three most recommended are:

1. **The Standard Projective Coordinates**

In the *Standard* Projective Coordinates system, the projective point $(X:Y:Z)$, $Z \neq 0$ corresponds to the point $(X/Z, Y/Z)$ in the affine system. The corresponding equation of the elliptic curve is presented as

$$Y^2Z + XYZ = X^3 + aX^2Z + bZ^3 \quad (16)$$

2. **The Jacobian Projective Coordinates**

The projective point $(X:Y:Z), Z \neq 0$ is mapped to the point $(X/Z^2, Y/Z^3)$ in the affine coordinates. The projective equation is presented as

$$Y^2Z + XYZ = X^3 + aX^2Z^2 + bZ^6 \quad (17)$$

3. **The López-Dahab (LD) Projective Coordinates**

In this coordinate system, the projective point $(X:Y:Z), Z \neq 0$ corresponds to the affine coordinate point $(X/Z, Y/Z^2)$ whilst its projective equation is

$$Y^2Z + XYZ = X^3Z + aX^2Z^2 + bZ^4 \quad (18)$$

Table 2 has a summary of computational cost analyses of the point operations in the three projective coordinate systems and the affine coordinate [8]. The counts of field multiplications (M) and field inversions (I) are measured in each system.

Table 2: Field arithmetic operations count in point arithmetic on non-supersingular curves

Coordinate system	General addition	General addition (mixed)	Doubling
Affine	1 I + 2M	-	1 I + 2M
Standard projective	13M	12M	7M
Jacobian projective	14M	10M	5M
López-Dahab projective	14M	8M	4M

2.6 Scalar Arithmetic

The scalar arithmetic, which involves the multiplication of a scalar *k* by a point *P,* denoted kP, is considered the most dominant and time consuming operation, estimated to take about 80% of the total execution time of any elliptic curve cryptographic scheme [15]. Over the years, a myriad of proposed techniques to provide efficient implementations of the scalar multiplication have emerged in the literature. A good survey on kP is found in [16].

2.7 Summary of findings from the literature

The elliptic curve scalar multiplication, being the most expensive operation in the overall elliptic curve based encryption/decryption, relies on the point arithmetic, which in turn, relies on the underlying field arithmetics. Surveys on various ECSM indicate that the field inversion is very expensive as compared relatively to multiplication and squaring, and it is responsible for the high computational overhead in ECSM [16, 17, 18, 19] . The cost of field addition and squaring can be ignored [14]. [5] did a computational cost analysis of these operations over GF(2^m) and established that the cost of field addition is negligible; field inversion is equivalent to ten times field multiplication; and field squaring is approximately 0.2 times field multiplication.

In terms of implementation, field addition (subtraction) of two elements $a, b \in GF(2^m)$ can be achieved simply by a bitwise XORing of a and b whilst multiplication and squaring using Polynomial Basis are quite simple in software implementation.

3. PROPOSED EFFICIENT ARITHMETICS FOR IMPLEMENTATION

The summary of findings from the literature suggests that, for efficient implementation of ECC, the scalar point multiplication methods devoid of field inversions will be lighter and yield faster speed encryptions/decryptions, and as such very suitable for real-time IoT applications. In this regard, this paper proposes as follows:

3.1 Proposed Point Arithmetic

From the data in Table 2, one can appreciate that Point Addition and Point Doubling methods in the López-Dahab projective coordinate system offer the minimum field multiplications count and they do not require the computational expensive field inversions, and for that matter, the best alternative. However, a critical analyses of the addition and multiplication formulas and algorithms, as they are presented in [20], reveals that, for software implementation, more improvements can be done to further improve performance of the López-Dahab methods as follows:

3.1.1 Introduction of concurrency.

The algorithms are executed in a step-by-step approach even if a previous step does not have dependencies in the preceding steps. Introducing multi-threading into these methods, otherwise termed concurrent processing, will be apt, in order to allow non-dependent segments to execute at the same time. This is expected to reduce the execution time although the field operations' counts will remain the same. In this regard, concurrent execution is proposed in Algorithm 4 and Algorithm 5.

A caveat to this proposal is that some IoT devices may not have multicore technology. Notwithstanding, the issues on architecture dependencies, massive scaling and design challenges of IoT based Industrial applications, in addition to security, are still in contention [21]. Moreover, multicore processors are becoming mass products and as such it is not farfetched to conceive the notion that the upcoming IoT devices will be multicore compliant.

Algorithm 4: Multi-Thread López-Dahab Point Doubling

Input: $P = (X_1 : Y_1 : Z_1)$ *in LD Coordinates on* $E: y^2 + xy = x^3 + ax^2 + b$
Output: $2P = (X_3 : Y_3 : Z_3)$ *in LD Coordinates*

1. $if\ P = \infty\ then\ return\ (\infty)$
2. $\boldsymbol{Thread1}\{\ Z_3 \leftarrow X_1^2 Z_1^2\ \}$ *// **end of thread 1***
3. ***Thread2*** $\{X_3 \leftarrow\ X_1^4 + bZ_1^4\ \}$ *// end of **Thread2***

Thread3 Starts

4. $if\ a = 1\ then\ T_1 \leftarrow aX_1^2 Z_1^2 + Y_1^2$
5. $T_1 \leftarrow\ aX_1^2 Z_1^2 + Y_1^2 + bZ_1^4$
6. $Y_3 \leftarrow (X_1^4 + bZ_1^4).T_1$
7. $Y_3 \leftarrow Y_3 + bZ_1^4.X_1^2 Z_1^2\ \}$

Thread3 ends

8. ***Wait for multiple threads***

$Return\ (X_3 : Y_3 : Z_3)$

Algorithm 5: Multi-Thread López-Dahab Point Addition

Input: $P_1 = (X_1 : Y_1 : Z_1)$ in LD projective coord, $P_2 = (x_2, y_2)$ in affine coordinate system on $E: y^2 + xy = x^3 + ax^2 + b$
Output: $P_1 + P_2 = (X_3 : Y_3 : Z_3)$ in LD coordinates

1. if $P_2 = \infty$ then return (P_1)
2. if $P_1 = \infty$ then return $(x_2 : y_2 : 1)$
3. $T_1 \leftarrow\ Z_1.x_2$
4. $T_2 \leftarrow Z_1^2$
5. $X_3 \leftarrow X_1 + X_2 Z_1$
6. $T_1 \leftarrow Z_1.X_3$
7. $T_3 \leftarrow T_2.y_2$
8. $Y_3 \leftarrow Y_1 + T_3$
9. If $X_3 = 0$ then
   9.1 if $Y_3 = 0$ then use Point doubling algorithm for computing $(X_3 : Y_3 : Z_3) = 2(x_2 : y_2 : 1)$ *and* return $(X_3 : Y_3 : Z_3)$
   9.2 else return(∞)
10. $T_4 \leftarrow T_1^2$

Thread1 Starts

11. $Z_3 \leftarrow T_4$
12. $T_3 \leftarrow T_1.Y_3$
13. *If a =1 then* $T_1 \leftarrow T_1 + T_2$
14. $T_2 \leftarrow X_3^2$
15. $X_3 \leftarrow T_2.T_1$
16. $X_3 \leftarrow X_3 + Y_3^2$
17. $X_3 \leftarrow X_3 + T_3$
18. $T_2 \leftarrow x_2 T_4 + X_3$

Thread1 Ends
Thread2 Starts

19. $T_1 \leftarrow T_4^2$
20. $T_3 \leftarrow T_3 + T_4$
21. $T_5 \leftarrow x_2 + y_2$
22. $T_6 \leftarrow T_1.T_5$

Thread2 Ends

23. *Wait for multiple threads*
24. $Y_3 \leftarrow T_3.T_2$
25. $Y_3 \leftarrow Y_3 + T_6$
26. Return $(X_3 : Y_3 : Z_3)$

3.2 Proposed Scalar Point Multiplication

From the literature, two main efficient approaches are used: binary methods and Windows based methods. Except a very few but with more computational overheads, majority of the methods are based on the binary expansion of the scalar

$$k = (k_{m-1}, k_{m-2}, \cdots, k_0)$$

and perform either Point Doubling and/or Point Addition depending on the value of k_i. In this regard, this paper proposes the interleaving of any efficient method with Algorithms 4 and 5. An example is presented in Algorithm 6

Algorithm 6: Left to right Operand Scanning Method for kP using Improved López-Dahab Point Arithmetic

Input: Integer $k = (k_{n-1}, k_{n-2}, \cdots k_1, k_0)$, Point P ∈ E($\mathbb{F}_{2^n}$)
Output: Point *R* = *kP*.

$R \leftarrow \infty$ // Initialization of R

1. **For** i = n – 1 **downto** 0 **do**
   2.1 $R \leftarrow 2R$ // PDBL using Algorithm 4
   2.2 **if** k_i = 1 **then** $R \leftarrow R + P$ //PADD using Algorithm 5
3. Return (R)

4. KEY PERFORMANCE ANALYSIS

In order to measure the performance of the proposed Multi-Thread López-Dahab (MTLD) Point Addition and Point Doubling, the original López-Dahab algorithms and the proposed were implemented in Borland Delphi on Intel Core i3 running Windows 10. Comparing the proposed to the original López-Dahab algorithms, a remarkable decrease in the running time was observed as presented in Table 3 and Figure 1. Point Doubling has seen 69.20% improvement whilst the Point Addition is improved by 44.68%. The scalar point multiplication execution time is decreased by 48.80%. These findings prove that the proposed MTLD is yet a further improvement of the most efficient point and scalar ECC arithmetic, a result that is much desired for IoT real-time applications.

Table 3: Running time (in µs) comparison between LD and proposed MTLD

Operation/ Method	López-Dahab	MTLD	Improvement (%)
Point Doubling	940	289	69.20
Point Addition	1860	1029	44.68
Scalar Point Multiplication	2930	1500	48.80

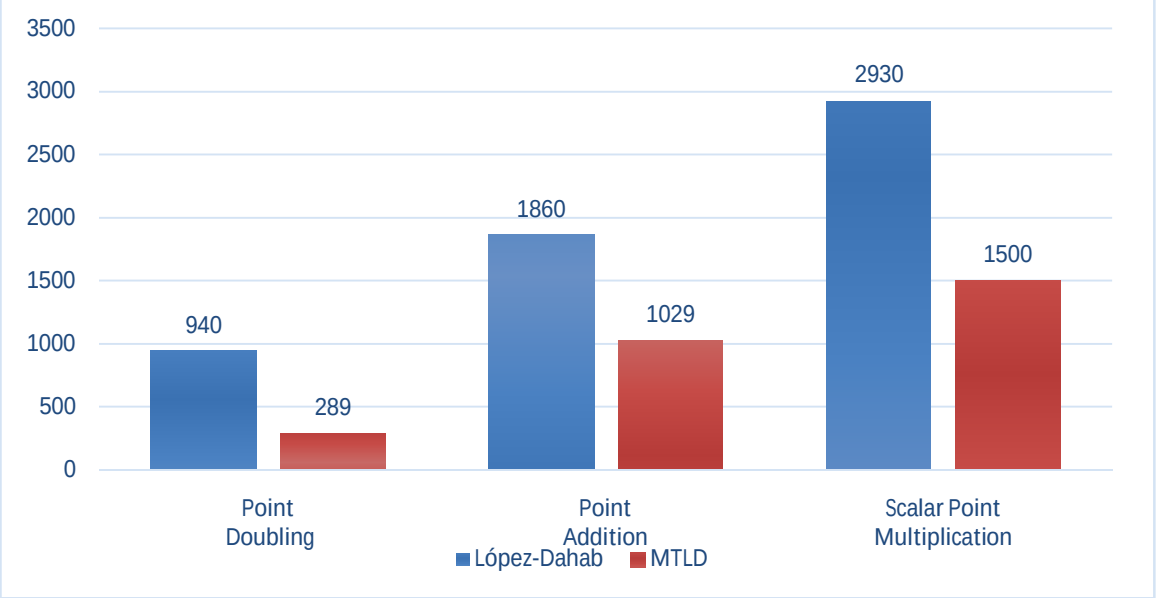

Figure 1: Running time (in µs) comparison between LD and proposed MTLD

5. CONCLUSION

In this work, a review of existing field arithmetic methods for binary extension fields and recommendations for implementing elliptic curves based lightweight cryptoschemes are done. A proposal is made for incorporating multi-threading into the López-Dahab projective coordinates point arithmetic methods, currently being the most efficient for $GF(2^m)$. A prototype implementation was done for performance analysis and the results have shown that the proposed methods yield close to 50% improvement of the López-Dahab for the ECSM.

REFERENCES

[1] J. Holdowsky, M. Mahto, M. E. Raynor and M. Cotteleer, "Inside the Internet of Things (IoT)," 2015. [Online].

[2] S. A. Kumar, T. Vealey and H. Srivastava, "Security in Internet of Things: Challenges, Solutions and Future Directions," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, Koloa, HI, USA, 2016.

[3] D. P. Bai T, A. S. Rabara and V. Jerald, "Elliptic Curve Cryptography based Security Framework for Internet of Things and Cloud Computing," *International Journal of Computer Science and Technology [IJCST]. 6,* pp. 223-229, 2015.

[4] I. Bhardwaj, A. Kumar and M. Bansal, "A review on lightweight cryptography algorithms for data security and authentication in IoTs," in *4th IEEE International Conference on Signal Processing, Computing and Control (ISPCC)*, Solan, 2017.

[5] H. Houssain, Elliptic curve cryptography algorithms resistant against power analysis attacks on resource constrained devices, Clermont-Ferrand II:

Université Blaise Pascal, 2012.

[6] P. Hosseinzadeh Namin, Efficient Implementation of Finite Field Multipliers over Binary Extension Fields, Electronic Theses and Dissertations. 5828. retreved from https://scholar.uwindsor.ca/etd/5828, 2016.

[7] R. Lidl and H. Niederreiter, Introduction to Finite Fields and Their Applications, NY, USA: Cambridge University Press, second ed, 1997.

[8] D. Hankerson, J. L. Hernandez and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields," *Koç, Ç.K., Paar, C. (eds) Cryptographic Hardware and Embedded Systems --- CHES 2000 Lecture Notes in Computer Science, vol 1965. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-44499-8_1,* pp. 1-23, 2000.

[9] T. F. Al-Somani and A. Amin, "Hardware Implementations of GF (2^m) Arithmetic Using Normal Basis," *Journal of Applied Sciences, Vol 6,* pp. 1362-1372, 2006.

[10] J. Guajardo, S. S. Kumar, C. Paar and J. Pelzl, "Efficient Software-Implementation of Finite Fields with Applications to Cryptography," *Appl. Math. vol 93(1),* pp. 3-32, 2006.

[11] C. K. Koc and T. Acar, "Montgomery multiplication in GF(2^k)," *Design, Codes and Cryptography, vol. 14(1),* p. pp. 57–69, 1998.

[12] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF(2m) using normal bases," *Information and Computation, available at http://www.sciencedirect.com/science/article/pii/0890540188900247,* pp. Pages 171-177, 1988.

[13] B. Rashidi, R. R. Farashahi and S. M. Sayedi, "High-performance and high-speed implementation of polynomial basis Itoh–Tsujii inversion algorithm over GF(2m)," *IET Information Security, vol. 11, no. 2,* pp. 66-77, 2017.

[14] D. Hankerson, A. Menezes and S. Vanstone, Guide to Elliptic Curve Cryptography, New York, USA: Springer-Verlag, 2004.

[15] NIST, "Digital Signature Standard (DSS)," *Federal Information Processing Standards Publication, FIPS PUB 186-4,* pp. 87-101, 2019.

[16] Y. R. Venturini, "Performance analysis of parallel modular multiplication algorithms for ECC in mobile devices," *Revista de Sistemas de Informaçao da FSMA, No. 13,* pp. 57-67, 2014.

[17] M. Rasmi, A. A. Sokhon, M. S. Daoud and H. Al-Mimi, "A Survey on Single Scalar Point Multiplication Algorithms forElliptic Curves over Prime Fields," *IOSR Journal of Computer Engineering (IOSR-JCE), Volume 18, Issue 2, Ver. V,* pp. 31-47, 2016.

[18] M. Rivain, "Fast and Regular Algorithms for Scalar Multiplication over Elliptic Curves," 2011. [Online]. Available: https://eprint.iacr.org/2011/338.pdf.

[19] E. Karthikeyan, "Survey of Elliptic Curve Scalar Multiplication Algorithms," *Int. J. Advanced Networking and Applications, Vol 04, Issue no. 02,* pp. 1581-

1590, 2012.

[20] D. Pamula, Arithmetic operators on GF(2^m) for cryptographic applications: performance-power consumption - security tradeoffs, Computer Arithmetic. Université Rennes 1, HAL Open Science, 2012.

[21] J. López and R. Dahab, "Improved algorithms for elliptic curve arithmetic in GF(2n)"," *S. Tavares and H. Meijer (Eds.): Selected Areas in Cryptography SAC '98, LNCS 1556*, pp. 201-212, 1999.

[22] J. A. Stankovic, "Research Directions for the Internet of Things," *IEEE Internet of Things Journal, Vol 1*, pp. 3-9, 2014.

[23] S. Singh and N. Singh, "Internet of Things (IoT): Security challenges, business opportunities & reference architecture for E-commerce," in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, Noida, 2015.

[24] F. V. Jean-François and S. Forster, "The history of Internet of Things (IoT)-Innovate UK," 3 July 2017. [Online]. Available: https://innovateuk.blog.gov.uk/2017/07/03/the-history-of-internet-of-things-iot/.

[25] S. Ranger, "What is the IoT? Everything you need to know about the Internet of Things right now," 21 August 2018. [Online]. Available: https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/.

[26] Gartner Inc., "Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016," 7 February 2017. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016.

[27] P. Mark, J. Shangkuan and C. Thomas, "What's new with the Internet of Things?," May 2017. [Online]. Available: https://www.mckinsey.com/industries/semiconductors/our-insights/whats-new-with-the-internet-of-things.

[28] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, Handbook of Applied Cryptography, Florida: CRC Press, 1996.