Journal of Computer Science

Original Research Paper

Accelerating Digital Forensics through Parallel Computing

Ghada Elkabbany, Mohamed Rasslan and Heba Aslan

Electronics Research Institute, Ministry of Higher Education and Scientific Research, Cairo, Egypt

Article history
Received: 24-12-2017
Revised: 25-01-2018
Accepted: 14-02-2018

Corresponding Author:
Ghada Elkabbany
Electronics Research Institute, Ministry of Higher Education and Scientific Research, Cairo, Egypt
Email: gelkabbany@eri.sci.eg
ghada_kabbany@yahoo.com

Abstract: Digital crimes in the era of big data and cloud computing imposes significant challenges in digital forensics. Cloud environment provides low cost, easy management and reasonable solutions. Moreover, it supports big data structures and solutions (i.e., security, privacy and digital forensics). In order to achieve a secure digital forensics analysis in cloud environment, researchers have proposed solutions with expensive communication cost and computation overheads. Among these solutions Nasereldin *et al.* proposed a protocol which solves the problem of authenticity and integrity of evidence using signcryption technique. This leads to low communication and implementation overheads. Furthermore, identity-based cryptography is used to solve Public Key Infrastructure (PKI) problems. In addition, it is characterized by the ability to divide the message into small messages which is suitable for pipelining techniques. Nasreldin *et al.*'s signcryption protocol is based on Elliptic Curve Cryptography (ECC) which is implemented by using different mathematical operations. In this protocol, ECC mathematical operations take huge time during the execution of the algorithm. ECC consists of point doubling and point addition operations. These operations require the execution of many Montgomery modular multiplications that consume time. In this study, we introduce a technique to speed up ECC operations in order to enhance the efficiency of Nasreldin *et al.* protocol. In particular, we propose a multi-stage parallel design which consists of three stages. First, we speed up the point doubling and point addition operation. Secondly, we enhance the execution time of Montgomery multiplications. Finally, pipelining is used to obtain a better performance. The results show that the proposed design enhances Nasreldin *et al.* protocol's execution time by 47.1, 64.7, 73.5 and 79.4%, assuming that the number of nodes is 2, 4, 6 and 12, respectively.

Keywords: Digital Forensics, Evidence Collecting, Signcryption, Elliptic Curve Cryptography, Parallel Computing

Introduction

Big data and cloud computing are hot topics that shape the future of both academia and industry. It is hard to acquire, handle, manage and process datasets in big data using legacy methods. Hence, big data requires optimal processing power and analytics capabilities. On the other hand, cloud computing offers a class of distributed data storage and processing platforms that provides on demand scalable and easy to use online resources in cost effective way. The extensive implementation raises the security and privacy anxieties. Cloud environment afford countless chances to criminals that allow them to misuse these new technologies by initiating attacks, capturing impeaching evidences and cracking encryption keys. The distributed computing power of big data in cloud environment makes the job of digital investigators more difficult in acquiring evidences for digital forensics purposes. Moreover, the amount of data generated through evidence acquisition is huge, complex and needs efficient analysis approaches in order to deal with its characteristics considering velocity and variety. Another problem could be raised while evidence collection, where the cloud administrator send the required data to the investigator. Therefore, it is crucial to protect the privacy of both uninvolved users and the investigation itself (Wall, 2007; Taylor *et al*., 2011; Fernandes *et al*., 2014; Lillis *et al*., 2016; Nasreldin *et al*., 2017; Hraiz, 2017; Samy *et al*., 2017).

In order to achieve a secure digital forensics analysis in cloud environment, researchers have proposed solutions with expensive communication cost and computation overheads (Hou *et al*., 2011; 2013a; 2013b; Zawoad and Hasan 2013; Zawoad *et al*., 2015; 2016). These solutions suffer from the lack of authenticity and integrity of evidence collected. To solve this problem Nasreldin *et al*. (2015a) proposed a solution which is based on using three blocks Sign-Encrypt-Sign. This solution suffers from computation, implementation and communication overheads. Signcryption techniques are used to solve this problem. In literature, many signcryption techniques which are based on Elliptic Curve Cryptography (ECC) are proposed (Zheng, 1997; Zheng and Imai, 1998; Deng and Bao, 1998; Jung *et al*., 2001; Han *et al*., 2004; Hwang *et al*., 2005b; Yuan and Hung, 2008; Toorani and Beheshti, 2009; Mohapatra, 2010; Ashraf *et al*., 2015; Nasreldin *et al*., 2015b; Singh, 2016).

ECC implies a set of point operations such as, point addition, point subtraction, point multiplication, point division and point doubling (Anoop, 2001). In these operations, the time complexity of point multiplication is higher than any other point operations. It is necessary to find out optimized implementations for point multiplication. Therefore, by using parallel computing, the implementation of point multiplication can be recovered to improve the performance of ECC (Sakthivel and Nedunchezhian, 2014). In literature, many solutions are proposed to improve the system performance. These solutions are divided into two categories: The first solution is based on parallelizing the different point operations, while, the second one is based on parallelizing the Montgomery modular multiplication operations.

Among the aforementioned solutions (Zheng, 1997; Zheng and Imai, 1998; Deng and Bao, 1998; Jung *et al*., 2001; Han *et al*., 2004; Hwang *et al*., 2005b; Yuan and Hung, 2008; Toorani and Beheshti, 2009; Mohapatra, 2010; Ashraf *et al*., 2015; Singh, 2016), Nasreldin *et al*. (2015b) proposed a protocol which solves the problem of authenticity and integrity of evidence with low communication and implementation overheads. Furthermore, it makes use of identity-based cryptography to solve Public Key Infrastructure (PKI) problems (such as: High storage cost, large bandwidth requirement, non-transparency to users and the need for Certificate Revocation Lists (CRLs)). In addition, it allows the message division into small messages which is suitable for pipelining techniques. In this protocol, ECC mathematical operations take huge time during the execution of the algorithm. In this study, we propose a multi-level parallel design in order to speed up Nasreldin *et al*.'s protocol execution time. The first level is based on parallelizing the operations required to perform the ECC point doubling and point addition, while the second one is used to enhance the execution time of Montgomery multiplications. Finally, pipelining is used to get a better performance. The results show that the proposed design enhances the execution time by 47.05, 69.12, 79.41, 86.03, 86.23 and 91.176% at the sender side assuming that the number of processors/nodes 'M' = 2, 4, 6, 12, 18 and 36, respectively. Moreover, in the receiver side, the degree of improvement is 47.1, 64.7, 73.5 and 79.4%, assuming that the number of nodes 'M' = 2, 4, 6, 12.

The remainder of this paper is organized as follows. In the next section, we give a review of digital forensics in cloud computing, Elliptic Curve Cryptography (ECC) and Nasereldin *et al*.'s protocol. Then, The proposed parallelization design of Nasereldin *et al*.'s protocol and its performance evaluation are presented. Finally, the conclusions are provided.

Background

Digital Forensics in Cloud Computing

Digital forensics is a particular form of auditing that has emerged in recent years to fight cybercrime (Fernandes *et al*., 2014). The development of this field has been motivated by the interest of organizations in audit tasks. It has the objective of determining potential digital evidence by means of analysis techniques. When applied to clouds, digital forensics face a complex scenario because data is pushed further back into the network and servers and is more spread out across them, rather than purely being on a physical computing device. Forensics also faces the data locality issues, making it hard to isolate particular resources. Zawoad *et al*. (2015; 2016; Zawoad and Hasan, 2013) proposed solutions which are based on the identification of the desired properties to support trustworthy forensics in the cloud. They proposed a Forensics Enabled Cloud (FECloud) architecture to maintain and afford required evidence. Unfortunately, they do not solve the authenticity and integrity of evidence problem.

Criminal investigation need to have the following characteristics: Protecting the privacy of involved users and keeping the administrator away from the investigation process. Hou *et al*. (2011; 2013a; 2013b) proposed several solutions which are based on administrator cooperation. Although, the administrator is responsible for protecting the data collection, he/she is not allowed to disclose this data. This solution's drawback is that the administrator cannot judge the relevance of data to the crimes under investigation. In addition, there is no guarantee that the data is not exposed to alteration or that it comes from the server (authenticity and integrity problems). To solve this problem, in (Hou *et al*., 2013b), they proposed an "encryption-then-blind signature with designated verifier" algorithm. They allow the administrator to search, retrieve and send the relevant data to the investigator in a secure manner. Nasreldin *et al*. (2015a), show that Hou *et al*. (2013b)'s scheme does not preserve its claimed integrity and authenticity. The common approach to achieve both evidence confidentiality and authenticity is to sign the evidence and encrypt it with its signature. The sender would sign the evidence using a digital signature scheme and then encrypt it with an appropriate encryption algorithm. The signature would use a private key encryption algorithm, under a randomly chosen message encryption key. The random evidence encryption key would then be encrypted using the recipient's public key. These are "sign-then-encrypt" or "encrypt-then-sign" techniques. Encrypt-then-sign is subject to the plaintext-subsection and text stealing attacks. The composition of the sign-then-encrypt approach suffers from a forwarding attack (Zheng and Imai, 1998). To mitigate these security breaches, Sign-Encrypt-Sign and Encrypt-Sign-Encrypt techniques are used. Sign-Encrypt-Sign and Encrypt-Sign-Encrypt suffers from computation, implementation and communication overheads. The term signcryption was originally introduced and studied by Zheng (1997) with the primary goal of reaching greater efficiency than can be accomplished when performing the signature and encryption operations separately. In spite of proposing some security arguments, most of the work on signcryption (Zheng, 1997) missed formal definitions and analysis. Moreover, signcryption schemes must achieve non-repudiation, which guarantees that the sender of a message cannot later repudiate that he/she has sent the message. Namely, the recipient of a message can convince a third party that the sender indeed sent the message. It is worth noting that typical signature schemes provide non-repudiation, since anyone, who knows only the sender's public key, can verify the signature. This is not the case for signcryption, because the confidentiality property entails that only the recipient can comprehend the contents of a signcrypted message sent to him/her. Nevertheless, it is feasible to accomplish non-repudiation by other means. Instead of using encryption/signing process, signcryption can be applied in place of separate encryption and signing to reduce both communication bandwidth and computational time overheads. Any authentication scheme for big data streams should verify the received packets without assuming the availability of the entire original stream.

Zheng (1997) proposed the first signcryption scheme based on discrete logarithmic problem. It saved about 50% computational cost and about 85% communication cost than the traditional signature-then-encryption scheme, but it fails the forward secrecy of message confidentiality. Deng and Bao (1998) improved Zheng's scheme such that the judge can verify signature without the recipient's private key. But a key exchange protocol was required in the process of verification. At, Zheng and Imai (1998) suggested an ECC based signcryption scheme that provided all the basic security features and saved about 58% computational cost and 40% communication cost than signature-then-encryption. As it is based on ECC the key size used was smaller as compared to the other schemes. This was one of the advantages of this scheme but it still needs forward secrecy (Jung *et al*., 2001). Hwang *et al*. (2005b) proposed a signcryption scheme based on elliptic curve with forward secrecy and publicly verifiable. This scheme satisfied the message confidentiality of previous encrypted message even if the sender divulged his private key inattentively with a cost comparable to the existing schemes. Toorani and Beheshti (2009) suggested a signcryption scheme based on elliptic curve which provide all the security attributes. But this scheme required more computational cost as compared to existing schemes. Singh (2016) proposed a signcryption scheme which provides encrypted message authentication, forward secrecy and public verification. The disadvantage of this scheme is that it still requires a comparable computational and communication costs.

Nasreldin *et al*. (2015b) propose an identity-based signcryption protocol to reduce the computation, communication and implementation overheads in evidence collection in cloud forensics. Their proposed protocol is more efficient than all the previously presented protocols. It allows the receiver (verifier) to restore the message blocks upon receiving their corresponding signature blocks. In addition, it is perfect for some application requirements and fits packet switched networks. This protocol has two stages of verification to ensure that the message has been recovered efficiently and correctly. The first verification step is to ensure the integrity and authenticity of the message (e.g., no modification or substitution in the ciphertext 'r_i'). The second verification step is to ensure that the i^{th} message is reconstructed successfully. This stage is useful for public verification in the case of a dispute takes place. It guarantees that Nasereldin *et al*. protocol satisfies the non-repudiation property. Nasreldin *et al*. (2015b) show that the security of their protocol is based on the intractability of reversing the secure cryptographic hash function and the Elliptic Curve Discrete Logarithm (ECDL) problem. Moreover, they analyzed the security of their protocol in terms of authenticity, unforgeability, confidentiality, non-repudiation and forward secrecy. As mentioned previously, the signcryption protocols (Zheng, 1997; Zheng and Imai, 1998; Deng and Bao, 1998; Jung *et al*., 2001; Han *et al*., 2004; Hwang *et al*., 2005b; Yuan and

Hung, 2008; Toorani and Beheshti, 2009; Mohapatra, 2010; Ashraf *et al.*, 2015; Nasreldin *et al.*, 2015b; Singh 2016) are based on ECC that will be described in details in the next subsection.

Elliptic Curve Cryptography ECC

Public key cryptography achieves evidence confidentiality, authenticity, non-repudiation and integrity. ECC is a better choice than RSA as it provides the same security level for shorter keys. For the last decade, ECC has gained increasing acceptance in the industry and the academic community and has been the subject of several standards. This interest is mainly due to the high level of security with relatively small keys, low cost and smaller hardware realization provided by ECC (Hwang *et al.*, 2005a; Meurice de Dormale and Quisquater, 2007; Lo *et al.*, 2010; Li *et al.*, 2013). It was first proposed independently by Koblitz (1987) and Miller (1985). The security of a public key system using elliptic curves is based on the difficulty of computing discrete logarithm in a group of points on an elliptic curve defined over a finite field (FOSIT, 2000). ECC is used in many applications such as smart cards, set top box, low power portable devices (cell phone), etc. In all these applications, the main operation in ECC is the scalar multiplication in authentication and certification (Thomas *et al.*, 2014). The Elliptic Curve Discrete Logarithm Problem (ECDLP) is currently believed to be asymptotically harder than the factorization of integers. ECC provides more security per key bit compared to other public key standards (Rao *et al.*, 2017; Parmar and Verma, 2017). Table 1 shows the key sizes of AES, ECC and RSA for the same security level. Private keys are 12-times larger for RSA compared to ECC at the 128-bit security level; as shown in Table 1.

ECC could work in GF(2^m) or GF(p), while GF(2^m) is suitable for hardware implementation, GF(p) is suitable for software implementation (Miller, 1985; FOSIT, 2000; Sakthivel and Nedunchezhian, 2014). In our work, we concentrate on parallelizing GF(p). Cryptographic schemes based on ECC rely on scalar multiplication of elliptic curve points. Given an integer 'k' and a point "$P \in E(F(p))$", scalar multiplication is the process of adding 'P' to itself 'k' times. The result of this scalar multiplication is denoted by 'kP'. Scalar multiplication of ECC can be computed efficiently using the double-and-add algorithm-as given in the following:

```
N = P; and R = O;                //point at infinity
for(i = 0; i < k-bit-length; i++)
{If(k[i] == 1)
        R = R + N;
            N = N + N;}
```

In this algorithm, 'O' represents point at infinity and k-bit-length represents the number of bits of 'k'. Scalar multiplication is used for the computation of the public key, the signature, encryption and key agreement in the ECC system. The mathematical operations of the ECC are defined over the elliptic curve are as follows:

$$y^2 \equiv x^3 + a.x + b(mod\ p) \quad (1)$$

where:

$$4.a^3 + 27.b^2 \neq 0(mod\ p).$$

The change of the parameters 'a' and 'b' gives different elliptic curves (Certicom Corp., 2000a; 2000b; Tawalbeh *et al.*, 2010; Srivastava and Mathur, 2013). One of the crucial decisions when implementing an efficient ECC over GF(p) is deciding which point coordinates system to use. In (Tawalbeh *et al.*, 2010), details of three different projective coordinate systems are given. The first one is the affine coordinate where a point is represented as (X_A,Y_A). The other two forms of the projective coordinates are: (X,Y) where $X_A = X/Z$ and $Y_A = Y/Z$ and (X,Y) where $X_A = X/Z^2$ and $Y_A = Y/Z^3$. Table 2 shows a comparison of these three projective coordinate systems. As shown in the table, the affine coordinate system uses inversion operation in both point addition and point doubling, which is costly in terms of computation time and makes it an inefficient choice. The other coordinate systems do not use modular inversions in point addition and doubling. As mentioned in (Tawalbeh *et al.*, 2010), the projection (X, Y) where $X_A = X/Z^2$ and $Y_A = Y/Z^3$ has the minimum number of modular multiplication operations.

Table 1: Keys sizes of ECC Vs RSA and AES (Malik, 2010)

AES (bits)	ECC (bits)	RSA (bits)
80	160	1024
128	256	3024
192	384	7680
256	512	16360

Table 2: Comparison between the three coordinates systems

Coordinates system	Adding	Doubling
Affine	6 Add +3 Mul + Inv	4 Add +4 Mul + Inv
Projective(x,y) ⇒ (X/Z^2,Y/Z^3)	6 Add +16 Mul	4 Add +10 Mul
Projective(x,y) ⇒ (X/Z, Y/Z)	6 Add +15 Mul	4 Add +12 Mul

For the projective coordinate system $(x,y) \Rightarrow (X/Z^2, Y/Z^3)$, point addition of $P + Q$ in projective coordinates $(x, y) \Rightarrow (X/Z^2, Y/Z^3)$ is computed as:

$P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$, $P + Q = (X_3, Y_3, Z_3)$, where $P \neq \pm Q$
$(x, y) = (X/Z^2, Y/Z^3) \rightarrow (X, Y, Z)$
$\lambda_1 = X_1Z_2^2$ $\quad \lambda_2 = X_2Z_1^2$ $\quad \lambda_3 = \lambda_1 - \lambda_2$
$\lambda_4 = Y_1Z_2^3$ $\quad \lambda_5 = Y_2Z_1^3$ $\quad \lambda_6 = \lambda_4 - \lambda_5$
$\lambda_7 = \lambda_1 + \lambda_2$ $\quad \lambda_8 = \lambda_4 + \lambda_5$
$Z_3 = Z_1Z_2\,\lambda_3$ $\quad X_3 = \lambda_6^2\,\lambda_3^2$ $\quad \lambda_9 = \lambda_7\,\lambda_3^2 - 2X^3$
$Y_3 = (\lambda_9\,\lambda_6 - \lambda_8\,\lambda_3^3)/2$

The doubling of a point $(P + P)$ is computed as:

$P = (X_1, Y_1, Z_1)$; $P + P = (X_3, Y_3, Z_3)$
$(x, y) = (X/Z^2, Y/Z^3) \rightarrow (X, Y, Z)$
$\lambda_1 = 3X_1^2 + aZ_1^4$ $\quad Z_3 = 2Y_1Z_1$ $\quad \lambda_2 = 4X_1Y_1^2$
$X_3 = \lambda_1^2 - 2\,\lambda_2$ $\quad \lambda_3 = 8Y_1^4\lambda_4 = \lambda_2 - X_3$
$Y_3 = \lambda_1\,\lambda_4 - X_3$

In next sub-section, we give a detailed description of (Nasreldin *et al*., 2015b) evidence acquisition protocol.

Nasreldin et al.'s Evidence Acquisition Protocol

Nasreldin *et al*. (2015b), in their work, proposed an identity-based signcryption protocol to solve the problem of authenticity and integrity of collected evidences. Nasereldin *et al*.'s protocol makes use of identity-based cryptography to overcome PKI problems mentioned previously. Although this protocol needs larger number of Elliptic Curve Point Multiplication (ECPM) operations than other protocols (Zheng and Imai, 1998; Han *et al*., 2004; Hwang *et al*., 2005b; Toorani and Beheshti, 2009; Mohapatra, 2010; Singh, 2016) (as shown in Fig. 1), Nasreldin *et al*.'s (2015b) protocol allows message to be divided into small messages which is suitable for pipelining techniques. Moreover, it allows the recipient to restore the message blocks upon receiving their corresponding signature blocks. It consists of two stages of verification: The first stage is to ensure the integrity and authenticity of the message. The second stage is to make sure that the message is reconstructed successfully. This leads to guarantee that the protocol satisfies the non-repudiation property.

In order to perform Nasreldin *et al*.'s protocol, the following steps must be performed.

Setup

The Private Key Generation center (PKG) chooses a Gap Diffie-Hellman group 'G_1' of prime order 'q', a multiplicative group 'G_2' of the same order and a bilinear map "e: $G_1 \times G_1 \rightarrow G_2$", together with an arbitrary generator $P \in G_1$. Then it chooses a random value "$s \in Z_q^*$" as the master secret key and computes the corresponding public key "$P_{pub} = sP.H_1$" and 'H_2' are two secure cryptographic hash functions, such that "H_1: 0, 1* $\rightarrow G_1$" and "H_2: 0, 1* $\rightarrow Z_q^*$". The system parameters (G_1, G_2, P, P_{pub}, H_1, H_2, e, q) and the master secret key is 's'.

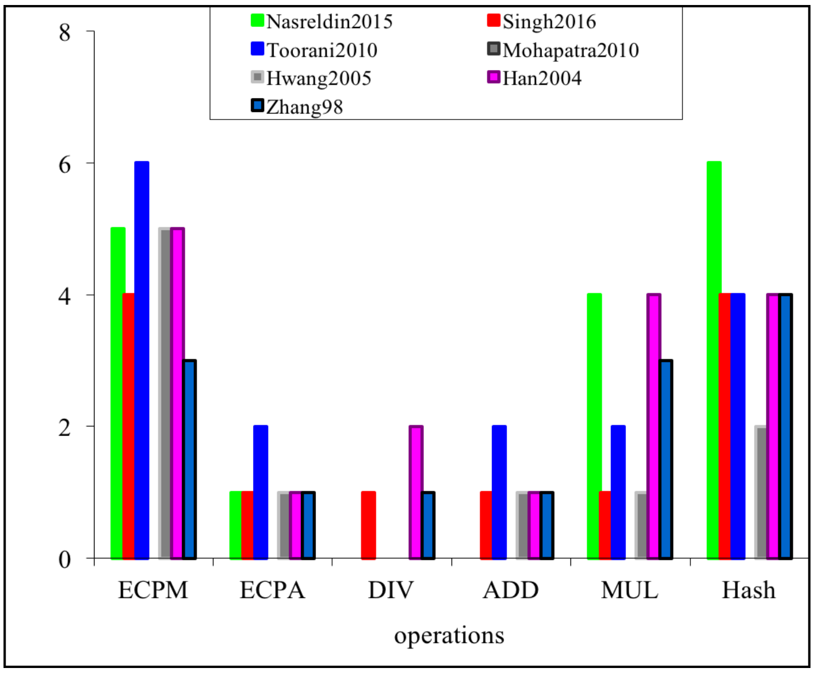


Fig. 1: Comparative analysis of computational cost of different signcryption schemes; ECPA: Elliptic Curve Point Addition; ECPS: Elliptic Curve Point Subtraction; ECPM: Elliptic Curve Point Multiplication operations

KeyExtract

Given identity *ID*, PKG computes "$S_{ID} = sH_1(ID$" and sends it to the user with identity *ID*. Nasreldin *et al.*'s protocol defines 'Q_{ID}' as the public key of the user with identity *ID*. In addition, it assumes that the sender 'A' (with secret key 'S_A' and public key 'Q_A') wants to send a message 'Mess' to the receiver '*B*' (with public key 'Q_B' and secret key 'S_B'), it divides the stream into blocks, '$Mess_i$', where $Mess_i \in Z_q^*$.

Signcrypt Operation (Sender Side)

The sender 'A' chooses a random number $k \in Z_q^*$ and lets $r_0 = 0$. The following steps must be done at the sender side before sending the signcrypted message:

$$r_i = Mess_i \cdot H_2\left(r_{i-1} \oplus e\left(P, Q_B\right)^k\right), for\ i = 1,2,3,\ldots,n \quad (2)$$

$$\alpha = H_2\left(r_1,\ \ldots\ ,\ r_n,\ e\left(P, Q_B\right)^k\right) \quad (3)$$

$$\beta = H_2\left(Mess_1,\ \ldots\ ,\ Mess_n, \alpha,\ e\left(P,\ P\right)^k\right) \quad (4)$$

$$\gamma = \beta \cdot P \quad (5)$$

$$\theta = \beta \cdot Q_B \quad (6)$$

$$S = \beta^{-1} \cdot k \cdot P - \beta^{-1} \cdot S_A \quad (7)$$

'*A*' sends (S, α, γ, θ, $r_1,\ldots, r_n$) to '*B*' over a non-secure channel.

Unsigncrypt Operation (Receiver Side):

- Verifies: $\alpha \overset{?}{=} H_2\ (r_1,\ldots,r_n,e(S,\theta)\cdot e(S_B,Q_A))$
- Recovers Mess:

$$Mess_i = r_i \cdot \left[H_2\left(r_{i-1} \oplus \left[e\left(S,\theta\right).e\left(S_B,Q_A\right)\right]\right)\right]^{-1} \quad (8)$$

- Checks:

$$\gamma^? = H_2\left(Mess_1, Mess_2,\ldots,Mess_n,\alpha,e\left(S,\lambda\right).e\left(P_{pub},Q_A\right)\right).P \quad (9)$$

Upon receiving the message, the receiver verifies the signature by making the comparison between: 'α' and "$Mess_i{\cdot}H_2\ (r_{i-1} \oplus e\ (P,\ Q_B)^k)$". In case of they are not equal, this implies that the received packets are altered and must be ignored. On the other hand, if they are equal, then the receiver retrieves the message blocks $Mess_i = r_i[H_2(r_{i-1}\oplus[e(S,\theta).e(S_B,Q_A)])]^{-1}$. Lastly, the recipient checks the correctness of the message reconstruction by comparing 'γ' to $H_2(Mess_i,\ldots, Mess_n, \alpha,e(S,\gamma).e(P_{pub},Q_A)).P$. For public verification, the receiver 'B' only needs to make the following public (Mess, S, α, γ, θ). Next, any verifier can check the message authenticity by comparing 'γ' to $H_2(Mess_i,\ldots, Mess_n, \alpha,e(S,\gamma).e(P_{pub},Q_A)).P$.

In the next section a parallel implementation of Nasreldin *et al.* protocol (2015b) is presented.

Methods

The Proposed Parallel Design of Nasreldin et al.'s Protocol

Nasreldin *et al.*'s protocol is based on ECC which is characterized by different mathematical point operations that take huge time during its execution. Among these operations, the time complexity of ECPM is higher than any other point operations on elliptic curve (Tawalbeh *et al.*, 2010). Therefore, by using parallel computation, the implementation of EPCM can be accelerated to improve the performance of ECC. Therefore, in order to accelerate Nasreldin *et al.*'s protocol, a multi-level parallel model is presented. Our proposed design consists of three levels: The first level is based on computing different point doubling and point addition operations of each ECPM operation in parallel, while the second one is used to enhance the execution time of Montgomery multiplications. Finally, different message blocks are pipelined.

Parallel Elliptic Curve Cryptography

Parallelizing ECC algorithms is a promising approach that can be used to reduce its computation time. Several research studies in the literature concerning parallelizing ECC over prime field GF(p) are given. These solutions are divided into two categories: The first solution is based on parallelizing the different point operations (Srivastava and Mathur, 2013; Anagreh *et al.*, 2014; Chung *et al.*, 2012; Gutub *et al.*, 2007). The other research direction is based on partitioning the Montgomery modular multiplication (Fan *et al.*, 2008; Guillermin, 2010). In this study, a hybrid parallel solution that makes use of the advantages of both categories is proposed. First, different operations of each ECMP (consists of point doubling and point addition operations) are computed in parallel. Then, the Montgomery modular multiplication operations are executed in parallel in order to enhance the execution time.

As mentioned at a previous section, the projection (X,Y) where $X_A = X/Z^2$ and $Y_A = Y/Z^3$ has the minimum number of modular multiplication operations. The dataflow graphs for point adding and point doubling are shown in Fig. 2 and 3 respectively.

As mentioned in Table 2, each point addition operation needs sixteen modular multiplications and six modular additions. On the other hand, each doubling operation needs ten modular multiplications and four modular additions. Assuming that, 'T_M' is the time needed to execute one modular multiplication operation and 'T_A' is the time needed to compute one modular addition operation respectively (for simplicity, we assume that the time needed to execute modular subtraction operation equals to that needed for modular addition operation). Then, the total execution time needed to execute each point addition operation '$T_{S\text{-}add}$' is given by:

$$T_{S-add} = 16T_M + 6T_A \tag{10}$$

On the other hand, the total execution time '$T_{S\text{-}doub}$' that is needed to compute each point doubling operation is given by:

$$T_{S-doub} = 10T_M + 4T_A \tag{11}$$

As mentioned in (Miller, 1985), field multiplication is the basic elliptic curve operation used in computing the point 'kP' from 'P'. Assuming that 'n' is the number of bits of 'k' which indicates the exact number of point doublings, but not point additions. Assuming that the bits of 'k' are half ones and half zeros (an average estimation for comparison reason), then the elliptic curve arithmetic operations required are 'n' point doublings and approximately '$n/2$' point additions. Then, the total sequential time of the elliptic curve point multiplication arithmetic operation '$T_{S\text{-}ECPM}$' is calculated as follows:

$$\begin{aligned} T_{S-ECPM} &= (T_{S-add})*(n/2)+(T_{S-doub})*n \\ &= (16T_M + 6T_A)*(n/2)+(10T_M + 4T_A)*n \\ &= (18n)T_M + (7n)T_A \end{aligned} \tag{12}$$

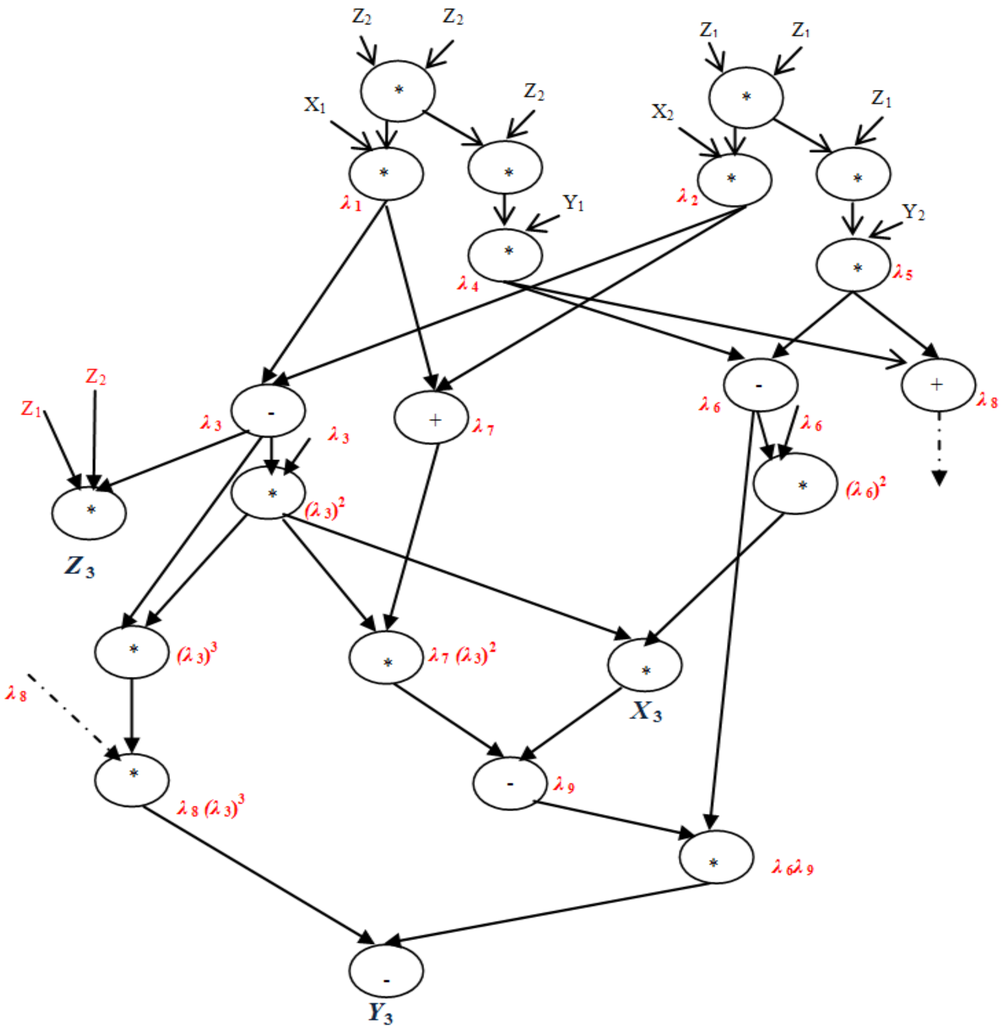


Fig. 2: Projective coordinate $(x, y) \Rightarrow (X/Z^2, Y/Z^3)$: Point Addition

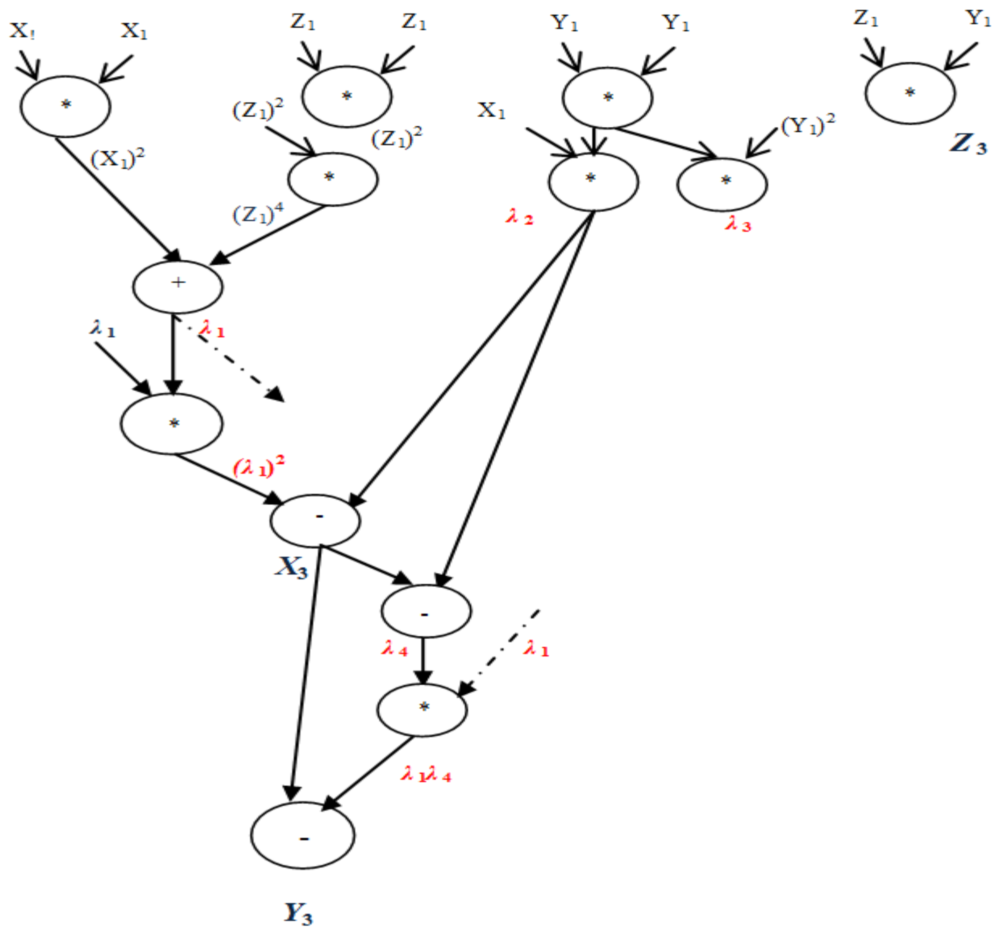


Fig. 3: Projective coordinate $(x, y)\Rightarrow(X/Z^2, Y/Z^3)$: Point Doubling

For the first level of parallelization, different point doubling and point addition operations (for each ECMP) operation, are computed in parallel. As shown in Fig. 2 and 3, there is some dependency in calculating both point doubling and point addition. Therefore, the maximum number of nodes that can be used to execute each ECPM is four.

Both point addition and point doubling operations require the execution of many Montgomery multiplications which consume time. This led us to propose the next level of our parallel model that is concerned of enhancing the execution time of Montgomery multiplications. Each modular multiplication operation can be represented by three simple multiplication operations and one simple addition operation (GroBschadl, 2000); therefore each modular multiplication operation can be executed in parallel. The optimal number of nodes to execute one modular multiplication is three. This level of parallelism enhances the ECC performance, since it solves the problem of load imbalance (Elkabbany *et al*., 2014). Then, to achieve load balancing, each ECPM operation can be computed by at most twelve nodes.

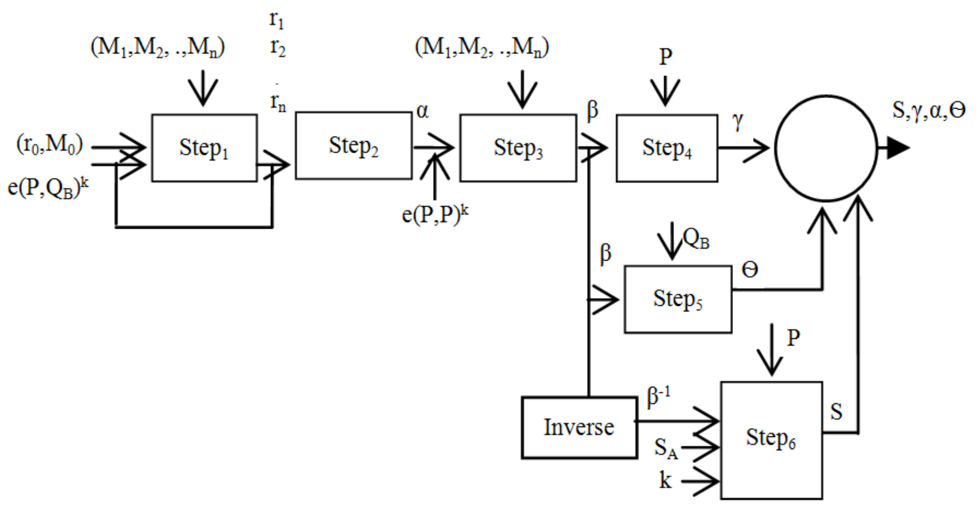
Assuming that, the time needed to compute a simple multiplication operation equals to 't_m' and the time needed for computing simple addition operation equals to 't_a'. Then, a modular multiplication operation can be calculated as:

$$T_M = 3t_m + t_a \tag{13}$$

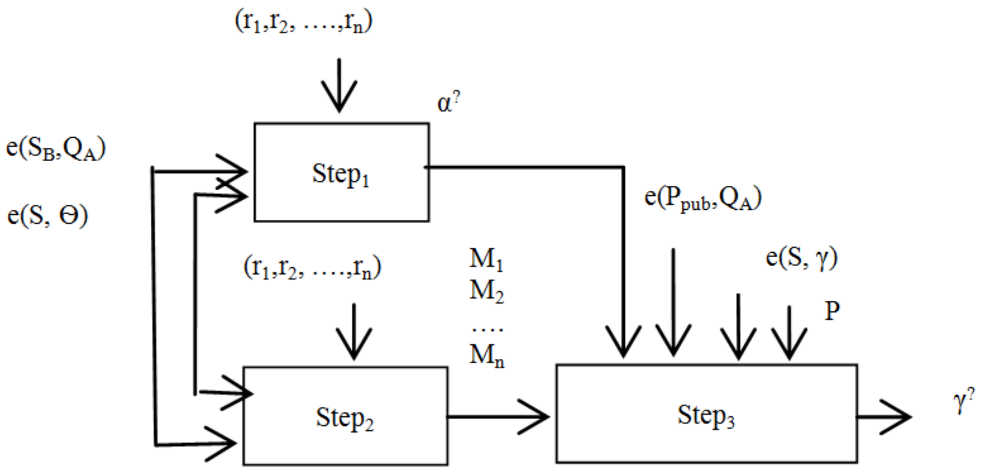
In addition, the modular addition could be calculated as the summation of addition and modulo operations. Using Barret algorithm (Barret, 1987), modulo operation needs one simple multiplication, one simple division and one simple subtraction. Then, the time needed to compute modular addition operation can be calculated as follows:

$$T_A = t_m + 2t_a + t_{div} \tag{14}$$

where, 't_{div}' is the time needed to compute one simple division.



(a)



(b)

Fig. 4: (a) Nasreldin *et al.*'s protocol steps for signcryption; (b) Nasreldin *et al.*'s protocol steps for unsigncryption

Since, the addition operation considerably needs less time than the multiplication operation, it can be neglected and assuming that and "$t_{div} = t_m$", therefore, the time needed to execute each modular multiplication is '$3t_m$' and the time needed to execute one modular addition is '$2t_m$'. Then, from Equation 12 to 14, the sequential time for each ECPM '$T_{S\text{-}ECPM}$' is calculated as follows:

$$T_{S-ECPM} = 68(nt_m) \qquad (15)$$

Due to the nature of Nasreldin *et al.* (2015b) protocol, the proposed parallel design assumes that the data stream is divided into 'N' messages, which can be executed in a pipelined manner. For simplicity, we assume that the number of pipeline stages equals to the number of steps to be executed and the output is shifted from step 'i' to step 'I +1'for all steps. As mentioned previously, Nasreldin *et al.* (2015b) requires four ECPM, in case of signcryption and only one ECPM in case of unsigncryption. Figure 4, presented different Steps of both sender and receiver sides. From this figure, we can noticed that: At the sender side, parallelization can be done within Step 1 that has one ECPM. In addition, Steps 4, 5 and 6 can be done in parallel (each has one ECPM). While at the receiver side, parallelization can be done only at Step 3 that has only one ECPM. Since, the maximum number of nodes that can be used for each ECPM is twelve. Then, for the signcryption operation (at the sender side), thirty six nodes are needed, while at the receiver side only twelve nodes are needed.

In order to simplify the calculations, we assume that the time needed for Add, Sub, Mul, Div and Hash operations will be neglected as they are very small compared to the time required for the ECPM operations and from Equation 15, the total sequential time at both sender and receiver side '$T_{s\text{-}sender}$' and '$T_{s\text{-}receiver}$' can be calculated as:

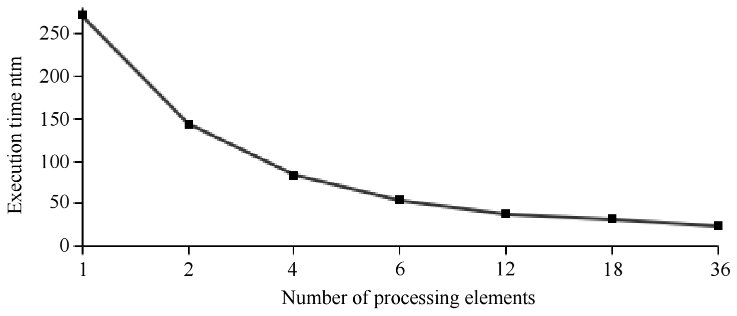
$$T_{s-sender} = 4 * T_s = 4 * 68(nt_m) = 272(nt_m) \quad (16)$$

$$T_{s-receiver} = 1 * T_s = 68(nt_m) = 68(nt_m) \quad (17)$$

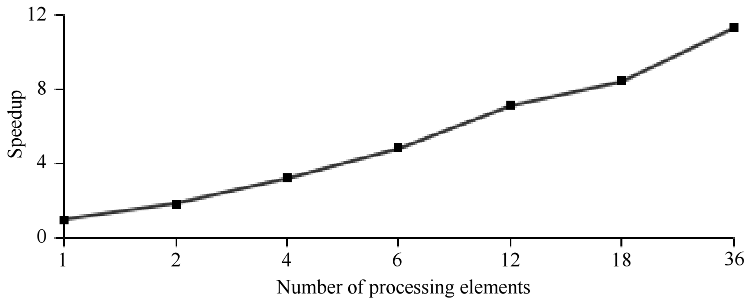
Results and Discussion

To evaluate the performance of the proposed parallel model, different metrics such as: Execution time, speed up, efficiency and the improvement degree are used (Borisenko, 2010; Zaghloul *et al*., 2017). Parallel execution time 'Tpar' can be defined as the time period between the starting of parallel computation and the time since the last processor/node finishes execution. Furthermore, the speedup can be defined as the ration between the sequential and parallel times "*Ts*/*Tpar*". Moreover, degree of improvement is determined by: "(*Ts*-*Tpar*)/*Ts*".

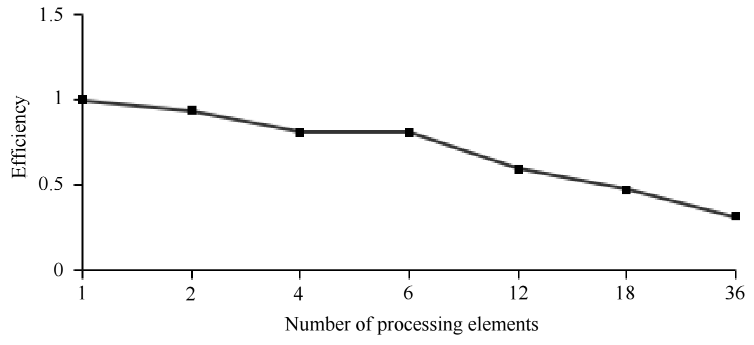
Table 3 illustrates the parallel execution time for both point addition and point doubling operations at each ECMP operation. In order to simplify the calculations, we will neglect the communication time as it is small compared to the time required to compute modular operations and then, Table 4 presents the parallel time of each ECPM operation '$T_{ECPM\text{-}par}$', for different number of nodes '*M*' = 2, 4, 6 and 12. Finally, Table 5 shows the total parallel execution time of the proposed parallel design of Nasereldin *et al*.'s protocol at both sender and receiver sides for '*M*'= 1, 2, 4, 6, 12, 18 and 36. On the other hand, Fig. 5 and 6 present the system performance: Execution time, speed up, efficiency and the improvement degree at the sender and the receiver respectively.



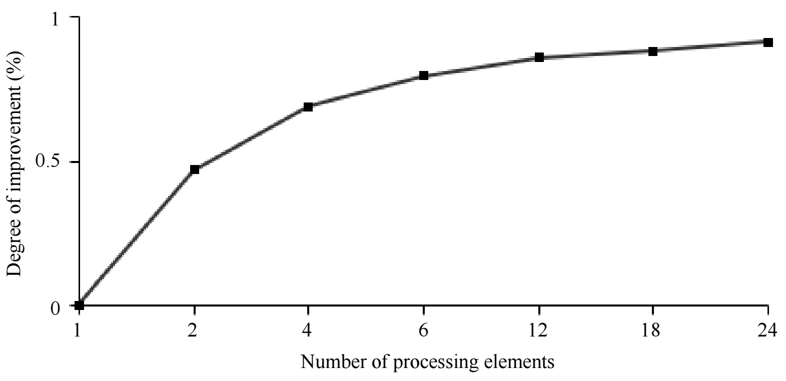
(a)



(b)

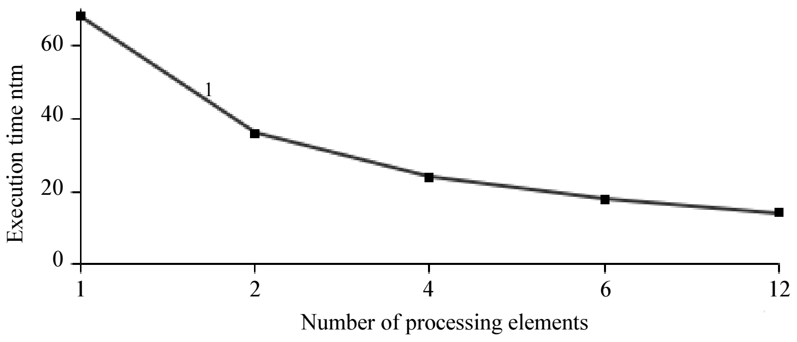


(c)

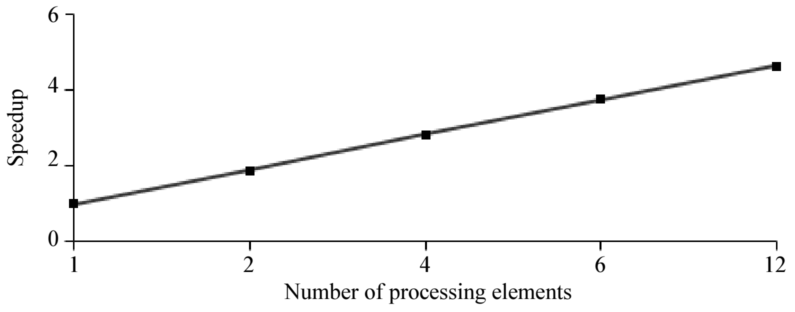


(d)

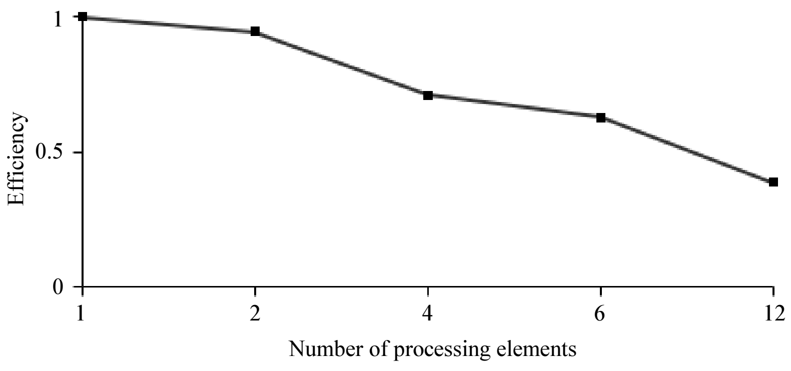
Fig. 5: System performance of the proposed parallel model (sender side)



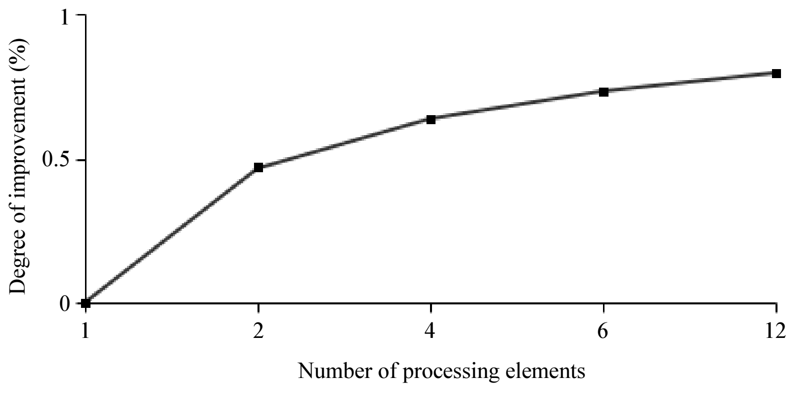
(a)



(b)



(c)



(d)

Fig. 6: System performance the proposed parallel model (receiver side)

Table 3: The parallel time for both point addition and point doubling operations when using $M = 2$ and 4

M	T_{a-par} (point-addition)	T_{d-par} (point-doubling)
2	$T_{a-par} = 8T_M + 3T_A + T_{comm}$	$T_{d-par} = 5T_M + 3T_A + T_{comm}$
4	$T_{a-par} = 4T_M + 3T_A + T_{comm}$	$T_{d-par} = 3T_M + 3T_A + T_{comm}$

Table 4: The parallel time of each ECPM using different number of nodes (M = 2, 4, 6 and 12)

M	$T_{ECPM-par} = (T_{a-par})*n/2 + (T_{d-par})*n$
2	$T_{ECPM-par} = n*(9T_M + 4.5T_A) = 36\ (nt_m)$
4	$T_{ECPM-par} = n*(5T_M + 4.5T_A) = 24\ (nt_m)$
6	$T_{ECPM-par} = n*(9t_m + 4.5T_A) = 18(nt_m)$
12	$T_{ECPM-par} = n*(5t_m + 4.5T_A) = 14(nt_m)$

Table 5: The total parallel time of the proposed parallel model of Nasreldin *et al.*'s protocol at both sender and receiver

M	Sender side	Receiver side
1	$272(nt_m)$	$68(nt_m)$
2	$144(nt_m)$	$36(nt_m)$
4	$84(nt_m)$	$24(nt_m)$
6	$54(nt_m)$	$18(nt_m)$
12	$38(nt_m)$	$14(nt_m)$
18	$32(nt_m)$	$14(nt_m)$
36	$24(nt_m)$	$14(nt_m)$

As shown in the above tables and figures, it is clear that the use of parallel system decreases significantly the execution time of Nasreldin *et al.*'s protocol. Figure 5a and 6a show that, as the number of nodes increases, the total execution/ parallel time decreases. Moreover, as the number of nodes increases, the speedup increases as shown in Fig. 5b and 6b. Figure 5c and 6c present the efficiency of the proposed parallel design. Parallel efficiency is the ratio between speedup and the number of nodes. It estimates how well the nodes are used in solving the problem. These figures illustrate an overall decrease in parallel efficiency achieved by the parallel model as the number of nodes increases. Figure 5d and 6d describe the improvement of the proposed parallel design compared to the performance prior to parallelization. As shown in these figures, as the number of nodes increases, the improvement degree increases. The degree of improvement at the sender side is 47.05, 69.12, 79.41, 86.03, 86.23 and 91.176% assuming that '*M*' = 2, 4, 6, 12,18 and 36 respectively. Moreover, in the receiver side, the degree of improvement is 47.1, 64.7, 73.5 and 79.4%, for 2, 4, 6 and 12 nodes respectively. Increasing the number of processors/nodes leads to the decrease in the system's efficiency. Therefore, the number of nodes must not exceed a certain number which is called system's saturation. As shown in Fig. 5 and 6, the saturation occurs when the number of processors equals 36 and 12 at sender and receiver sides respectively.

Conclusion

Nasreldin *et al.* proposed a protocol for securing the digital evidence collection in cloud environments. This protocol solves the problem of authenticity and integrity of evidence with the following characteristics: It has low communication and implementation overheads. Furthermore, it makes use of identity-based cryptography to solve PKI problems such as: High storage cost, large bandwidth requirement, non-transparency to users and the need for CRLs. In addition, it allows the message division into small messages which is suitable for pipelining techniques. In this study, a multi-level parallelism model is presented in order to accelerate Nasreldin *et al.*'s protocol. In their protocol, ECC mathematical operations take a huge time during the execution of the protocol. ECC is implemented by using a set of point operations, in these operations, the time complexity of ECPM is higher than any other point operations on elliptic curve. Therefore, by using parallel computation the implementation of EPCM can be accelerated to improve the performance of ECC. Since the ECPM is the most consuming time, then reducing its time will improve the Montgomery multiplication's performance.

Our design consists of three levels of parallelization: The first level is based on computing different point doubling and point addition operations in parallel, while the second one is used to enhance the execution time of Montgomery multiplications. Finally, pipelining different message blocks is used to get a better performance. The analysis shows that the use of parallel system will enhance its performance. The experimental results show that the maximum number of nodes that can be used for each ECPM is twelve. Then, for the signcryption operation (at the sender side), thirty-six nodes are needed. While, for unsigncryption operation (receiver side) only twelve nodes are needed. At the sender side, the degree of improvement of the proposed parallel design, compared to the performance prior to parallelization is 47.05, 69.12, 79.41, 86.03, 86.23 and 91.176% assuming that '*M*' = 2, 4, 6, 12, 18 and 36 respectively. On the other hand, at the receiver side, the degree of improvement is 47.1, 64.7, 73.5 and 79.4%, assuming that the number of nodes '*M*' = 2, 4, 6 and 12.

Author's Contributions

The author prepared the study, elaborated the methodology, performed the analysis and wrote the manuscript.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that no ethical issues involved.

References

Anagreh, M., A. Samsudin and M. Omar, 2014. Parallel method for computing elliptic curve scalar multiplication based on MOF. Int. Arab J. Inform. Technol., 11: 521-525.

Anoop, M.S., 2001. Elliptic curve cryptography an implementation tutorial. Technical Report, Tata Elxsi Ltd, Thiruvananthapuram, India.

Ashraf, S., N. Uddin, M. Sher, A. Ghani and H. Naqvi *et al.*, 2015. An efficient signcryption scheme with forward secrecy and public verifiability based on hyper elliptic curve cryptography. Multimed. Tools Applied, 74: 1711-1723. DOI: 10.1007/s11042-014-2283-9

Barret, P., 1987. Implementing the rivest shamir and adleman public key encryption algorithm on standard digital signal processor. Proceedings of the Advances in Cryptolgy, (AC' 87), Springer, Santa Barbara, California, pp: 311-323. DOI: 10.1007/3-540-47721-7_24

Borisenko, A., 2010. Performance evaluation in parallel systems. ACM Sigplan Notes, 17: 150-155.

Certicom Corp., 2000a. SEC1: Elliptic curve cryptography standards for efficient cryptography. Technical Report, Ontario, Canada.

Certicom Corp., 2000b. SEC2: Recommended elliptic curve domain parameters standards for efficient cryptography. Technical Report, Ontario, Canada.

Chung, S., J. Lee, H. Chang and C. Lee, 2012. A high-performance elliptic curve cryptographic processor over GF(p) with SPA resistance. Proceedings of the IEEE International Symposium on Circuits and Systems, May 20-23, IEEE Xplore Press, Seoul, South Korea, pp: 1456-1459. DOI: 10.1109/ISCAS.2012.6271521

Deng, R. and F. Bao, 1988. A signcryption scheme with signature directly verifiable by public key. Proceedings of the 1st International Workshop on Public Key Cryptography, Feb. 5-6, Springer, Pacifico Yokohama, Japan, pp: 55-59. DOI: 10.1007/BFb0054014

ElKabbany, G., H. Aslan and M. Rasslan, 2014. A design of a fast parallel-pipelined implementation of AES: Advanced Encryption Standard. Int. J. Comput. Sci. Inform. Technol., 6: 39-35. DOI: 10.5121/ijcsit.2014.6603

Fan, J., K. Sakiyama and I. Verbauwhede, 2008. Elliptic curve cryptography on embedded multicore systems. Design Automat. Embedded Syst., 12: 231-242. DOI: 10.1007/s10617-008-9021-3

FOSIT, 2000. Elliptic curve cryptography version 2.0. Federal Office for Security in Information Technology, Technical Guideline TR-03111, Bonn, Germany.

Fernandes, A., F. Soares, V. Gomes, M. Freire and R. Inácio, 2014. Security issues in cloud environments: A survey. Int. J. Informat. Security, 13: 113-170. DOI: 10.1007/s10207-013-0208-7

GroBschadl, J., 2000. High-speed RSA hardware based on barret's modular reduction method. Proceedings of the 2nd International Workshop on Cryptographic Hardware and Embedded Systems, Aug. 17-18, Springer, London, pp: 91-203. DOI: 10.1007/3-540-44499-8_14

Guillermin, N., 2010. A high speed coprocessor for elliptic curve scalar multiplications over Fp. Proceedings of the 12th International Conference on Cryptographic Hardware and Embedded Systems, (HES' 10), Berlin, Heidelberg, pp: 48-64.

Gutub, A., M. Ibrahim and T. Al-Somani, 2007. Parallelizing GF(P) elliptic curve cryptography computations for security and speed. Proceedings of the 9th International Symposium on Signal Processing and its Applications, Feb. 12-15, IEEE Xplore Press, Sharjah, UAE, pp: 1-4. DOI: 10.1109/ISSPA.2007.4555449

Han, Y., X. Yang and Y. Hu, 2004. Signcryption based on elliptic curve and its multi-party schemes. Proceedings of the 3rd ACM International Conference on Information Security, Nov. 14-16, ACM, Shanghai, China, pp: 216-217. DOI: 10.1145/1046290.1046336

Hou, S., T. Uehara, S. Yiu, L. Hui and K. Chow, 2011. Privacy preserving confidential forensic investigation for shared or remote servers. Proceedings of the 7th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Oct. 14-16, IEEE Xplore Press, Dalian, China, pp: 378-383. DOI: 10.1109/IIHMSP.2011.28

Hou, S., R. Sasaki, T. Uehara and S. Yiu, 2013a. Verifying data authenticity and integrity in server-aided confidential forensic investigation. Proceedings of the International Conference on Information and Communication Technology, Mar. 25-29, Springer, Yogyakarta, Indonesia, pp: 312-317. DOI: 10.1007/978-3-642-36818-9_33

Hou, S., Y. Siu-Ming, U. Tetsutaro and S. Ryoichi, 2013b. A privacy-preserving approach for collecting evidence in forensic investigation. Int. J. Cyber-Security Digital Forens., 2: 70-78.

Hraiz, S., 2017. Challenges of digital forensic investigation in cloud computing. Proceedings of the 8th International Conference on Information Technology, May 17-18, IEEE Xplore Press, Amman, Jordan, pp: 568-571. DOI: 10.1109/ICITECH.2017.8080060

Hwang, M.S., C.C. Lee, J.Z. Lee and C.C. Yang, 2005a. A secure protocol for bluetooth piconets using elliptic curve cryptography. Telecommun. Syst., 29: 165-183. DOI: 10.1007/s11235-005-1689-0

Hwang, R., C.H. Lai and FF. Su, 2005b. An efficient signcryption scheme with forward secrecy based on elliptic curve. Applied Math Comput., 167: 870-881. DOI: 10.1016/j.amc.2004.06.124

Jung, H., K. Chang, D. Lee and J. Lim, 2001. Signcryption schemes with forward secrecy. Proceeding of the Information Security Application, Sept. 13-14, Seoul, Korea.

Koblitz, N., 1987. Elliptic curve cryptosystems. Mathmat. Comput., 48: 203-209. DOI: 10.1090/S0025-5718-1987-0866109-5

Li, C.T., C.C. Lee and C.W. Lee, 2013. An improved two-factor user authentication protocol for wireless sensor networks using elliptic curve cryptography. Sensor Lett., 11: 958-965. DOI: 10.1166/sl.2013.2669

Lillis, D., B. Becker, T. O'Sullivan and M. Scanlon, 2016. Current challenges and future research areas for digital forensic investigation. Proceedings of the 11th ADFSL Conference on Digital Forensics, Security and Law, (FSL' 16), Daytona Beach, Florida, USA.

Lo, J.W., C.C. Lee, M.S. Hwang and Y.P. Chu, 2010. A secure and efficient ECC-based AKA protocol for wireless mobile communications. Int. J. Innovative Comput. Inform. Control, 6: 1-9.

Malik, M., 2010. Efficient implementation of elliptic curve cryptography using low-power digital signal processor. Proceedings of the 12th International Conference on Advanced Communications Technology, Feb.7-10, IEEE Xplore Press, Phoenix Park, South Korea, pp: 1464-1468.

Meurice de Dormale, G. and J. Quisquater, 2007. High-speed hardware implementations of elliptic curve cryptography: A survey. J. Syst. Architecture, 53: 72-84. DOI: 10.1016/j.sysarc.2006.09.002

Miller, V., 1985. Use of elliptic curves in cryptography. Proceedings of the Advances in Cryptology, Aug. 18-22, Springer-Verlag, London, pp: 417-426.

Mohapatra, R.K., 2010. Signcryption schemes with forward secrecy based on elliptic curve cryptography. MSc Thesis, Department of Computer Science and Engineering, National Institute of Technology, India.

Nasreldin, M., H. Aslan, M. El-Hennawy and A. El-Hennawy, 2015a. New secure communication design for digital forensics in cloud computing. Int. J. Comput. Sci. Inform. Security, 13: 8-17.

Nasreldin, M., M. El-Hennawy, H. Aslan and A. El-Hennawy, 2015b. Digital forensics evidence acquisition and chain of custody in cloud computing. Int. J. Comput. Sci. Issues, 12: 153-160.

Nasreldin, M., H. Aslan, M. Rasslan and G. Weir, 2017. Evidence acquisition in cloud forensics. Proceedings of the IEEE 4th International Conference on New Paradigms in Electronics and Information Technology, Nov. 5-8, Alexandria, Egypt.

Parmar, N. and V. Verma, 2017. A comparative evaluation of algorithms in the implementation of an ultra-secure router-to-router key exchange system. Security Commun. Netw., 2017: 1467614. DOI: 10.1155/2017/1467614

Rao, A., K. Sujatha, A. Deepthi and L. Rajesh, 2017. Survey paper comparing ECC with RSA, AES and Blowfish Algorithms. Int. J. Recent Innovat. Trends Comput. Commun., 5: 44-47.

Sakthivel, A. and R. Nedunchezhian, 2014. Analyzing the point multiplication operation of elliptic curve cryptosystem over prime field for parallel processing. Int. Arab J. Inform. Technol., 11: 322-328.

Samy, G., B. Shanmugam, N. Maarop, P. Magalingam and S. Perumal *et al.*, 2017. Digital forensic challenges in the cloud computing environment. Proceedings of the International Conference of Reliable Information and Communication Technology, Apr. 23-24, Johor-Bahru, Malysia, pp: 669-679. DOI: 10.1007/978-3-319-59427-9_69

Singh, A., 2016. A modified signcryption scheme using elliptic curve cryptography. Proceedings of the National Conference on Recent Innovations in Science, Technology and Management, Feb. 26-27, Gurgaon Institute of Technology and Management, Gurgaon, pp: 12-16.

Srivastava, A. and A. Mathur, 2013. The Rabin cryptosystem and analysis in measure of Chinese reminder theorem. Int. J. Scientific Res. Public., 3: 1-4.

Tawalbeh, L., A. Mohammad and A. Gutub, 2010. Efficient FPGA implementation of a programmable architecture for GF(p) elliptic curve crypto computations. J. Signal Process. Syst., 59: 233-244. DOI: 10.1007/s11265-009-0376-x

Taylor, M., J. Haggerty, D. Gresty and D. Lamb, 2011. Forensic investigation of cloud computing systems. Netw. Security, 2011: 4-10. DOI: 10.1016/S1353-4858(11)70024-1

Thomas, C., G. Sheela and S. Krishnan, 2014. A survey on various algorithms used for elliptic curve cryptography. Int. J. Comput. Sci. Inform. Technol., 5: 7296-7301.

Toorani, M. and A.A. Beheshti, 2009. An elliptic curve-based signcryption scheme with forward secrecy. J. Applied Sci., 9: 1025-1035. DOI: 10.3923/jas.2009.1025.1035

Wall, D., 2007. Cybercrime: The Transformation of Crime in the Information Age. 1st Edn., Willy, Polity Press, Cambridge, ISBN-10: 0745627358, pp: 276.

Yuan, J. and C. Hung, 2008. Elixir: High-throughput cost-effective dual-field processors and the design framework for elliptic curve cryptography. IEEE Trans. Very Large Scale Integrat. Syst., 16: 1567-1580. DOI: 10.1109/TVLSI.2008.2001239

Zaghloul, S., L. AlShehri, M. AlJouie, N. AlEissa and N. Al-Mogheerah, 2017. Analytical and experimental performance evaluation of parallel merge sort on multicore system. Int. J. Eng. Comput. Sci., 6: 21764-21773. DOI: 10.18535/ijecs/v6i6.36

Zawoad, S. and R. Hasan, 2013. Digital forensics in the cloud. J. Defense Software Eng., 26: 17-20.

Zawoad, S., R. Hasan and J. Grimes, 2015. LINCS: Towards building a trustworthy litigation hold enabled cloud storage system. Digital Invest., 14: S55-S67. DOI: 10.1016/j.diin.2015.05.014

Zawoad, S., A. Dutta and R. Hasan, 2016. Towards building forensics enabled cloud through secure logging-as-a-service. IEEE Trans. Dependable Secure Comput., 13: 148-162. DOI: 10.1109/TDSC.2015.2482484

Zheng, Y., 1997. Digital signcryption or how to achieve cost(signature and encryption) << cost(signature) + cost(encryption). Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, Aug. 17-21, Springer, London, pp: 165-179. DOI: 10.1007/BFb0052234

Zheng, Y. and H. Imai, 1998. How to construct efficient signcryption schemes on elliptic curves. Inform. Process. Lett., 68: 227-233. DOI: 10.1016/S0020-0190(98)00167-7