High-Speed and Secure Elliptic Curve Cryptosystem for Multimedia Applications

Mohammad Alkhatib
College of Computer and Information Sciences
Al Imam Mohammad Ibn Saud Islamic University (IMSIU)
Riyadh, Saudi Arabia

Abstract—**The last few years witnessed a rapid increase in the use of multimedia applications, which led to an explosion in the amount of data sent over communication networks. Therefore, it has become necessary to find an effective security solution that preserves the confidentiality of such enormous amount of data sent through unsecure network channels and, at the same time, meets the performance requirements for applications that process the data. This research introduces a high-speed and secure elliptic curve cryptosystem (ECC) appropriate for multimedia security. The proposed ECC improves the performance of data encryption process by accelerating the scaler multiplication operation, while strengthening the immunity of the cryptosystem against side channel attacks. The speed of the encryption process has been increased via the parallel implementation of ECC computations in both the upper scaler multiplication level and the lower point operations level. To accomplish this, modified version of the Right to Left binary algorithm as well as eight parallel multipliers (PM) were used to allow parallel implementation for point doubling and addition. Moreover, projective coordinates systems were used to remove the time-consuming inversion operation. The current 8-PM Montgomery ECC achieves higher performance level compared to previous ECC implementations, and can reduce the risk of side channel attacks. In addition, current research work provides performance and resources-consumption analysis for Weierstrass and Montgomery elliptic curve representations over prime field. However, the proposed ECC implementation consumes more resources. Presented ECCs were implemented using VHDL, and synthesized using the Xilinx tool with target FPGA.**

Keywords—Elliptic curves cryptosystem; performance; binary method; projective coordinates; security applications

LIST OF NOTATIONS

ECC	Elliptic Curve Cryptosystem
RLA	Right to Lift Algorithm
SPA	Simple Power Attack
STA	Simple Time Attack
SM	Sequential Multiplication
SA	Sequential Addition
PM	Parallel Multiplier
TSM	Time-consumption for one sequential multiplication
T_M	Time-consumption for one multiplication operation
T_{KP}	Time-consumption for scaler multiplication
GF	Galious Field
NAF	Non-Adjacent-Form

I. INTRODUCTION

Elliptic Curve Crypto-system (ECC) is a type of public key cryptosystems that depend on the discrete logarithm problem for elliptic curves. It has been introduced by Miller and Koblitz in 1985 [1,2]. Since that date, it has been widely used in many security applications due to its reliability and efficiency. By using much shorter key length, ECC can provide equivalent security level to that obtained by other asymmetric ciphers such with consuming less time and resources, which made it very efficient for multimedia applications that need to provide the security services for huge amount of data in the shortest possible period of time and, of course, with the least amount of resources consumed.

A variety of ECC representations over GF(p) and GF (2^n) were presented and used for different elliptic curves applications. First, ECC represents the paintext as a point on an elliptic curve. Then, it encrypts the plaintext by performing a number of arithmetic operations over finite fields. ECC computations can be categorized into upper and lower layers. The upper layer's computations are mainly point doubling and point addition operations, which are performed by the scaler multiplication operation. It is worth mentioning here that the scaler multiplication is the key operation in ECC encryption process. On the other hand, lower level of computations includes addition, multiplication, and inversion operations. The latter is the most time-consuming operation [3].

Previous research works focused on improving the performance and security of ECC encryption. These works studied several possible techniques to accelerate the encryption process by speeding up scaler multiplication operation as well as increasing the cryptosystem immunity against side channel attacks such as simple time (STA) and simple power (SPA) attacks. The major performance improvement techniques include the use of projective coordinates to avoid the costly inversion operation and the parallel implementation of ECC arithmetic computations, especially in the lower level [1-5].

The current research utilizes both the use of projective coordinates and the inherited parallelism in ECC computations, and perform these computations in parallel for both upper and lower computational layers. This is achieved by using parallel hardware components, which are multipliers and adders. In addition to performing the lower level computations in parallel, this study implements the upper layer's operations in parallel to achieve higher speed for encryption process. In particular, proposed ECC performs the two main operations (point doubling and point addition) in scaler multiplication in parallel. This plays crucial role in increasing the speed of scaler multiplication and thus ECC encryption.

In order to enable parallel implementation of point doubling and point addition operations, the current ECC uses modified version of the Right to Left binary algorithm (RLA), which is widely used to perform scaler multiplication. The modified RLA has the ability to apply both operations in parallel once required. This is obtained by assigning an independent variable to save the point operations result of certain iteration of the RLA and use it for point addition in the next iteration. To accomplish optimum performance, proposed ECC uses eight parallel multipliers (8-PM).

The parallel ECC implementation can be realized by using parallel hardware components for hardware implementations, and the known multithreading technique for software implementations [6-8]. This research uses hardware implementations because they are faster and more secure than software implementation for ECC.

Several projective coordinates systems are studied and implemented in this research to achieve greater speed for ECC encryption. Moreover, the main ECC representations over GF(p) are implemented to study their characteristics in terms of performance and security. It should be mentioned that proposed ECC implementation strengthen the security against SPA and STA.

The core contribution of this research is represented by developing a novel and fast ECC using modified version of the RLA. The proposed ECCs utilize the maximum parallelization levels for both Montgomery and Weierstrass curves to achieve optimum performance.

Results showed that the use of proposed RLA to implement ECC operations improved the speed of the encryption considerably. Such ECC designs and implementations are highly recommended for securing applications that need a high-speed ECC, such as multimedia applications.

The remaining parts of this article are the background and related works, ECCs computations and architectures, Results and analysis, and conclusion.

II. Background and Related Works

ECC is a public key cryptography that can be used to provide different security services including confidentiality of data transmitted over communication networks. ECC supper passes other public key cryptosystems because it can provide equivalent security level with using much shorter key size, which represents considerable improvement in terms of performance and resources consumption [1,6].

Scaler multiplication is the main operation in ECC encryption, and it consists of two operations; point doubling and point addition. Several algorithms were used to perform scaler multiplication. It can be noticed that scaler multiplication algorithms use iterative approach to perform point operations, which leads in the end to converting the plaintext to the ciphertext [1, 7-9].

ECC computations performed during scaler multiplication are called upper level computations. The Binary method, the Non Adjacent Form (NAF), and the Montgomery ladder are the main algorithms used to apply ECC scaler multiplication. These algorithms vary in performance and security. The RLA is a form of the binary algorithm which is intensively used for ECC encryption due to its security advantages and the ability to withstand against side channel attacks [10-12].

The RLA, scans the binary bits of the key and always performs point doubling operation regardless the value of the key bit. If the value of the bit is one, the point addition operation is performed as well. Therefore, the RLA assumes that, on average, the point addition operation is executed half times of point doubling during the entire scaler multiplication. However, previous ECC that uses standard RLA implements point operations sequentially, which increases the critical path delay of scaler multiplication [13, 14].

Point doubling is the dominating operation in scaler multiplication. In this operation, the elliptic curve point is added to itself, while in point addition, two different points are added. Computations performed within each point doubling and addition operation are called lower level of computations, which represent arithmetic operations over finite fields. Those operations are mainly modular multiplication, addition, and inversion operations [2-6].

The inversion is the most time-consuming operation in elliptic curve cryptography. Previous researches suggested to use projective coordinates instead of affine coordinates to eliminate the inversion operation by converting it to consecutive multiplications. This played important role in reducing the time delay of scsler multiplication operation. Several projective coordinates systems were presented including homogenous, López-Dahab, and Jacobean coordinates systems [5, 6, 14-18].

Since its first introduction in 1985, different elliptic curves forms over GF(p) were presented [1-3, 16-20]. Some curves such Montgomery, and Tripling Oriented curves have less computational complexity in comparison with other curves such as Weierstrass and Binary Edwards. Thus, particular types of elliptic curves can reduce the time delay for the scaler multiplication operations. This made the curves with lower computational complexity more suitable for security applications that require high-speed cryptosystem [21-23].

The majority of researches in this field focused on hardware implementations of ECC since they are faster and more secure than software implementations. Researchers studied the effect of using the different projective coordinates systems with the main forms of elliptic curves. The majority of previous research works studied the performance and resources-consumption of Weierstrass curve over GF(p) [7-9].

Researchers found that the homogenous projection system (X/Z, Y/Z) accomplished the highest performance levels when implemented with many GF(p) elliptic curve representations; including [8, 11, 12-21]. On the other side Jacobean and López-Dahab coordinates showed better performance when used with other types of elliptic curves. The performance or speed of ECC is usually estimated by the number of sequential multiplication (SM) and addition (SA) levels consumed by point doubling and point addition operations [7, 9-10].

In addition to using projective coordinates, researchers used specific hardware components such as Montgomery multiplier to increase the speed of ECC operations [8].

However, the majority of previous researches studied the use of usual serial implementation of ECC computations, in which only one multiplication or addition operation is performed in every level of computations. Although, serial ECC implementation consumes the least possible resources, it cannot satisfy performance requirements for applications that need to provide the confidentiality of many data streams simultaneously as in multimedia applications [20-23]. A high-performance cryptosystem is in demand to satisfy the requirements of many emerging cloud services technologies such as the frame work presented in [31], which aims to provide secure environment for cloud infrastructures that allows to serve clients in efficient and secure way.

Another study developed hardware implementations for Weierstrass ECC over GF(2^n). The presented fast ECC is suitable for smart card implementations [25]. In [26], researchers introduced Weierstrass ECC, which uses the López-Dahab projective coordinates to eliminate the costly inversion operation.

In [24], authors developed Weierstrass ECC design that can support both types of finite fields. However, the results obtained from proposed ECC showed that it needs extra memory resources.

Recent research studies proposed parallel implementations of ECC computations to increase the speed of encryption process. Actually, this technique improved ECC performance, but with consuming more resources [10, 13-23]. It should be highlighted that the inherited parallelism in elliptic curve computations make it possible to perform lower level operations in parallel manner, which shortens the time delay of the scaler multiplication [19].

In addition to improving the performance, parallel ECC implementation can strengthen the security of the cryptosystem against the STA and SPA. Furthermore, using parallel ECC designs considerably enhance the area×time-consumption^2 (AT^2) factor [20-23].

Authors is [10] developed parallel hardware designs for Weierstrass ECC over GF(p). Homogenous projective coordinates were used to remove inversion operation. The ECC implementation with four parallel multipliers (4-PM) consumed the least time for the encryption process. However, with consuming more resources compared to usual serial ECC implementation. Similar parallel ECC implementation over GF(2^n) was presented in [14]. This study found that using Jacobean projection with Weierstrass curve obtained shorter time-delay compared to other projective coordinates systems.

Another ECC implementation that uses Weierstrass curve over GF(2^n) was proposed in [15]. Although it provided considerable trade-off between performance and power consumption, the presented ECC implementation has low system utilization level, which is necessary for efficient ECC [10].

Many ECC design introduced in previous researches [10, 12, 14, 15] worked on the standard Weierstrass elliptic curve representation, while there are many elliptic curve representations that have not been sufficiently studied. Furthermore, parallel ECC implementations in the aforementioned studies consume additional resources and area. Therefore, they are not appropriate for applications with limited resources.

Other research works provided different ECC design choices by using variable degree of parallelism to mitigate the problem of resources consumption. In other words, researchers studied and implemented all possible ECC design schemes that provided significant trade-off between performance and resources consumption. These research efforts aim to introduce a variety of ECC designs that can satisfy the requirements of several security applications in terms of performance and resources-consumption [17-23].

A number of parallel ECC design schemas over GF(p) were introduced in [17]. Proposed designs used variable degrees of parallelism for ECC computations. Experimental results showed that the 4-PM design achieved the shortest time delay for ECC point addition, which is estimated by four SMs. Other researchers [19] studied the use of variable degrees of parallel designs to increase the speed of point doubling operation. Experiments of both researches found that Weierstrass ECC accomplishes the highest performance level when implemented using four PMs and homogenous coordinates system. Other presented design schemes in [17, 19] provide important trade-off between area and performance, which could be useful for a variety of elliptic curves applications.

Few research works investigated the parallel implementation of ECC forms. In [22] authors analyzed the performance and resources-consumption levels of Montgomery elliptic curve over GF (p) when implemented using different projective coordinates and parallel hardware designs. A set of design choices for Montgomery ECC were presented. Experimental results illustrated that the 2-PM design accomplished the best trade-off between area and performance, where the 4-PM ECC obtained the highest speed for encryption process.

Similar research works studied the characteristics of Tripling Oriented ECC when implemented using different degrees of parallelism [23].

Authors in [20] and [21] used similar methodology to study the performance and cost features of Binary Edward and Edward elliptic curves respectively. Experiments showed that the greatest speed level of Edward ECC can be reached using the 5-PM design. On the other side, Binary Edward ECC accomplished the shortest time delay when implemented using the 7-PM design. Homogenous projective coordinates system represented best choice for both forms since it involves less arithmetic computations than other projections.

The vast majority of the previous studies used the known Binary method to apply the ECC scaler multiplication. It should be mentioned here that recent few researches investigated the use of other algorithms to process ECC point operations.

Authors in [29] used the NAF algorithm and parallel hardware designs the perform ECC upper level of computations. The results of that research proved that the use of NAF algorithm to perform Montgomery ECC operations can reduce the time delay of the encryption process considerable in comparison with usual Binary method. These outcomes are supported by the fact that NAF algorithm requires performing less number of point addition operations during the scaler multiplication.

Another ECC implementation that uses NAF algorithm was reported in [30]. Researchers studied possible improvement for both Edward and Binary Edward ECCs using parallel hardware designs as well as the NAF algorithm for scaler multiplication. It was found that NAF algorithm can enhance the performance of the encryption process compared to parallel ECC implementations using Binary method. However, the Montgomery ECC presented in [29] remains ahead of the later ECCs developed by [30] in terms of performance.

In [32], researchers presented secure ECC suitable for compact devices by using a modified addition formula for usual RLA. Although proposed method needs less space complexity, the time delay of the encryption process was increased because of using affine coordinates. This makes it unable to satisfy the performance requirements for many web applications. Another promising research work tried to accelerate ECC computations using a modified approach for the Montgomery ladder algorithm [33]. However, the proposed algorithm did not utilize inherited parallelism in the different levels of computations in the encryption process, which is essential to develop high speed crypto processor. Moreover, ECCs presented in [32-33] are vulnerable to side channel attacks.

The research works toward improving the speed and security of ECC needs to give more attention to the development of fast and secure algorithms that benefit from the fact that encryption's computations can be implemented in parallel in both the upper and lower levels of operations.

The current research work proposes a modified version of the RLA that enables parallel implementation of the upper level of computations for ECC represented by point addition and point doubling operations. In addition, proposed ECC implementations utilize the inherited parallelism of the lower level of computations for ECC represented by arithmetic computations over GF(p). This research analyzes the performance and resources consumption level of the major types of elliptic curves, which are Weierstrass, Edward, and Montgomery curves. The use of different projective coordinates systems is also investigated to find the most suitable ECC design for applications that need high-performance cryptosystem such as multimedia applications.

The next section presents the research methodology followed in this study to obtain the research outcomes.

III. Research Methodology and Methods

The research methodology used to conduct this study is depicted in Fig. 1.

It can be noticed from Fig. 1 that the research methodology of the current study is divided into three phases, as follows:

- Preparation of elliptic curve computational schemes,
- Improving the speed of ECC's lower level of computations, and
- Improving the speed of ECC's scaler multiplication (upper level of computations).

In the first stage, the main research efforts focused on studying the key ECC representations over GF(p). According to security, and computational complexity features, three forms of elliptic curves were chosen for implementation in the current research. These forms are Weierstrass, Edward, and Montgomery curves.

In addition to selecting the elliptic curves forms, the use of different projective coordinates systems was investigated. In particular, the computations of point doubling and point addition operations for each ECC form were performed using different projection. The use of projective coordinate avoids the need for the time-consuming inversion operation. Inversion is converted to a number of multiplications. This contributes in reducing the time delay of the lower level of computations. The projective coordinates implemented in this study are homogenous, López-Dahab, and Jacobean systems.

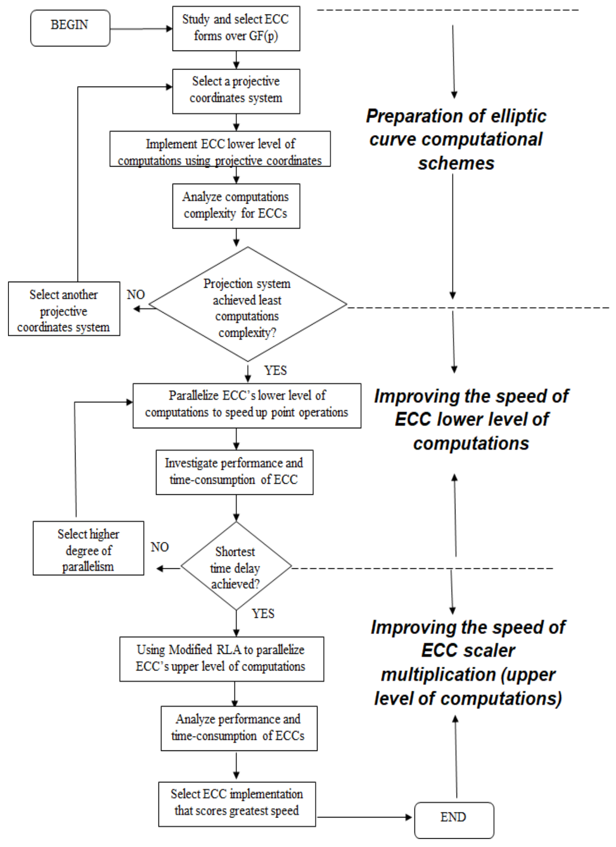
BEGIN
Study and select ECC forms over GF(p)
Select a projective coordinates system
Implement ECC lower level of computations using projective coordinates
Analyze computations complexity for ECCs
Projection system achieved least computations complexity?
NO
Select another projective coordinates system
YES
Preparation of elliptic curve computational schemes
Parallelize ECC's lower level of computations to speed up point operations
Investigate performance and time-consuming of ECC
Shortest time delay achieved?
NO
Select higher degree of parallelism
YES
Improving the speed of ECC lower level of computations
Using Modified RLA to parallelize ECC's upper level of computations
Analyze performance and time-consuming of ECCs
Select ECC implementation that scores greatest speed
END
Improving the speed of ECC scaler multiplication (upper level of computations)

Fig. 1. Research Methodology.

The last step in the first stage involves analyzing the performance or time consumption level for the ECC computations. Such analysis aims to find the most efficient projective ECC that has the least computational complexity. This is essential for the process of developing a high-speed ECC for different security applications.

At the second stage of this work, researchers performed ECC's lower level of computations using different degrees of parallelism in order to reduce the time delay for finite fields arithmetic included in each point doubling and addition. The ECC design or computational scheme that gives the shortest possible time delay for each ECC form is then implemented using parallel hardware designs. Performance and resources-consumption features for ECCs are thoroughly analyzed and studied.

It should be mentioned that the performance of proposed ECCs is estimated using the number of sequential multiplication levels consumed during point doubling or point addition operation. On the other side, resources consumption is estimated by the number of parallel hardware components needed for each design.

In the final stage of this research, a modified version of RLA is used to apply scaler multiplication. This modification allows to perform point addition in parallel to point doubling via saving the results of the previous RLA's iteration in a separate variable, and then it is added to the current elliptic curve point, while performing the point doubling computations.

In other words, researchers parallelized the ECC's upper level of computations to accomplish the maximum speed for encryption process. Such ECC implementation increases the performance but needs more resources to apply ECC computations.

Proposed ECCs will be first studied theoretically by observing the parallel computational schemes and estimating the time and resources consumption levels. Then, presented ECCs are implemented using the Xilinx tool to find out, more accurately, the time and resources consumed for encryption process.

In addition to performance, other factors that are important to determine the efficiency and appropriateness of ECC are investigated. Those factors include the area × time (AT), area × time^2 (AT^2), and hardware/system utilization.

The next section of this research illustrates the computations and hardware designs for proposed ECCs.

IV. Computational Schemes and Designs for High-Speed ECCs

This section presents the equations of the main ECC forms studied in this research as well as the points computations involved in ECC encryption process.

In addition, this section shows computational schemes for each curve and the parallel hardware designs needed to perform ECC computations.

The modified version of RLA used in scaler multiplication is illustrated as well.

A. ECC Points Computations

The two main operations in ECC scaler multiplication are point doubling and point additions. The key difference in the computations of the two operations is in finding the slop (m), which is the derivation of elliptic curve equation.

For point addition the calculation of m is similar for all ECC forms over GF(p), while in point doubling the slope differs from one form to another. Equations required to compute the slope for point doubling and point addition operations are presented in (1) and (2), respectively. For more information about calculating the slope for ECC point operations, the reader may refer to [27-29].

In point doubling, the slope (m) =

$$\frac{dy}{dx} = \frac{3[x^2+2a(x+1)]}{(2y)} \tag{1}$$

In point addition,

the slope (m) = $(y_2 - y_1)/(x_2 - x_1)$ (2)

It can be noticed that elliptic curve points are represented by x and y coordinates. In point doubling (3), the elliptic curve point is added to itself where in point addition (4), to different points are added. The result is a third point $P_3(x_3, y_3)$ on an elliptic curve, which can be computed as follows:

$$x_3 = m^2 - x_1 - x_2 \tag{3}$$

$$y_3 = m(x_1 - x_3) - y_1 \tag{4}$$

This research studies two major forms of elliptic curves, which are Weierstrass, and Montgomery curves represented in equations (5), and (6), respectively. These curves are very important because the Weierstrass curve is the most used form for ECC cryptographic operations, while the Montgomery curve is distinguished from its counterparts in terms of the degree of complexity for ECC computation, which may lead to the development of high-speed cryptosystem.

$$E: Y^2 = X^3 + aX + b \tag{5}$$

where a and b belong to GF (p) and $4a^2 + 27$ and $b \neq 0$.

$$by^2 = x^3 + ax^2 + x \tag{6}$$

where a, b belong to GF (p), and with $b^*(a^2 - 4) \neq 0$.

According to previous studies, homogenous projective coordinates (X/Z, Y/Z) achieves the least computational complexity for points operations in the above mentioned elliptic curve forms represented by equations (5), and (6) [25-30].

Using projective coordinates, the elliptic curve points are represented by three coordinates, which are x, y, and z.

In Montgomery ECC, the result of point doubling can be calculated as follows:

The slope M= $\frac{3X^2+2aXZ+Z^2}{2bYZ}$

$$X_3 = (2bYZ)[(3X^2 + 2aXZ + Z^2)^2 - 8b^2Y^2ZX]$$

$$Y_3 = \{(3X^2 + 2aXZ + Z^2)[12b^2Y^2ZX - (3X^2 + 2aXZ + Z^2)^2] - 8b^3Y^4Z^2\}$$

$$Z_3 = (2bYZ)^3$$

The result of point doubling operation for Weierstrass is another point represented by the three coordinates x_3, y_3, and z_3, as follows:

The slope $M = \frac{3X^2+aZ^2}{2YZ}$

$$X_3 = 2YZ * [(3X^2 + aZ^2)^2 - 8XZY^2]$$

$$Y_3 = (3X^2 + aZ^2) * [12XZY^2 - (3X^2 + aZ^2)^2] - 8Y^4Z^2$$

$$Z_3 = 8Y^3Z^3$$

It can be noticed that the slope (M) depends on the elliptic curve equation, while in point addition the calculation of the slope does not relate to the elliptic curve equation. Therefore, point addition computation is similar for all curves using the homogenous projection, and can be computed as follows:

$$M = \frac{Y_1Z_2 - Y_2Z_1}{X_1Z_2 - X_2Z_1}$$

$$X_3 = (X_1Z_2 - X_2Z_1) * [Z_1Z_2 * (Y_1Z_2 - Y_2Z_1)^2 - (X_1Z_2 + X_2Z_1) * (X_1Z_2 - X_2Z_1)^2]$$

$$Y_3 = \left\{ \begin{matrix} (Y_1Z_2 - Y_2Z_1) * \\ \left[\begin{matrix} X_1Z_2(X_1Z_2 - X_2Z_1)^2 - \\ (Z_1Z_2(Y_1Z_2 - Y_2Z_1)^2 - (X_1Z_2 + X_2Z_1) * (X_1Z_2 - X_2Z_1)^2) \end{matrix} \right] - \\ Y_1Z_2 * (X_1Z_2 - X_2Z_1)^3 \end{matrix} \right\}$$

$$Z_3 = Z_1Z_2 * (X_1Z_2 - X_2Z_1)^3$$

In order to find the result of point doubling and point addition for each curve, a number of arithmetic operations, such as modular multiplication and addition over GF(p), should be performed. The required computations for point doubling and addition are called computational scheme for certain ECC and they play important role in determining the performance of the cryptosystem.

The next section presents the computational schemes for proposed Weierstrass and Montgomery ECCs as well as the hardware designs needed to implement them.

B. *ECC Computational Schemes and Hardware Designs*

The computational scheme for ECC is the series of modular multiplication, addition, and subtraction operations required to perform point doubling and addition operations and hence calculating the values of x3, y3, and z3 presented in the previous section.

Usually, ECC computations are implemented sequentially using hardware components, which are multipliers (M) and adders (A). In this research, all possible parallelization levels for performing ECC computations were investigated to find out the parallelization level that obtains the shortest time delay.

Table I presents the estimated performance levels for both Weierstrass and Montgomery ECCs when implemented using different parallelization levels as well as the required hardware devices for each implementation.

As can be noticed from Table I, the performance of ECC implementation is estimated by the number of sequential multiplication (SM) and sequential Addition (SA) levels required to perform point doubling operation. It is worth mentioning that the time consumed by SAs is neglected compared to SMs. So, the later will be the main factor used to estimate the time-consumption of ECC.

It is obvious that the 4-PM implementation for Montgomery ECC achieves the shortest time delay estimated by three SMs and three SAs. Similar performance was reported by Weierstrass ECC implementation but with using higher degree of parallelism, and hence more resources. In particular, the 5-PM Weierstrass ECC implementation can get comparable performance level to that obtained by Montgomery curve using 4 PMs.

Theoretical results showed that the use of more than 4 PMs for Montgomery ECC has no impact on the performance level. The same applies for the 5-PM ECC implementation for Weierstrass curve. In other words, the performance level is saturated at this degree of parallelism and cannot be improved further by adding more parallel Ms and As.

Experiments showed that the use of parallel hardware components is controlled by the inherited parallelism in ECC computations for both curves. For example, no additional parallel computations for Montgomery ECC point doubling can be performed in the first SM level, which involves five parallel multiplication operations. In order to start the second level's computations, the crypto processor needs to obtain the results of the first level, and so on.

Although other ECC implementations presented in Table I requires less resources because of the use of less number of hardware components, it seems that they consume greater time to perform point doubling. This research focuses on ECC implementations that score the highest possible performance level.

Similarly, Table II presents the theoretical results for studying the performance and resources consumption for ECC point addition. It is worth remembering here that the computations of point addition are similar for all ECC forms.

TABLE I. COMPUTATIONS REQUIRED TO PERFORM ECC POINT DOUBLING WITH DIFFERENT PARALLELIZATION LEVELS USING PROJECTIVE COORDINATES (X/Z, Y/Z)

Elliptic Curves Form	**Hardware Design**	**Parallel Hardware Units**	**Sequential multiplication and addition levels**
ECC Point Addition for Montgomery and Weierstrass Curves	Serial Design	1M, 1A	16 SM, 6 SA
	2-PM	2M, 2A	8 SM, 5 SA
	3-PM	3M, 2A	6 SM, 5 SA
	4-PM	4M, 3A	4 SM, 4 SA
	5-PM	5M, 3A	4 SM, 4 SA

TABLE II. COMPUTATIONS REQUIRED TO PERFORM ECC POINT ADDITION WITH DIFFERENT PARALLELIZATION LEVELS USING PROJECTION (X/Z, Y/Z)

Elliptic Curves Form	Hardware Design	Parallel Hardware Units	Sequential multiplication and addition levels
Weierstrass ECC	Serial Design	1M, 1A	12 SM, 4 SA
	2-PM	2M, 2A	6 SM, 3 SA
	3-PM	3M, 2A	4 SM, 3 SA
	4-PM	4M, 2A	4 SM, 3 SA
	5-PM	5M, 2A	3 SM, 3 SA
Montgomery ECC	Serial Design	1M, 1A	12 SM, 4 SA
	2-PM	2M, 2A	6 SM, 3 SA
	3-PM	3M, 2A	4 SM, 3 SA
	4-PM	4M, 2A	3 SM, 3 SA
	5-PM	5M, 2A	3 SM, 3 SA

It can be noticed from Table II that the 4-PM ECC implementation for point addition can achieve the shortest time delay estimated by four SM and four SA levels. Using higher degrees of parallelism has no positive impact on the performance as can be seen from the table.

Other presented ECC implementations represent a considerable trade-off between performance and time-consumption. This could benefit the purpose of developing efficient ECC for different security applications in accordance with required speed and the limitations on available resources. This research concerns the high-speed ECC implementations appropriate for multimedia security applications.

Fig. 2 and 3 present the parallel hardware designs of the computational schemes for high-speed ECCs using Montgomery and Weierstrass curves over GF(p), respectively.

Note that only the hardware designs that provide the shortest time delay are introduced, which the 4-PM and 5-PM designs for Montgomery and Weierstrass ECCs, respectively.

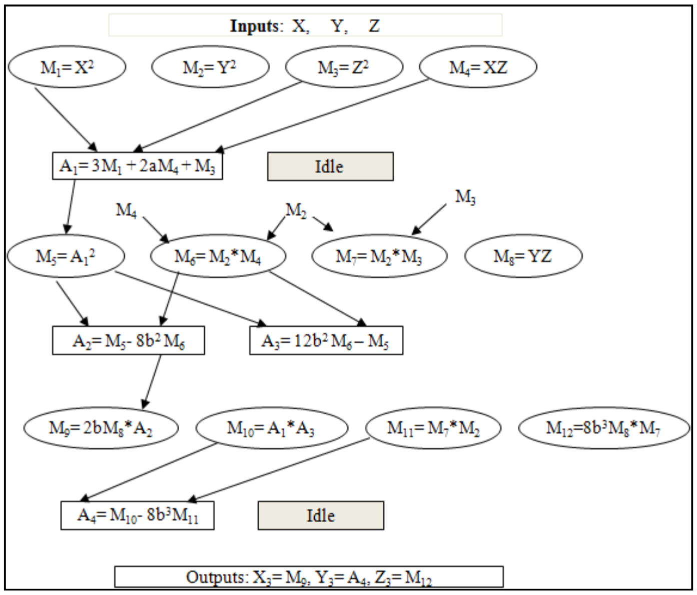

Fig. 2. Hardware Design for Montgomery ECC Computations.

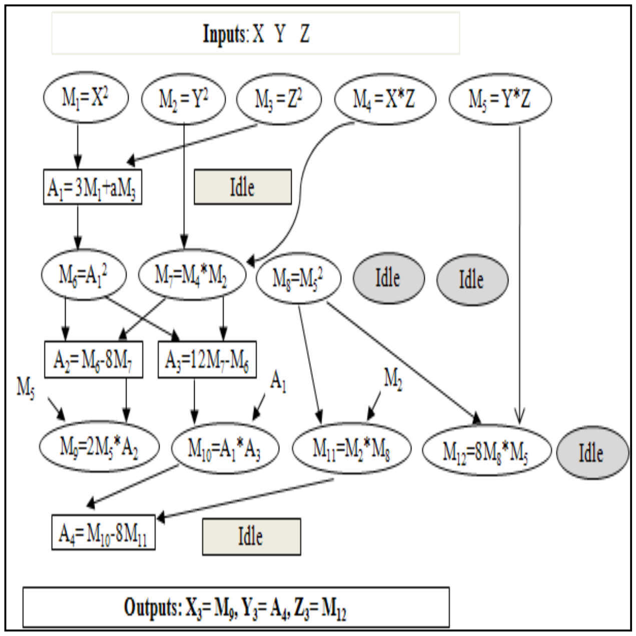

Fig. 3. Hardware Design for Weierstrass ECC Computations.

It should be highlighted here that the time consumed by one SM level is equivalent to the time consumption of one multiplication operation regardless how many multiplications are performed in that level.

For example, the first level SM level in Weierstrass ECC involves five multiplications. In despite of that, it consumes similar time to that needed for one multiplication operation. This represents a great benefit in terms of performance that can be achieved using parallel hardware implementation for ECC.

Fig. 4 shows the parallel hardware design for the computational scheme of ECC point addition operation. This design applies for both elliptic curves studied in this research.

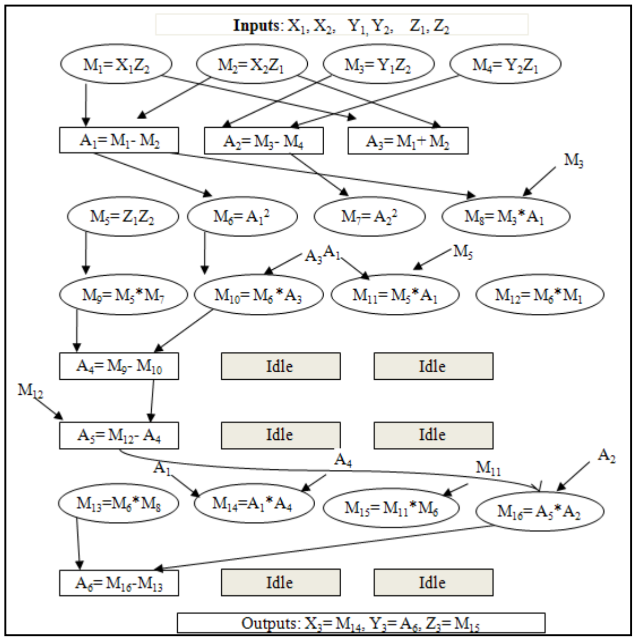

Fig. 4. Hardware Design for Point Addition Computations.

The results of point addition can be obtained after performing four SM and four SA levels.

As can be seen from Fig. 2, 3, and 4, the parallel implementation of the lower level of computations for ECC can improve the performance greatly. In addition, the next section presents the modified version of RLA, which allows the parallel implementation of upper level of computations for ECC. This contributes in accelerating the scaler multiplication even further.

C. Modified Implementation of RLA

This section introduces an implementation of a modified version of known RLA for scalar multiplication, which is the main operation in ECC encryption process. The current proposed implementation allows to perform the upper level computations, represented by point doubling and point addition operations in parallel once required during scalar multiplication. The modified version of RLA is depicted in Fig. 5.

As can be noticed from Fig. 5, the inputs to the RLA are as follows:

- P which is a point on an elliptic curve E. The point P represents the original plaintext.
- K is the scalar represented by a series of binary bits (from 0 to n-1). N is the key length.

Outputs of RLA is another point (Q), which represents the encrypted message. Q is the result of scalar multiplication operation ([k]P).

```
Modified right to left binary algorithm
Inputs: $P \in E(GF(p))$, $k = (k_{n-1},\ldots,k_1,k_0)_2 \in N$
Output: $Q = [k]P$
1. $R_0 = O$ (Infinity); $R_1 = P$ ; $J = P$
2. For $i = 1$ to $n-1$ do
3. $J = R_1$ (Save the result of point doubling ($R_1$) from previous
   iteration)
4. IF $k_i = 1$ then
      4.1. $R_0 = R_0 + J$ (point addition)   } Performing point addition
      4.2. $R_1 = 2R_1$ (point doubling)       } & doubling in parallel
5. $R_1 = 2R_1$ (only point doubling is performed if the binary bit $k_i \neq$
   0)
6. End for
7. Return $R_0$ (final result which equals [k]P)
```

Fig. 5. Modified Version of RLA for Scalar Multiplication.

At the beginning of the algorithm, three elliptic curve points are defined, as follows:

- R_0 = O, which is used to save the result of point addition during the scalar multiplication.
- R_1 = P, which is used to save the result of point doubling during the scalar multiplication.
- J = P, which is additional elliptic curve point used for temporary saving the value of R1 of previous iteration in scalar multiplication.

The RLA depends on the iterative approach and starts by scanning the binary bits of k from right to left.

For each bit in k, if the value of k_i equals one then both point addition and doubling operations are performed. Otherwise, only the point doubling operation is performed.

Note that the variable point J is used in each iteration to save the value of R1 obtained from the previous iteration of scalar multiplication. Then, J is used to perform the point addition operation in the current iteration. Thus, the point addition does not require the current value of R1 which is being calculated by point doubling operation. In other words, both operations become independent and the can be processed in parallel manner.

The use of proposed modified version of RLA allows to perform the upper level computations of ECC in parallel to reduce the time delay of scalar multiplication and hence improve the performance of the cryptosystem.

In order to implement point doubling and point addition operations in parallel, extra hardware resources are needed. In particular, eight PMs are required for Montgomery curve. The 8-PM ECC design allows to perform the four parallel multiplications included in each point doubling and point addition as illustrated in Fig. 2 and 4. If the value of k_i is zero, only the point doubling is performed which makes four out of eight multipliers idle as this iteration. On the other hand, the Weierstrass ECC needs nine PMs to implement point operations in parallel. Five PMs for point doubling and four PMs for point addition as clarified in Fig. 3 and 4. The 8-PM and 9-PM designs accomplish the highest performance level for Montgomery and Weierstrass ECCs, respectively.

It should be mentioned here that the time complexity of proposed RLA is estimated by O(n), where n is the size of inputs key or the number of binary bits of the key. This means it takes linear time complexity, which can satisfy required performance level for many applications.

Another advantage of using the proposed ECC is that it can increase the immunity of the cryptosystem against side channel attacks such the known STA. Each iteration of the modified RLA consumes approximately similar time regardless the value of the binary bit of the scalar k. This is because both operations are conducted in parallel if the value is one. So, the attacker cannot reveal the value of the binary bit k_i in each iteration by analyzing the time consumed. In this way the secret key, represented by the binary bits' vector of the scalar k, is kept confidential and cannot be exposed to the attacker by using the SPA.

On the other side, usual ECC implementations that use serial design implements point doubling and addition operations sequentially. This enables the attacker to trace the time consumed by certain iteration of the scalar multiplication operation and thus determining whether one or two operations have been performed by that iteration. Thus, it is possible to infer the values of the key bits by observing and analyzing the time consumption of RLA's iterations.

As mentioned previously, the current research work aims to develop secure and high-performance ECC that can encrypt/decrypt huge amount of data within the least possible time.

The next subsection describes the hardware implementation environment for proposed ECCs.

D. Implementation Environment

This research work seeks to develop high-speed ECC using hardware implementation. Thus, experiments conducted in this study investigates possible hardware design choices and analyzes the time delay and resources consumption for proposed hardware implementations. For this purpose, this study uses the popular hardware description language VHDL to implement proposed designs. The code for each proposed ECC is written in VHDL and then simulated using the ModelSim tool for validation purposes.

Authors used the standard carry save multiplier and carry save adder to perform field arithmetic operations for elliptic curve.

In order to simulate and synthesize suggested ECC implementations and obtain performance and resources-consumption results, researchers also use the Xilinx tool with the target FPGA (Field Programmable Gate Array) chip family chosen to be virtex5 (XC5VLX30).

The next section presents and analyzes the implementation results for proposed Montgomery and Weierstrass ECCs.

V. Results and Analysis

This section discusses the performance and resources consumption results for proposed ECCs in the current research.

Researchers observed the architectures of the proposed ECC implementations for Weierstrass and Montgomery forms. It can be noticed from previously presented hardware designs in Fig. 1 and 3 that such design requires eight parallel multipliers to implement the Montgomery ECC point doubling and point addition operations in parallel manner. In particular, four parallel multipliers are needed for each operation.

The Weierstrass ECC designs presented in Fig. 2 and 3 needs nine parallel multipliers; four for point addition and five for point doubling.

The required hardware resources, the estimated time consumption for point doubling and point addition as well as the overall time delay for the scalar multiplication are shown in Table III. It also shows the AT and AT^2 cost factors.

It should be mentioned here that time consumption for proposed ECCs is estimated in terms of the number of SM levels consumed by each point operation. The overall time consumption is calculated using the following equation:

$$\text{Overall time} = (\text{No. SMs for point doubling}) + (0.5 \times (\text{TIME of POINT ADDITION}) \quad (7)$$

The RLA assumes that point addition happens in half the number of the binary bits for the scalar (k). So, the time of point addition is multiplied by 0.5 in the above equation.

Remember that point doubling and addition are performed in parallel and the point addition consumes one SM level more than point doubling. Therefore, if the binary bit of k equals one, the time consumed is equivalent to the time of point doubling plus one SM level. The time delay of one SM is about one third the time allocated for point doubling operation as can be seen from Fig. 2 and 3. It can be estimated by (0.33 × TIME OF POINT DOUBLING). Therefore, equation (7) can be written as follows:

$$\text{Overall time} = (\text{No. SMs for point doubling}) + (0.5 \times (0.33 \times (\text{No. SMs for point doubling}))) \quad (8)$$

Table III shows performance and area consumption features for the proposed Montgomery and Weierstrass ECCs. This research uses the modified version of RLA to apply point doubling and point addition operations in parallel as depicted in Fig. 4. The 8-PM ECC for Montgomery curve scores the shortest time delay, estimated by 3.5 multiplication cycles. It also achieves the best AT^2 results compared to other ECC implementations presented in Table III. It is worth mentioning that the AT^2 factor focuses more on the time delay and it should be considered very important for developing high-speed ECC. Results showed that proposed 9-PM Weierstrass ECC achieves similar performance level but with consuming more resources.

It can be noted from Table III that proposed Montgomery and Weierstrass ECCs outperform their counterparts that use normal RLA in terms of performance. However, they consume more resources to enable the parallel implementation of the upper level of computations for the scalar multiplication operation.

Table III presents estimated results obtained by studying proposed ECC designs and analyzing their time delay and resources consumption features.

This research provides hardware implementation for proposed ECCs using the modified RLA. Implementation results show acual performance and resources consumption levels for Montgomery and Weierstrass ECCs when implemented using 8-PM and 9-PM designs. Obtained results are then compared with the corresponding ECC implementations that use usual RLA.

The performance is the most important factor to be considered when developing a high-speed ECC. It is assessed using the time required to perform the scalar multiplication operation (T_{KP}), which can be computed through the following four steps of calculations.

TABLE III. COMPARISON BETWEEN PROPOSED PARALLEL ECC IMPLEMENTATIONS USING MODIFIED RLA AND ECCS THAT USE USUAL RLA

Scalar Multiplication Algorithm	Elliptic Curve Representation over GF (p)	ECC design	Consumed Resources	Time Delay			Cost Factors	
				Point Doubling	Point Addition	Overall Time	AT	AT²
Proposed Parallel RLA	Montgomery ECC	8-PM	8 M, 5 A	3	4	3.5	28	98
	Weierstrass ECC	9-PM	9 M, 5 A	3	4	3.5	31.5	110.25
Usual RLA implemented in [10-13, 17-21]	Montgomery ECC	4-PM	4 M, 2 A	3	4	6	24	144
	Weierstrass ECC	5-PM	5 M, 2 A	3	4	6	30	180

In the first step, we need to calculate the time consumption of one multiplication (T_M) operation by using the following equation:

$$T_M = (\text{cycles/bit}) \times m \times \text{clockperiod} \quad (9)$$

where **m** is the key size. The current research experiments implement proposed ECCs using the key sizes of 256 bits, 512 bits, and 1024 bits.

For example, by using equation (9), the T_M can be computed as follows:

T_M= 1 × 256 × 9.079 = 2324.224 nano sec (n sec)

Remember that in our proposed parallel ECC implementations the time consumption of one multiplication operation equals the time consumed by an entire SM level **(T_{SM})**.

Secondly, compute the time consumption for each point doubling and point addition operation, which represent the main building blocks of the scaler multiplication.

This research concerns the ECC design that accomplish the heist performance level with consuming the least resources. Therefore, the 8-PM Montgomery ECC is implemented. It requires 3 SMs for point doubling and 4 SMs for point addition. The time consumed by one point addition (T_{ADD}) and one point doubling (T_{DBLE}) can be computed using the following equation:

$$T_{ADD} = 4*T_{SM} = 9296.869 \text{ n sec}, \; T_{DBLE} = 3*T_{SM} = 6972.672 \text{ n sec} \quad (10)$$

In the third step, we compute the time of one inversion operation (T_{INV}), which is required to convert the projective coordinates back to the affine coordinates. The time of one inversion is equivalent to the time of three SMs [20-24]. Hence, the T_{INV} can be computed as follows:

T_{INV}= 3 * T_{SM}= 6972.672 n sec

Finally, in the last step the total time consumption of the scalar multiplication (T_{KP}) is computed via the following equation:

$$T_{KP} = ((256 \times 0.5 \times T_{ADD}) + (256 \times T_{DBLE}) + T_{INV}) \times 10^{-6} \quad (11)$$

Remember that using proposed parallel implementation for point doubling and point addition the **T_{ADD}** can be estimated one SM (**T_{SM}**), which is equivalent to **(0.33 × T_{DBLE}),** approximately. Therefore, equation 11 can be rewritten as follows:

$$T_{KP} = ((256 \times 0.5 \times (T_{SM})) + (256 \times T_{DBLE}) + T_{INV}) \times 10^{-6} \quad (12)$$

The result of the equation (12) is multiplied by 10^{-6} to convert the final value from nano to mille seconds.

The performance or time consumption of the proposed 8-PM Montgomery ECC can be calculated using equation 12 as follows:

T_{KP}= ((256× 0.5 × 2324.224) + (256 × 6972.672) + 6972.672) × 10^{-6} = 2.089 m sec.

The performance of proposed Montgomery ECC equals 2.089 m sec. It overcomes the corresponding ECC implemented using normal RLA as well as the fastest known implementation using NAF algorithm introduced in [30]. ECCs are implemented using key sizes 256 bits 512 bits, 1024 bits, and 2048 bits as can be noticed from the table.

Table IV shows a comparison between proposed Montgomery ECC and other ECC implementations using NAF algorithm and usual RLA presented in [30] and [22] respectively. It is obvious that proposed 8-PM ECC accomplishes the best performance results compared to other ECC implementations using the different key sizes. For security reasons, it is recommended to use key sizes of 2048 bits and above for encryption and decryption processes. Current implementation shows the time consumption of ECC encryption using different key sizes. It was noticed from the experiments that the time consumptions of encryption and decryption operations are comparable when using the same key size. The Montgomery ECC using NAF algorithm obtains higher performance in comparison with the corresponding ECC implementation that uses normal RLA.

It should be mentioned that Montgomery elliptic curve has less computational complexity compared to the majority of other elliptic curves forms. Therefore, it is highly recommended to use it when developing high-speed cryptosystem.

The 8-PM ECC introduced in this research improves the cryptosystem immunity against STA. Using such cryptosystem's architecture makes it difficult to identify the binary bits of the decryption key via tracing and analyzing the time consumed by each RLA's iteration. The parallel implementation of upper level's operations prevents the attacker from distinguishing the points operation performed in every iteration, and thus the key bit cannot be revealed.

For example, assume that the time-consumption of point doubling is equivalent to TD while the time of point addition is estimated by TA. In usual RLA, if the current key bit equals one the time consumed by the cryptosystem for the processing of the finite field arithmetic computations can be estimated by TD+TA, while it can be estimated by TD if the bit value is zero. The proposed ECC algorithm consumes approximately similar time regardless the value of the key bit. This prevents the attacker from tracing the time-consumption of ECC to find out the sequence of bits in the key.

However, it seems that proposed ECC consumes more area and resources than other ECC implementations presented previously. In despite of that, the additional cost can be afforded for the sake of developing high-speed and secure ECC that fits certain security applications.

Although the proposed 9-PM Weierstrass ECC achieves comparable performance results to that achieved by the 8-PM Montgomery ECC, it needs more resources.

Fig. 6 shows a comparison between the performance levels of ECC implementations presented in Table IV. It can be clearly noted that proposed 8-PM Montgomery ECC using modified RLA outperforms its counterparts for all key lengths ranging from 256 bits to 2048 bits. Therefore, it represents considerable choice for the development of high-performance cryptosystem.

Table V presents time-consumption comparison between proposed Montgomery ECC and the main ECC implementations found in previous research works. As can be noticed, the proposed high-speed ECC achieves better performance results for the encryption/decryption process than results reported in previous literature.

This highlight the importance of the parallel implementation of point doubling and point addition once needed in the modified RLA iterations.

The next section of this article presents the key conclusions and future research works.

TABLE IV. COMPARISON BETWEEN PROPOSED 8-PM ECC USING MODIFIED RLA AND OTHER MONTGOMERY ECCS USING NORMAL RLA AND NAF ALGORITHM

ECC Implementation	256 Key bits	512 Key bits	1024 Key bits	2048 Key bits
Proposed 8-PM Montgomery ECC using modified RLA	**2.089 m sec**	**4.171 m sec**	**8.337 m sec**	**16.674 m sec**
The 4-PM Montgomery ECC using NAF algorithm [30]	2.506 m sec	5.012 m sec	10.024 m sec	20.048 m sec
The 4-PM Montgomery ECC using normal RLA [22]	2.979 m sec	5.957 m sec	11.916 m sec	23.832 m sec

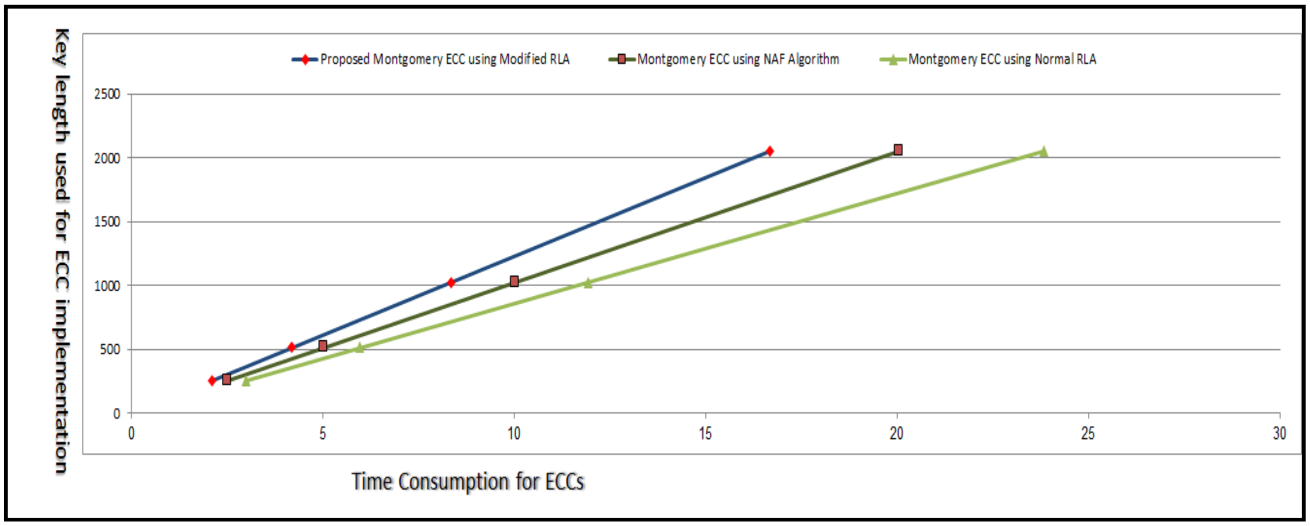

Fig. 6. Comparison between Performance Levels for Presented Montgomery ECC using different Key Lengths.

TABLE V. PERFORMANCE COMPARISON WITH PREVIOUS ECC IMPLEMENTATIONS

No	Name, Ref.	Finite Field, Key size	KP time (m sec)	Device
1	Lo'ai et al. [12] (Edwards)	GF (p), 512-bit	33.301	XC5VLX30
2	Kerins et al. [25]	GF (2^n), 233-bit	13.2	XCV2000E
3	Nele et al. [26]	GF (2^n), 160-bit	3.8	XCV800-4
4	Kocabas et al. [28] (Binary Edwards)	GF (2^n), 163-bit	149.50	ASIC
5	Alkhatib [18] (Binary Edward curve)	GF (p), 256-bit	3.874	XC5VLX30
6	Alkhatib [21] (Edward curve)	GF (p), 256-bit	4.469	XC5VLX30
7	Montgomery ECC implementation using NAF [30]	GF (p), 256-bit	2.506	XC5VLX30
8	Montgomery ECC implementation using Normal RLA [22]	GF (p), 256-bit	2.979	XC5VLX30
9	**Proposed 8-PM Montgomery ECC using modified RLA**	**GF (p), 256-bit**	**2.089**	**XC5VLX30**

VI. CONCLUSION AND FUTURE WORK

This article introduced high-speed Montgomery ECC appropriate for security applications that requires fast encryption/decryption process.

Experimental results illustrated that proposed ECC surpasses previously developed ECCs in terms of performance. Much shorter time delay was reported by proposed ECC using different key lengths. The least time-consumption reported in this study is 2.089 m sec. This research developed high-performance cryptosystems for both Montgomery and Weierstrass curves. The later consumes more resources and hence the priority is given for Montgomery curve implementation.

Furthermore, proposed ECC improves the cryptosystem security against side channel attacks such as STA. It hinders the attacker's mission to analyze the time consumed by each iteration of the RLA. This was achieved by applying point doubling and point addition in parallel manner to hide the variation of time consumed by the RLA iterations.

A number of factors contributed in improving the performance and security level for proposed ECCs. Projective coordinates were used to avoid the time-consuming inversion operation. Finite field computations in each point operation were implemented using parallel hardware design. This increase the speed of performing the lower level of computations in ECC.

This research also proposed modified version of RLA to allow the parallel implementation of the upper level of computations represented by scalar multiplication operations. In order to achieve this, 8-PM and 9-PM designs were used for Montgomery and Weierstrass ECCs, respectively.

Results and comparisons illustrated that proposed high-speed cryptosystem accomplished better performance level compared to Montgomery ECCs using the usual RLA and NAF algorithm. It exceeds the performance of the major ECC implementations proposed in previous studies.

Such high-speed cryptosystem is considered the optimum choice for security applications that need fast encryption/decryption process to provide confidentiality for huge amount of data with consuming the least possible time. In multimedia applications, the cryptosystem needs to perform encryption/decryption very fast to handle the enormous amount of data being transmitted inbound or outbound.

However, proposed ECCs consumes more area and resources in comparison with previous research works

In future, researchers may investigate the parallel implementation of ECC's upper computational level using NAF algorithm. Studying the performance and resources consumption levels for other forms of elliptic curves when implemented using proposed crypto processor architecture is another significant research direction.

REFERENCES

[1] Wade Trappe, Lawrence. c, Introduction to Cryptography with Coding Theory. Washington, Pearson Prentice Hall, 2002.

[2] N. Koblitz, "Elliptic curve cryptosystem", Mathematics of Computation, Vol. 48, pp. 203-209, 1987.

[3] V. Miller, "Uses of elliptic curves in cryptography", Lecture Notes in Computer Science, Vol. 218, pp. 417-426, 1986.

[4] Blake, Seroussi, and Smart. Elliptic Curves in Cryptography. Cambridge University Press: New York, 1999.

[5] Tanja Lange, "A note on L´opez-Dahab coordinates", Faculty of Mathematics, Technical University of Denmark, 2006.

[6] David, Nigel, and Jacques, "Projective coordinates Leak", Applied research and security center, France.

[7] GuerricMeurice de Dormale, Jean-Jacques Quisquater. High-speed hardware implementations of Elliptic Curve Cryptography: A survey. Journal of Systems Architecture 53 (2007) 72-84, by Elsevier.

[8] A. Satoh, K. Takano, "A scalable dual-field elliptic curve cryptographic processor," IEEE Transactions Computers 52 (4) (2003) 449–460.

[9] G. Orlando, and C. Paar, "A scalable GF (p) elliptic curve processor architecture for programmable hardware," Cryptographic Hardware and Embedded Systems - CHES 2001, Paris, France, May 14-15, 2001.

[10] Adnan Gutub and Mohammad K. Ibrahim., "High Radix Parallel Architecture For GF(p) Elliptic Curve Processor", IEEE Conference on Acoustics, Speech, and Signal Processing, ICASSP 2003, Pages: 625-628, Hong Kong, April 6-10.

[11] Adnan Gutub. Efficient Utilization of Scalable Multipliers in Parallel to Compute GF (p) Elliptic Curve Cryptographic Operations. Kuwait Journal of Science & Engineering (KJSE) 2007, 34(2): 165-182.

[12] L. Tawalbeh and Q. Abu Al-Haija. Speeding up Elliptic Curve Cryptography Computations by Adopting Edwards Curves over GF (P). International Journal of Security (IJS) 2009, CSC Journals, Malaysia, Vol.3, Issue.4, IJS-19.

[13] Q. Abu Al-Haija and Mohammad Al-Khatib. Parallel Hardware Algorithms & Designs for Elliptic Curves Cryptography to Improve Point Operations Computations Using New Projective Coordinates. Journal of Information Assurance and Security (JIAS) 2010, By Dynamic Publishers Inc., USA, Vol.4, No.1, Paper 6: (588-594).

[14] Adnan Abdul-Aziz Gutub and Mohammad K. Ibrahim, "High performance elliptic curve GF (2^k) crypto-processor architecture for multimedia", IEEE International Conference on Multimedia & Expo, ICME 2003, pages 81- 84, Baltimore, Maryland, USA, July 6-9, 2003.

[15] Adnan Gutub, "High Speed Low Power GF (2^k) Elliptic Curve Cryptography Processor Architecture", IEEE 10th Annual Technical Exchange Meeting, KFUPM, Dhahran, Saudi Arabia, March 23-24, 2003.

[16] L. Tawalbeh and Q. Abu Al-Haija. Enhanced FPGA Implementations for Doubling Oriented and Jacobi-Quartics Elliptic Curves Cryptography. Journal of Information Assurance and Security (JIAS), 2010, Volume 6, pp. 167-175.

[17] Mohammad Al-khatib, Qacem, and AzmiJaafar. Hardware Architecture & Designs for Projective Elliptic Curves Point Addition Operation using Variable Levels of Parallelism. International Review on Computers and Software 2011 Vol. 6 N. 2, pp. 237-243.

[18] Mohammad Al-Khatib, Q. Abu Al-Haija, and Ramlan Mahmud. Performance Evaluation of Projective Binary Edwards Elliptic Curve Computations with Parallel Architectures. Journal of Information Assurance and Security (JIAS) 2011, By Dynamic Publishers Inc., USA, Vol.6, No.1, Paper1: (001-009).

[19] Mohammad Al-khatib, AzmiJaafar, and Q. Sbu Al-Haija. Choices on Designing GF (p) Elliptic Curve Coprocessor Benefiting from Mapping Homogeneous Curves in Parallel Multiplications. International Journal on computer science and engineering 2011, Vol.3, No.2, Paper 2: (467-480).

[20] Mohammad Alkhatib, Azmi B. Jaafar, ZuriatiZukarnain, and Mohammad Rushdan. On the Design of Projective Binary Edwards Elliptic Curves over GF (p) Benefiting from Mapping Elliptic Curves Computations to Variable Degree of Parallel Design. International Journal on computer science and engineering 2011, Vol.3, No.4, Paper 44: (1697-1712).

[21] Mohammad Alkhatib, Azmi B. Jaafar, ZuriatiZukarnain, and Mohammad Rushdan, "Trade-off between Area and Speed for Projective Edwards Elliptic Curves Crypto-system over GF (p) using Parallel Hardware Designs and Architectures", International Review on Computers and Software, July 2011 Vol. 6 N. 4, pp. 163-173.

[22] Mohammad Al-khatib, AzmiJaafar, Zuriati Ahmad Zukarnain, and MohamadRushdanMd Said, "Hardware Designs and Architectures for Projective Montgomery ECC over GF (p) benefiting from mapping elliptic curve computations to different degrees of parallelism", International Review on Computers and Software, Vol. 6, N. 6, November 2011.

[23] Mohammad Al-khatib, and Adel Al-Salem. Efficient hardware implementations for Tripling Oriented Elliptic Curve Crypto-system. International Review on Computers and Software. 2014, Vol. 9, N. 4.

[24] A. Satoh, K. Takano, "A scalable dual-field elliptic curve cryptographic processor," IEEE Transactions Computers 52 (4) (2003) 449–460.

[25] T. Kerins, E. M. Popovici and W. P. Marnane. "An FPGA Implementation of a Flexible, Secure Elliptic Curve Cryptography Processor", International Workshop on Applied Reconfigurable Computing-ARC 2005, IADIS press, pp.22-30.

[26] Nele Mentens, Siddika Berna Ors, Bart Preneel, "An FPGA Implementation of an Elliptic Curve Processor over GF (2^m)", *In Proceedings of the 2004 ACM Great lakes Symposium on VLSI*, GLSVLSI 2004: VLSI in the Nanometer Era. pp. 454-457.

[27] Turki F. Al-Somani. Performance Evaluation of Elliptic Curve Projective Coordinates with Parallel GF (p) Field Operations and Side-Channel Atomicity. Journal of Computers 2010. JCP.5.1.99-109.

[28] Kocabas, U., J. Fan, and I. Verbauwhede, "Implementation of Binary Edwards curves for very-constrained devices," In 21st IEEE International Conference on Application-specific Systems Architectures and Processors, ASAP 2010, Rennes, France, pages 185 –191.

[29] Mohammad Alkhatib, "Cost-Effective Implementations for Weirstrass and Montgomery Elliptic Curve Crypto-systems". International Journal of Computer Science and Information Security (IJCSIS) 2016, Vol. 14 N. 9, pp. 98-109.

[30] Mohammad Alkhatib, "Improved ECC Performance Using NAF Algorithm for Binary Edward and Edward Elliptic Curves", IJCSNS International Journal of Computer Science and Network Security, 2019, Vol. 19 No. 6 pp. 98-109.

[31] I Khan, H Rehman, Mohammad Al-Khatib, Z Anwar, M Alam, "A thin client friendly trusted execution framework for infrastructure-as-a-service clouds", Future Generation Computer Systems 89, 239-248, 2017.

[32] Jin Y., Miyaji A. (2019) Secure and Compact Elliptic Curve Cryptosystems. In: Jang-Jaccard J., Guo F. (eds) Information Security and Privacy. ACISP 2019. Lecture Notes in Computer Science, vol 11547. Springer, Cham.

[33] Susella, R., Montrasio, S.: A compact and exception-free ladder for all short Weierstrass elliptic curves. In: Lemke-Rust, K., Tunstall, M. (eds.) CARDIS 2016. LNCS, vol. 10146, pp. 156–173. Springer, Cham (2017).

AUTHOR'S PROFILE

[1]Imam University, faculty of Computer and Information Sciences, Department of Computer Science.

Mohammad Al-khatibe is an assistant Professor in faculty of Computer and Information Sciences at Imam University. He received the bachelor degree in Computer Science. His Master degree was obtained from Depaul University in the field of Information Systems. He also achieved PhD in Computer Science (Security in Computing) from University PUTRA Malaysia (UPM). His research interest includes: information security, cryptography, and Elliptic Curve algorithm.