International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 5, October 2019, pp. 4479~4492
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i5.pp4479-4492

Elliptic curve and pseudo-inverse matrix based cryptosystem for wireless sensor networks

Shomen Deb, Md. Mokammel Haque
Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Bangladesh

Article Info

Article history:

Received Feb 14, 2019
Revised May 23, 2019
Accepted Jun 11, 2019

Keywords:

Discrete Logarithm Problem (DLP)
Elliptic Curve Cryptography (ECC)
Pseudo-inverse
Public key Cryptography (PKC)
Wireless Sensor Network (WSN)

ABSTRACT

Applying asymmetric key security to wireless sensor network (WSN) has been challenging task for the researcher of this field. One common trade-off is that asymmetric key architecture does provide good enough security than symmetric key but on the other hand, sensor network has some resource limitations to implement asymmetric key approach. Elliptic curve cryptography (ECC) has significant advantages than other asymmetric key system like RSA, D-H etc. The most important feature of ECC is that it has much less bit requirement and at the same time, ensures better security compared to others. Hence, ECC can be a better option for implementing asymmetric key approach for sensor network. We propose a new cryptosystem which is based on Pseudo-inverse matrix and Elliptic Curve Cryptography. We establish a relationship between these two different concepts and evaluate our proposed system on the basis of the results of similar works as well as our own simulation done in TinyOS environment.

Corresponding Author:

Shomen Deb,
Department of Computer Science and Engineering,
Chittagong University of Engineering and Technology,
Kaptai Highway, Raozan, Chittagong 4340, Bangladesh.
Email: shomendeb@gmail.com

1. INTRODUCTION

There was a believe that due to resource limitations, PKC is not feasible in WSN to ensure security. Some recent works on public key cryptography have shown reasonable performance on wireless sensor network. With the current generation sensor, asymmetric approach is feasible in terms of both software and hardware perspective. PKC like ECC is very much achievable on 8 bit energy constrained platforms (MICA2, MICA2DOT motes using Atmel Atmega128L). In [1], it is stated that D. Nikam and V. Raut utilized ECC and Enhanced Adaptive Acknowledgment (EAACK) to improve the security of MANETs. V. L. Shivraj et al. reviewed the suitability of onetime passwords for IoT devices then developed a scheme using identity based ECC and Lamport's OTP algorithm. A. Dua et al. have proposed a scheme for secure smart city vehicle message communication. G. Sahebi et al. have designed a framework utilizing ECC for its fast speed, smaller keys, and greater security for E-health applications such as sensors and wearables. R. Fujdiak et al [2] has done analysis on 60 curves of different international standards and they emphasize on the importance of parameterization of ECC for performance issues. They showed that even 10-50 % execution time reduction can be possible on the prime field. We have seen the implementation of the Diffie-Hellman algorithm over EC on low power devices which is used in power grid and smart grid networks [3]. Their method can be used for key distribution over public channel.

As per our future predictions, ECC will play the leading role especially in wireless sensor network genre. By dint of the contribution of many researchers, we can now say that asymmetric key approach can be implemented in sensor network.There are many ideas presented earlier to implement asymmetric key in sensor network. Among those, a pseudo-inverse matrix based key handshaking scheme in asymmetric

manner for WSN, is proposed in [4]. The system mostly depends on TTP (Trusted Third Party) which is used to do major calculations. Their scheme is asserted to be more secure than Diffie-Hellman key exchange protocol and all the calculations are based on linear calculation. One year later in [5], Abedelaziz Mohaisen et al. raised an issue of security in that scheme since base station can be impersonated by any malicious entity and existence of TTP in WSN will be at high cost. In some cases, deployment of TTP seems to be challenging especially in hostile and adversarial environment.

Our contribution in this paper are as follows: first, we propose a new asymmetric cryptosystem which is mainly based on ECC and pseudo-inverse matrix algorithm. In terms of initial setup, participating nodes have to have common EC settings as they need to choose random point on EC and derive line equation to get pseudo-inverse matrix. This leads us to set up a unique relation between ECC and pseudo-inverse matrix. Second, we eliminate any base node requirement in our design. Third, rather than working on plain text like [6], we emphasize on secure code used to encrypt the message.

The rest of the paper is organized as follows: Related Works – Preliminaries - Proposed System – Implementation of the proposed cryptosystem - Performance and Evaluation – Security Analysis – Conclusion.

2. RELATED WORKS

Gupta et al [7] showed that ECC is not only feasible for sensor node but also enables the creation of a complete, secure web server stack that runs efficiently with very stringent resource constraints. Piotrowski et al [8] investigated four types of nodes; MICA2DOT, MICA2, MICAz, and TelosB, and estimated the power consumption for most common RSA and ECC operations. Roman and Alcaraz [9] discussed the applicability of public key infrastructures to wireless sensor networks and Ugus et. al. [10] implement elliptic curve and finite field arithmetic operations on a MICAz mote, which is a typical device employed in wireless sensor networks.Moreover, ECC has significant advantages over another popular PKC called RSA (Rivest-Shamir-Adleman). In [11], ECC-160 provides equivalent security to RSA-1024. Although, to emphasize on data security more, it is recommended to use ECC-224 which is equivalent to RSA-2048. RSA key generation requires generation of large prime numbers whereas ECC generates only random number as private key to build user public key.

3. PRELIMINARIES

Before start giving details on our proposed system, we would like to give a brief description about the two concepts (Pseudo-inverse matrix and ECC) which are ultimately the basement of our concept.

3.1. Elliptic curve cryptography

ECC (Elliptic Curve Cryptography) has become promising public key cryptography which offers asymmetric approach along with smaller key size, bandwidth savings and faster in implementations while compared to the RSA (Rivest-Shamir-Adleman) cryptography. Integer factorization is the basement of the security of RSA. For simplified elliptic curve E defined over a finite prime field F_p $(p > 3)$ is given below:

$$y^2 = x^3 + ax + b \bmod p \text{ where } a, b \in F_p \text{ and Discriminant, } \Delta = -(4a^3 + 27b^2) \neq 0$$

The discriminant must not be zero for an elliptic curve polynomial $x^3 + ax + b$ to possess three distinct roots. If discriminant is zero that would imply that two or more roots have coalesced, giving the curves in singular form. It is not safe to use singular curves for cryptography as they are easy to crack

3.2. Pseudo-inverse matrix

For a given matrix $A \in \mathbb{R}^{n \times m}$ and a matrix $A^g \in \mathbb{R}^{m \times n}$, A^g is a generalized inverse of A if it satisfies the conditions are: $AA^gA = A$, $A^gAA^g = A^g$, $(A^gA)^* = A^gA$, $(AA^g)^* = AA^g$. It is also known as Moore-Penrose inverse [12]. If we have more rows than columns $(m > n)$, then we have more equations than unknowns and the system is called over-determined linear system. Pseudo-inverse can be calculated as $A^g = (A^TA)^{-1}.A^T$. On the other hand, if columns are more than rows, then we have more unknowns than equations and the system is called under-determined linear system. Pseudo-inverse of this system can be calculated as $A^g = A^T.(AA^T)^{-1}$.

4. PROPOSED METHOD

Our system emphasizes on secret code which is used to encrypt the message. We have applied pseudo-inverse algorithm, matrix mapping technique and EC encryption and decryption on randomly generated secret code rather than applying on whole message. First of all, we would like to notice that there is no head node or base station involved in this cryptosystem. Sender and receiver nodes are given a specific elliptic curve equation $y^2 = x^3 + ax + b \bmod p$ where the order is prime and they will have a common generator point. We assume that they will have EC generated points based on that generator point. There will be a fixed nonsingular matrix and its inverse matrix. We also assume, all these pre-calculated parameters are accumulated in a security certificate. The certificate comes with a set of characters randomly mapped with those EC generated points for a certain period of time. Participating nodes in a specific network should be authenticated with this certificate. Figure 1 shows the detail concept of our proposed cryptosystem.

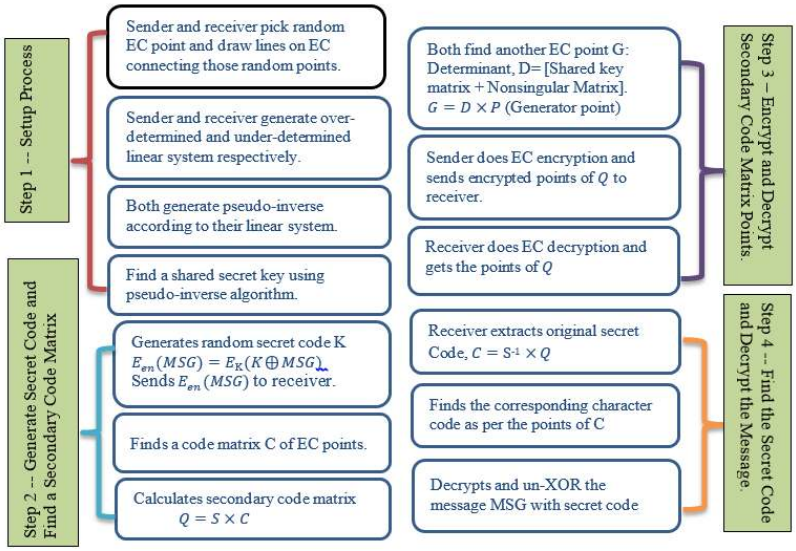


Figure 1. Block diagram of the proposed system

4.1. Setup process

Any sensor node wants to communicate with other node, has to initialize the connection using pseudo inverse algorithm. But in this case, choosing pseudo inverse matrix has to go through the below operation. Sender will choose over-determined linear system $Y = AX$ where $A \in \mathbb{R}^{n \times m}$ with $m > n$ (rows greater than columns). On the other hand, receiver will choose under-determined linear system $Y = AX$ where $A \in \mathbb{R}^{n \times m}$ with $n < m$ (rows less than columns). Considering the dimension of A matrix, both of them will pick random points from the EC generated points. In this case, number of random points should be double of the number of rows. They will draw lines on EC by connecting those random points and have their A matrix (which is ultimately coefficients of linear equation) finally. In both cases, A is not square matrix. Both sender and receiver will find their pseudo-inverse by $(A^{\mathrm{T}}A)^{-1}.A^{\mathrm{T}}$ and $A^{\mathrm{T}}.(AA^{\mathrm{T}})^{-1}$ respectively.

There is a bit different concept deployed to find out co-ordinate of A matrix in the receiving end. To meet nxm dimension (number of columns greater than number of rows), affine coordinate point will be converted to projective coordinate point or homogeneous coordinate point [13]. Suppose, we will follow (x, y, z) instead of (x, y). If we divide x and y coordinate value by z in such a way that coordinate points remains unchanged. In this way, receiver will build under-determined system which is one of the basic requirements of having pseudo-inverse matrix. Finally, both of them will get a shared secret key XY in setup process. See the Figure 2 and Algorithm 1.

4.2. Generate secret code and find a secondary code matrix

Sender will choose a random secret code K which length is n and each character of the secret code must exist in the character set of the certificate. It will first XOR and then encrypt the message $(E_K(K \oplus MSG))$ with this secret code. Then, it will send the encrypted MSG to receiver. Rather than sending the chosen code directly to receiver, sender will disguise the original code by doing some calculations. See the Algorithm 2.

Both parties have a set of generate EC points along with the character set and generator point on EC available for them. Mapped EC points for each character of secret code are $[C_1(X_1, Y_1), C_2(X_2, Y_2), C_2(X_3, Y_3), \ldots\ldots\ C_n(X_n, Y_n)]$.

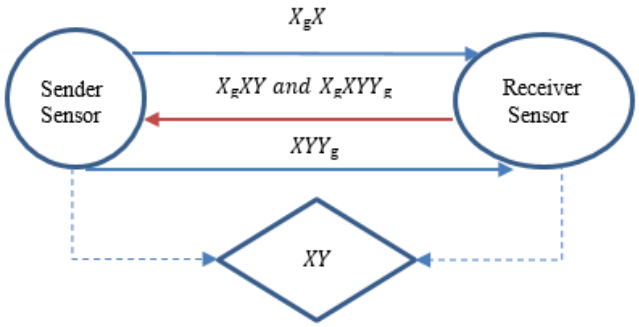


Figure 2. Key handshaking technique based on Pseudo-inverse matrix

If C be the random code matrix of length n, then generate a $3 \times r$ matrix where $r = n/3$ and $s = 2n/3$.

$$\boldsymbol{C} = \begin{pmatrix} C_1 & C_2 & C_3 & C_r \\ C_{r+1} & C_{r+2} & C_{r+3} & C_s \\ C_{s+1} & C_{s+2} & C_{s+3} & C_n \end{pmatrix}$$

Then, sender will do scalar multiplication (point doubling and adding) between code matrix and nonsingular 3×3 matrix which is only integer matrix. Secondary code matrix is:

$$\boldsymbol{Q} = S \times C = \begin{pmatrix} Q_1 & Q_2 & Q_3 & Q_r \\ Q_{r+1} & Q_{r+2} & Q_{r+3} & Q_s \\ Q_{s+1} & Q_{s+2} & Q_{s+3} & Q_n \end{pmatrix}$$

It will get another set of points T on EC; That is, $\boldsymbol{T} = [Q_1(X_1, Y_1), Q_2(X_2, Y_2), P_3(X_3, Y_3), \ldots\ldots\ Q_n\ (X_n, Y_n)]$.

Algorithm 1: Setup Process

Input: EC parameters [a,b,prime], generator point P, generated EC points $p_1, p_2 \ldots\ldots p_n$
Output: a shared key matrix XY
1: **for** i ← 1 to n do
2: Assign $p_1, p_2 \ldots\ldots p_n$ to Coorlist
3: **end for**
4: **for** i ← 1 to n do
5: Apply randomization on Coorlist → output is a randomized list of EC points for senderpointlist and receivepointlist
6: **end for**
7: Call line_fn(senderpointlist, receiverpointlist) → outputs are Sender Matrix X_A and Receiver Matrix Y_A
8: **if** $(X_A \neq null)$ **then**
9: call overdeterminedlinear(X_A) → output is a sender pseudo-inverse matrix X_{gA}
10: **end if**
11: **if** $(Y_A \neq null)$ **then**
12: call underdeterminedlinear(Y_A) → output is a receiver pseudo-inverse matrix Y_{gA}
13: **end if**
14: Call pseudoinverse_algorithm (X_A, Y_A, X_{gA},Y_{gA}) → output is a Shared key matrix XY

Algorithm 2: Generating secret code C and find a secondary code matrix

Input: Generate a random secret code K, choose a message MSG, Nonsingular matrix S
Output: Q be a secondary code matrix
1: E(MSG) = E_K(K ⊕ MSG) and send it to receiver
2: **for** i ← 1 to n **do**
3: Find the corresponding EC point on Coorlist for each character of K and build a code matrix C
4: **end for**
5: **for** i ← 1 to n **do**
6: Do EC scalar Multiplication between C and S → output is a secondary code matrix Q
7: **end for**

4.3. Encrypt and decrypt secondary code matrix points

Let E be an Elliptic Curve and P be a generator point on the elliptic curve. G is another Elliptic Curve point which is calculated as $\boldsymbol{G} = D \times P$ where D be a determinant of [Shared key matrix + Nonsingular Matrix].

Secondary code encryption:

Sender's secret key = m Receiver's secret key= n
Sender's public key = mG Receiver's public key = nG and send it to sender.
Encrypted each point represented as $[E_1, E_2 - E_n]$ where $E_1 = mG$ and $E_2 = Q_1 + m(nG)$ and calculate other $E_3 - E_n$ in the same way and send those to receiver at a time.

Secondary code decryption:

Decrypted point, $D_p = E_2 - n.E_1 = Q_1 + m.(nG) - n.(mG) = Q_1 + m.(nG) - m.(nG) = Q_1$
Receiver will decrypt other cipher text $E_3 - E_n$. Algorithm 3 is implemented for this step.

4.4. Find the secret code and decrypt the message

After getting decrypted point, receiver needs to extract the secret code by which message was encrypted. As we mentioned earlier, both of them have 3×3 non-singular matrix S and its inverse S^{-1}. So, it will do EC scalar multiplication to find the actual secret code, $\boldsymbol{C} = S^{-1} \times Q$

As all the EC points of C are mapped with a character (as per the certificate), so receiver can easily decrypt and un-XOR the message MSG with secret code. Step 4 is described in the Algorithm 4.

Algorithm 3: Encrypt and decrypt secondary code matrix points.

Input: a shared key matrix XY, nonsingular matrix S, Generator point P, sendersecret, receiversecret, secondary code matrix Q
Output: E_p be the encrypted points, D_p be the decrypted points
1: **for** i ← 1 to n **do**
2: Do addition between XY and S → output is a another matrix Z
3: **end for**
4: Call determinant(Z) → output is a integer value D
5: **if** ($D \neq 0$) **then**
6: Do EC point multiplication between D and P → output is a EC point G
7: **end if**
8: **if** (G ≠ null) **then**
9: Do EC point multiplication between G and sendersecret, G and receiversecret → outputs are senderpublickey, receiverpublickey
10: **end if**
11: **for** r ← 1 to Q.Count **do** // sender side encryption
12: Call ECPoint(Q[r].Xcoordinate,Q[r].Ycoordinate) → output is a EC point E1
13: Do EC point multiplication between sendersecret and receiverpublickey → output is a EC point E2
14: Do EC addition between E1 and E2 → output is a encrypted point Ep
15: **end for**
16: **for** t ← 1 to Ep.Count **do** // receiver side decryption
17: Call ECPoint(E_p[t].Xcoordinate,E_p[t].Ycoordinate) → output is a EC point E1
18: Do EC point multiplication between receiversecret and senderpublickey → output is a EC point E2
19: Do EC addition between E1 and ECPoint(E2.Xcoordinate,-E2.Ycoordinate) → output is a decrypted point D_p
20: **end for**

Algorithm 4: Find the secret code and decrypt the message.

Input: decrypted points D_p , Inverse of nonsingular matrix S^{-1}
Output: Secret code matrix C, Secret code K, message MSG
1: **for** i ← 1 to n **do**
2: Call ECPoint(D_p[i].xcoordinate, D_p[i].ycoordinate) → output is a EC point E1
3: Do EC scalar Multiplication between E1 and S^{-1} → output is a secret code matrix C
4: **end for**
5: **for** i ← 1 to n **do**
6: find the corresponding character as per the points of C. // secret code K
7: **end for**
8: Call decrypt_XOR(K, MSG) → output is a message MSG

5. IMPLEMENTATION OF THE PROPOSED CRYPTOSYSTEM

We start with the sample Elliptic curve $y2 = x3 + x + 13$ mod 59 and it has 67 points including ∞. Generator point is (1, 29).

We have 3×3 nonsingular matrix $S = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix}$ and Inverse of such matrix $S^{-1} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$

In the initital step, both sender and receiver will choose random EC points in such a way that over determined and under determined linear system can be made respectively. On the basis of those linear system, they will find out pseudo-inverse matrix and ultimately shared secret key matrix using pseudo-inverse matrix algorithm. The algorithm is given in [4]. A per Figure 3,

Sender Matrix $\boldsymbol{X_A}$: $\begin{pmatrix} -15 & 26 \\ -17 & 23 \\ -13 & 16 \end{pmatrix}$ Receiver Matrix $\boldsymbol{Y_A}$: $\begin{pmatrix} -10 & 26 & 1 \\ -14 & 26 & 1 \end{pmatrix}$

Note: For receiving end, affine (X, Y) coordinate is transformed into projective coordinate (X, Y, Z) point in which all the points are divided by z (which is 1) so that original coordinate point remains same. So, sender and receiver both have X_A and Y_A respectively. They can find their pseudo-inverse X_{gA} and Y_{gA} matrix respectively and do the rest of the calculations according to the pseudo-inverse matrix algorithm to establish a shared secret key XY.

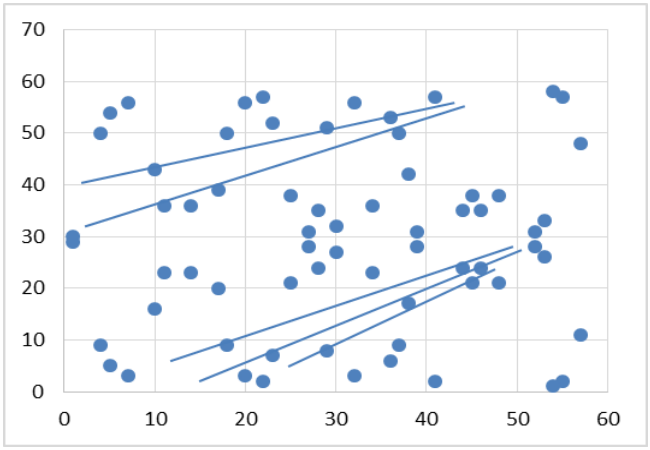


Figure 3. Draw lines on EC $y^2 = x^3 + x + 13$ mod 59

In the second step, sender generates a random secret code "**Aub#en!cr**". The message is first xored and then encrypted with that random secret code and sent to the receiver. Each character of the secret code must exist in the certificate. EC generated points are randomly mapped with the character set. Suppose, for our secret code, mapped EC points for each character are: A = 38P = (4,9), u= 26P = (48,21), b = 35P = (5,5), # = 29P = (4,50), e = 39P = (46, 35), n = 48P = (38,42), ! = 44P = (52,28), c = 16P = (23,7), r = 2P = (14,23).

So, we will get a code matrix C which is based on EC points. $C = \begin{pmatrix} (4,9) & (48,21) & (5,5) \\ (4,50) & (46,35) & (38,42) \\ (52,28) & (23,7) & (14,23) \end{pmatrix}$

So, Secondary code matrix, $\boldsymbol{Q} = S \times C = \begin{pmatrix} (52,28) & (20,3) & (39,28) \\ (36,53) & (14,23) & (1,29) \\ (17,39) & (39,28) & (10,16) \end{pmatrix}$

Third step starts with shared key matrix, $XY = \begin{pmatrix} -214 & 286 & 11 \\ -152 & 156 & 6 \\ -94 & 78 & 3 \end{pmatrix}$ and a matrix Z which is a summation of

shared key matrix XY(Pseudo-inverse matrix key handshaking technique) and nonsingular matrix S.

$$\boldsymbol{Z} = \begin{pmatrix} -214 & 286 & 11 \\ -152 & 156 & 6 \\ -94 & 78 & 3 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix} = \begin{pmatrix} -213 & 287 & 12 \\ -151 & 158 & 8 \\ -93 & 80 & 6 \end{pmatrix}$$

Let D be the determinant of matrix Z. So, $\boldsymbol{D} = 12258$. If generator point P is $(1, 29)$, then we can calculate another point $\boldsymbol{G} = D \times P = 12258 \times (1, 29) = (10, 43)$.
Sender's secret key = $m = 11$, Sender's public key = $mG = 11 \times (10, 43) = (22, 57)$
Receiver's secret key = $n = 7$, Receiver's public key = $nG = 7 \times (10, 43) = (55, 57)$ and send it to sender.
Encrypted each point represented as $[E_1, E_2 - E_{10}]$ where $E_1 = mG = (22,57)$
$E_2 = Q_1 + m(nG) = (52,28) + 11(7(10,43)) = (52,28) + (44,35) = (20,3)$.
Sender will calculate other encrypted points from $E_3 - E_{10}$ similarly and send those to receiver at a time.
Decrypted points are calculated at receiver side.
$D_p = E_2 - n.E_1 = (20,3) + (44, -35) = (52,28) = Q_1$. Receiver will find out other points $Q_2 - Q_9$ by decrypting cipher text $E_3 - E_{10}$.

In this final step, receiver needs to do only EC scalar multiplication between secondary code matrix Q and inverse of the nonsingular matrix S.

$$\boldsymbol{C} = S^{-1} \times Q = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \times \begin{pmatrix} (52,28) & (20,3) & (39,28) \\ (36,53) & (14,23) & (1,29) \\ (17,39) & (39,28) & (10,16) \end{pmatrix} = \begin{pmatrix} (4,9) & (48,21) & (5,5) \\ (4,50) & (46,35) & (38,42) \\ (52,28) & (23,7) & (14,23) \end{pmatrix} = \begin{pmatrix} A & u & b \\ \# & e & n \\ ! & c & r \end{pmatrix}$$

At last, receiver can decrypt and un-XOR the message with the secret code C.

6. PERFORMANCE AND EVALUATION

We would like to do performance analysis and evaluation in terms of energy cost of computation and communication, speed and memory consumption. We take reference from other papers to do our analysis and evaluate our system. Our first reference is [4] which entails pseudo-inverse matrix to develop a key handshaking scheme between participating nodes. To make it truly asymmetric, we have involved elliptic curve cryptography which is popular for WSN. We develop a pseudo-inverse matrix by choosing random points from the generated EC points. Putting those points into line equation and deriving pseudo-inverse

matrix are very simple algebraic calculations and it will take negligible amount of time and cost. As per their analysis, number of bits transmitted during key handshaking process is $n(2n + k + m)$. These calculations are also linear.

6.1. Energy analysis

We assume that a certificate will consist of the elliptic curve and its generated points along with mapped character set, a generator point or primitive point or base point, nonsingular matrix and its inverse matrix. We believe that it will save a lot of energy. An experiment [14] conducted on Mica2Dot (Atmel ATmega128L 8-bit microcontroller) showed that energy cost of computation is much less than data transmission. Unit cost of transmitting and receiving one packet consisting of 41 byte are 2.9 mJ and 1.4 mJ respectively. Figure 4 gives us a comparison between various PKC based handshake schemes. In terms of transmitting and receiving during key handshake scheme, simplified SSL consumes much energy than others due to the amount of exchanged data. Total energy cost of our sender and receiver node are respectively 7.2 mJ and 5.7 mJ that are less than or closer to Micro PKI [15].

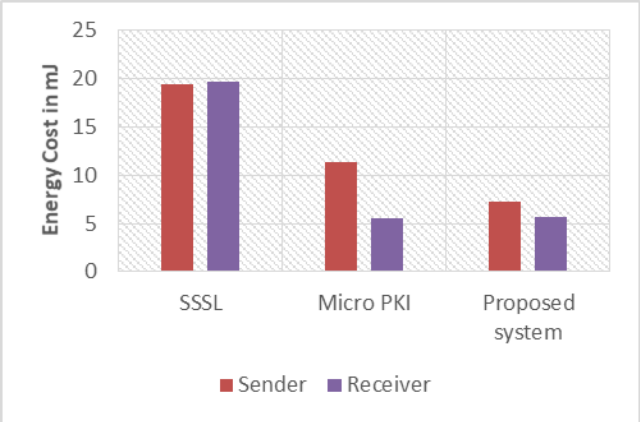


Figure 4. Comparing energy consumption for PKC schemes

In Micro PKI, the scheme can consume a lot of energy if the base node is far from the participating sensors since packets are sent over multihop link. Unlike [4], absence of base node also reduces the cost of energy consumption as both the pairing nodes initiate their communication by themselves. Rest of the transmission are happened in other steps where sender and receiver need 1 transmit and 1 receive (cost 4.3 mJ) respectively in the step 2. In step 3, 1 transmit and 1 receive are needed by both the sensor nodes (cost 8.6 mJ). Hence, if we consider step by step process, enegy cost of our system is much reasonable for the sensor node.

Giacomo et al [16] demonstrates the energy cost of two sensors named MICAz and TelosB in terms of computation, transmit, receive, listen and sleep. Using the same settings, we would like to compare three differnet key agreement protocols in the MICAz and TelosB sensor respectively. In this case, we have considered only transmit and receive. Comparison is given in the Table 1.

Table 1. MicaZ and TelosB-Energy consumption during transmission and reception

Protocols	Process	MicaZ (mJ)	TelosB (mJ)
Kerberos	Send	0.9	1.1
Kerberos	Receive	1.1	1.3
Proposed key handshake	Send	0.71	0.85
Proposed key handshake	Receive	0.79	0.95
ECDH-ECDSA	Send	1.3	1.6
ECDH-ECDSA	Receive	1.5	1.8

It is clearly seen that energy cost of our EC-Pseudoinverse based key handshake scheme is less than others. Kerberos [17] uses Trusted Third Party which leads the protocol to increase the number of calculations, transmisions and receptions. In our case, no TTP or base node is used. On the other hand,

ECDH-ECDSA does not involve TTP, instead Elliptic Curve Digital Signature Algorithm is used for authentication purpose. Overall cost of that protocol is higher than others due to its longer calculation. Although, we are using EC concept into our system, but in the initial connection setup or key hand shaking process, two things are improving our performance. First, most of the parameters are precalculated. Second, expensive point multiplications are not done in this step.

If we had sensor, we could have estimated the cost of computation by measuring the timing performance (mJ/T_{clk}) of the implementation on the sensor node. We believe that Step 1 (Key handshaking) does not consume much energy since randomly choosing EC points and consequently, deriving line equations and pseudoinverse matrix are all linear operations. As per [14], energy cost of RSA-1024 and ECC-160 based handshake are approximately 390 mJ and 93 mJ. Considering our given example, we want to calculate computation cost [16] in terms of ECC-160 scalar multiplications. Experimental data is given in the Table 2. Encryption on MICAz in Step 2 consumes much energy than that of TelosB. For TelosB, energy consumption in other steps is much less than RSA-1024 handshake. Although it depends on the nonsingular matrix and the code matrix. Qty refers to the number of point multiplication.

Table 2. Energy consumption while doing point multiplication for MicaZ and TelosB

	Sender						Receiver					
Process	MICAz			TelosB			MICAz			TelosB		
	Qty.	Cost (mJ)	Total Cost (mJ)	Qty.	Cost (mJ)	Total Cost (mJ)	Qty.	Cost (mJ)	Total Cost (mJ)	Qty.	Cost (mJ)	Total Cost (mJ)
Step 1:	0	55	0	0	17	0	0	55	0	0	17	0
Step 2:	12	55	660	12	17	204	0	55	0	0	17	0
Step 3:	3	55	165	3	17	51	0	55	0	0	17	0
Step 4:	0	55	0	0	17	0	7	55	385	7	17	119
	Total Cost:		825			255			385			119

Compared to the pseudo-inverse based handshake, energy cost of transmitting and receiving for RSA-1024 based SSL handshake is much higher because of its longer key sizes. In our case, for integer data type and 3×3 matrix, 108 bytes of payload is transmitted by both sender and receiver whereas with SSL handshake based on ECC-160, both parties need to transmit almost double of the payload of pseudo-inverse matrix based handshake. Figure 5 illustrates the fact.

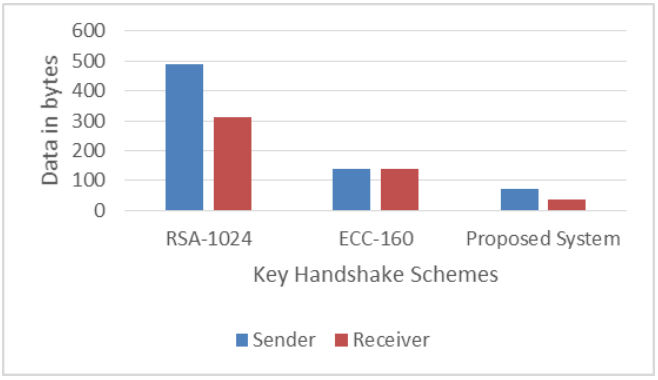


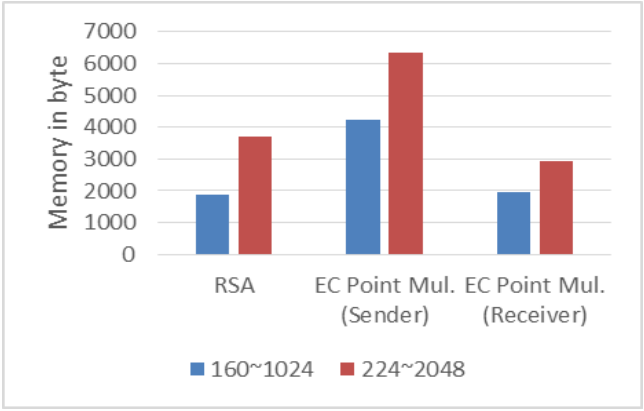
Figure 5. Amount of data transmitted during handshake

6.2. Speed and Memory Consumption

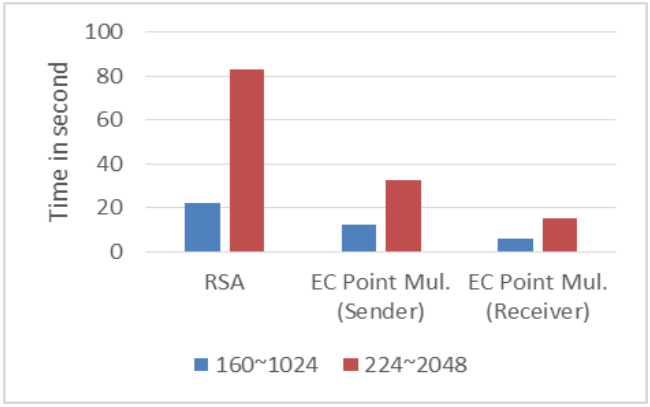
Due to longer key size of RSA or other public key scheme, ECC has much lower execution time and memory requirements. Nils Gura et al [18] shows that ECC-160 takes only 0.81s with 282 bytes of data memory whereas RSA-1024 private key modular exponentiation takes nearly 11s with 930 bytes of data memory. Moreover, ECC-224 (2.19s, 422 bytes data) is equivalent to RSA-2048 (83.26s, 1853 bytes data). For our given example, sender needs 12 point multiplications in step 2, 3 point multiplications in step 3 and receiver needs 7 point multiplications in step 4. In this case, we consider EC curve secp160r1 and secp224r1 for point multiplication along with the equivalent RSA-1024 and RSA-2048 private key modular exponentiation.

In Figure 6, both the comparisons have been done between the number of EC multiplications through the steps and two RSA private key modular exponentiation. Even we do number of EC operations, time cost is still less than RSA operation. If we would have done such amount of RSA operations as the number of EC multiplications in our steps, memory cost would be far higher than memory consumption of EC based system.It clearly shows us that ECC has great advantages over RSA in terms of time and memory.

To evaluate the speed of our proposed cryptosystem, we have developed a program which is built using python 2.7. We have used different configuration CPU and The Prime Pages Library. In this case, we have changed only prime value to check our result and taken a random secret code of length 9 as per our given example. Table 3 shows various speed values in second for some sample prime values.



(a)



(b)

Figure 6. RSA private key modular exponentiation versus EC point multiplication,
(a) comparing memory requirement, (b) comparing time requirement

Table 3. Proposed system running on different CPU

Prime Number	OS: Ubuntu 14.04 Quad Core, 4 GB			OS: Windows 10 Core 2 Duo, 1 GB			OS: Windows 7 Quad Core, 256 MB		
	R.T.	P.G.T.	En/De	R.T.	P.G.T.	En/De	R.T.	P.G.T.	En/De
P=10691	0.02	1.5	0.007/0.0007	0.17	2.78	0.015/0.00138	0.09	1.88	0.010/0.001
P=56333	0.15	38.29	0.009/0.001	0.66	73.32	0.02/0.0032	0.58	40.9	0.016/0.0017
P=130199	0.32	200	0.015/0.002	0.97	400.99	0.03/0.005	0.93	294.55	0.02/0.003
P=150193	0.62	945	0.02/0.0033	1.1	1202.56	0.05/0.009	0.98	1188.42	0.02/0.004
P=599983	1.35	1735	0.045/0.013	2.15	2515.43	0.08/0.024	1.87	2812.36	0.05/0.017

R.T. = Running Time, P.G.T.= Point Generation Time, En/De= Encryption Time / Decryption Time

We calculate the estimated time cost of each step separately. We have found that for even large prime value, encryption/decryption time cost remains very negligible and total running time is very much reasonable. Sample data shows that most of the time is consumed for calculating point generation and is increased as per the prime value size. Since our assumption is to have certificate consisting of generated points on the curve, So we can easily ignore this time required for point generation. Total running time is distributed among calculating pseudo-inverse matrix based algorithm, random code generation (In our example, code length is 9) along with scalar multiplication with nonsingular matrix, secondary code encrytion/decryption and extracting original code using inverse of nonsingular matrix. Hence, the longer the code is, the longer execution time and memory are required. There is a trade-off between code size and sensor's resource limitations.

To calculate time to transmit data, we develop a nesc program in TinyOS 2.1.2 and debugged the program with python interface. For this simulation, we create a data file containing the nodes connected and the gain value. We observe that it takes 0.282 s – 0.46 s to transmit one packet which consists of matrix value or public key or encrypted EC points. Simulation data is given in the Table 4.

Table 4. Simulating packet transmission in TinyOS environment

Process	Transmission					
	Sender			Receiver		
	Qty.	Unit Time in Sec.	Total Time in Sec.	Qty.	Unit Time in Sec	Total Time in Sec.
Step 1:	2	0.46	0.92	1	0.46	0.46
Step 2:	1	0.46	0.46	0	0.46	0
Step 3:	1	0.46	0.46	1	0.46	0.46
Step 4:	0	0.46	0	0	0.46	0
Total:			1.84			0.92

6.3. Optimization

Our first observation for improvement is that since our implementation is focused on the curve over prime field, domain parameters of specific curve plays a significant role to increase computation speed. As per R. Fujdiak et al. [19], we can have fastest curve and slowest curve within the same group. They showed that having changes in parameters will have effect on speed upto 50%. As an example, the curve wtls9-160 bits has CPU cycles 277533 which is almost 50% increase of the CPU cycles (184404) of wtls7-160 bits. So, choosing right curve with right domain parameters results in increased speed as well as reduced in memory requirements.

Second observation is that ECC operations (both addition and doubling) need a field inversion and multiple multiplications. It is known that inversion is considerably expensive than multiplication. Hence, it is advantageous to adopt projective coordinates rather than affine coordinates to avoid inversion operations. We can consider various types of projective coordinates such as standard projective coordinates, Jacobian projective coordinates, chudnovsky coordinates and mixed Jacobian-affine coordinates. Among those projective coordinates, Jacobian coordinates has fastest point doubling and Jacobian-affine coordinates has fastest point adding as per the book written by D. Hankerson et al. [11]. If the point is fixed and storage is available, then pre-computation on some data $2P, 2^2P \ldots\ldots 2^{t-1}P$ will eliminate all doublings and results in accelerated point multiplication consequently.

Third observation for the betterment of this cryptosystem is that there are numerous method to calculate scalar multiplication which is k times addition of a point P. So, it can be written as

$$Q = kP = P + P + P + \cdots + P$$

Where k is a positive integer and P, Q are elliptic curve points. Target is to reduce the number of these additions that will make scalar multiplication faster and more efficient. In this case, basic method used in this purpose is called binary method. This method scan every bits of k and its performance depends on the type representation of k. There are some other methods such as Non-Adjacent Form (NAF) proposed by Booth [20], generalization of NAF method called w-NAF, Direct Recording by Pathak and Sanghi [21] in 2010, Mutual Opposite Form (MOF) by Okeya et al [22] etc. More than 77% of the total execution time on the ATmega128 are spent on multiplications and squarings for point multiplication over secp160r1. Nils Gura et al [18] proposed a hybrid multiplication strategy which improves the performance for ECC point multiplication upto 24.8% for sec160r1 and 25% for secp224r1 on the ATmega128 8-bit microcontroller.

As per assumption, we have considered nonsingular matrix, inverse of its matrix, generated EC points along with character set to be included into the security certificate. All the sensor nodes will have a security certificate to validate or authenticate themselves before joining in the transmission. Most of the

parameters are calculated and fixed in advance. We have done little bit change in encryption process. Rather than sending encrypted text one by one, we have sent all the encrypted text (E_2— E_n) at a time. That reduces the significant amount of computation and transmission cost. Number of operations such as EC addition, doubling or multiplication depend on nonsingular matrix and its inverse matrix. Presence of nonsingular matrix enables us to disguise the original secret code which is used to encrypt the message. Unlike other research works, most of the computations & transmission are done on the secret code rather than whole plain text. Higher code length will ensure higher security. We can optimize cost of EC calculation by considering various projective co-ordinate system and methods [23] to avoid expensive inversion operation or to reduce the number of multiplication.

7. SECURITY ANALYSIS

In this section, we will analyse each step of our proposed system in terms of security issues. In the first step, there doesn't exist any base node, then there is no question of being compromised with others. For every transmission, both will choose their EC points randomly and will form their linear system to build pseudo-inverse matrix. They can find a new shared key in every transaction or after a certain period of time. So, first step is very much secure.

In the second step, only sender initiates or generates a secret code. Original message or plain text is xored and encrypted with this code. This code is totally disguised by multiplying nonsingular matrix and code matrix. Since, secret code only belongs to sender, so this step is also secure.

In the third step, we have chosen an EC function which order is prime, then any point except zero point will be a generator. All points form a cyclic group. Moreover, we will have another point G ($G = D \times P$) to get involved into secondary code encryption. As we said, shared secret key was generated from random linear system by choosing random points on EC. On the other hand, nonsingular matrix is a common parameter for both the parties involved into transmission. If we do analysis encryption and decryption process, we will find following observation:

a. If an adversary finds the public key of sender and receiver, he cannot be able find out secret key. Because it is elliptic curve discrete logarithm problem over a finite field.
b. If a hacker finds encrypted points E_1, E_2 where $E_1 = mG \; and \; E_2 = Q + m(nG)$. Solving this two cipher text will require to solve DLP.
c. At the decryption side, if attacker wants to calculate $D = E_2 - n.E_1$, then he needs to calculate $E_1 \; and \; E_2$ first which is stated earlier that it's a DLP.

As stated in [24], finding $\#E(Fq)$ exactly is computationally difficult when a large prime factor is involved. All EC points rely on the hardness of the ECDLP. If the EC is chosen carefully, the best known algorithm (Exponential time algorithm) for computing the ECDLP requires $\approx \sqrt{P}$ steps. Ex: $P \approx 2^{160}$, attacker requires $\sqrt{P} \approx \sqrt{2^{160}} = 2^{160/2} = 2^{80}$ steps.So, if we work with ECC-160, this step becomes safe and secure. Even ECC-224 or ECC-256 will make this step strong enough.

The final step requires scalar multiplication between inverse of nonsingular matrix and decrypted points matrix Q to find out original code which was used to encrypt original message/plain text. Ultimately, it is also DLP.

8. CONCLUSION

In our proposed system, we have tried to limit almost all the calculations to be performed on secure code which enables a sender to reduce running time, energy consumption, expensive calculations etc. Even for a large message, system will only concentrate on secure code and that makes the system simple and protected. In one hand, there is a unique concept introduced in which a relation between pseudo-inverse matrix and elliptic curve has been established using random linear system and consequently, we get random shared key in every transactions. On the other hand, we understand that there are some expensive EC computations involved into the proposed system. Since some crucial parameters are calculated in advance before initiating any transactions, so we are hopeful that our proposed cryptosystem will be realistic for wireless sensor network.

REFERENCES

[1] R. Harkanson, Y. Kim, "Applications of Elliptic Curve Cryptography," *the 12th Annual Cyber and Information Security Research Conference*, Oak Ridge, Tennessee, USA, April 04-06, 2017.

[2] R. Fujdiak, P. Dzurenda, P. Mlynek, et al., "Anomalous Behaviour of Cryptographic Curves over Finite Field," *Financed by the National Sustainability Program under grant LO1401*, vol. 23, no. 5, 2017.

[3] R. Fujdiak, P. Dzurenda, P. Mlynek, et al., "Cryptograph Key Distribution with Elliptic Curve Diffie-Hellman Algorithm in Low-Power Devices for Power Grids," *Rev. Roum. Sci. Techn.– Électrotechn. et Énerg. Bucharest*, vol. 61, no. 1, pp. 84-88, 2016.

[4] M.M. Haque, A.S.K. Pathan, C.S. Hong, *et al.*, "An Asymmetric Key-based Security Architecture for Wireless Sensor Networks," *KSII Transactions on Internet and Information System*, vol. 2, no. 5, pp. 265-279, 2008.

[5] A. Mohaisen, J.W. Choi, D. Hong, "On the Insecurity of Asymmetric Key-based Architecture in Wireless Sensor Networks," *KSII Transactions on Internet and Information System,* vol. 3, no. 4, pp. 376-384, 2009.

[6] X. Fang, Y. Wu, "Investigation into Elliptic Curve Cryptography," *Proceedings of 3rd International Conference on Information Management (ICIM)*, Chengdu, China, 2017.

[7] V. Gupta, V. Wurm, M. Zhu, et al., "Sizzle: A Standards-based End-to-End Security Architecture for the Embedded Internet," *SMLITR-2005-145*, Kauai Island, HI, USA, vol. 1, no. 4, pp. 425-445, June 2005.

[8] K. Piotrowski, P. Langendoerfer, and S. Peter, "How Public Key Cryptography Influences Wireless Sensor Node Lifetime," *Proceedings of ACM SASN 2006*, Virginia, USA, pp. 169-176, 2006.

[9] R. Roman, C. Alcaraz, "Applicability of Public Key Infrastructures in Wireless Sensor Networks," EuroPKI 2007, *Springer-Verlag*, LNCS 4582, pp. 313-320, 2007.

[10] O. Ugus, A. Hessler, and D. Westhoff, "Performance of Additive Homomorphic EC-ElGamal Encryption for TinyPEDS," Technical Report 6, July 2007.
Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.7766&rep=rep1&type=pdf

[11] D. Handerson, A. Menezes, S. Vanstone, *"Guide to Elliptic Curve Cryptography," Springer-Verlag New York,* 2004.

[12] P. Courrieu, "Fast Computation of Moore-Penrose Inverse Matrices", *Neural Information Processing – Letters and Reviews*, Université de Provence, France, vol. 8, no. 2, August 2005.

[13] V. Skala, "Length, Area and Volume Computation in Homogeneous Coordinates," *International Journal of Image and Graphics,* Czeck Republic, Vol. 6, no. 4, pp. 625-639, 2006.

[14] A.S. Wander, N. Gura, H. Eberle, et al., "Energy Analysis of Public-key Cryptography for Wireless Sensor Networks," *Proceedings of the 3rd IEEE Int'l Conf. on Pervasive Computing and Communication PerCom2005*, pp. 324-328, 2005.

[15] E. Munivel and G M Ajit, "Efficient Public key Infrastructure Implementation in Wireless Sensor Network," *International Conference on Wireless Communication and Sensor Computing*, pp. 1-6, 2010.

[16] G. de Meulenaer, F. Gosset, F.X. Standaert, et al., "On the Energy Cost of Communication and Cryptography in Wireless Sensor Network," *IEEE International Conference on Wireless and Mobile Computing*, Networking and Communications, Avignon, France, pp. 580-585, 2008.

[17] J.Großsch¨adl, A. Szekely, and S. Tillich, "The Energy Cost of Cryptographic Key Establishment in Wireless Sensor Networks," *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS 2007)*, pp. 380–382. ACM Press, 2007.

[18] N. Gura, A. Patel, A. Wander, et al., "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," *Published in Cryptographic Hardware and Embedded Systems – CHES 2004: 6th International Workshop Cambridge*, MA, USA, 3156, pp. 119-132, August 11-13, 2004.

[19] R. Fujdiak, P. Masek, J. Hosek, et al., "Efficiency Evaluation of Different Types of Cryptography Curves on Low-Power Devices," *Proceedings of 2015 7th International Congress on Ultra-Modern Telecommunications and Control Systems and Workshops (ICUMT)*, Brno, Czech Republic, 2015.

[20] A.D. Booth, "A signed binary multiplication technique," *The Quarterly Journal of Mechanics and Applied Mathematics,* vol. 4, no. 2, pp. 236-240, 1951.

[21] H.K. Pathak and M. Sanghi, "Speeding up computation of scalar multiplication in elliptic curve cryptosystem," *International Journal on Computer Science and Engineering*, vol. 2, no. 4, pp. 1024–1028, 2010.

[22] K. Okeya, K. Schmidt-Samoa, C. Spahn, et al., "Signed binary representations revisited," *In Advances in Cryptology-CYRPTO2004*, Springer-Verlag, LNCS 3152, pp. 123-139, 2004.

[23] N.F.H. Al Saffar, M. Rushdan, "High Performance Methods of Elliptic Curve Scalar Multiplication," *International journal of computer applications*, vol. 108, no. 20, pp. 39-45. December 2014.

[24] F. Minfeng, C.Wei, "Elliptic Curve Cryptosystem ElGamal Encryption and Transmission Scheme," *International Conference on Computer Application and System Modeling (ICCASM 2010)*, Taiwan, China, 2010.

BIOGRAPHIES OF AUTHORS

Shomen Deb: He is a student of masters program in the Department of Computer Science and Engineering (CSE) at Chittagong Universit of engineering and Technology (CUET, Bangladesh). He graduated in CSE from Stamford University Bangladesh.

Md. Mokammel Haque: He is currently working as an Associate Professor in the Department of Computer Science and Engineering at Chittagong University of Engineering and Technology (CUET, Bangladesh). He received his Doctorate degree from Macquarie University, Australia. He got this MS degree in Computer Engineering from Kyung Hee University, South Korea. He completed B.Sc. Eng. in CSE from CUET, Bangladesh. His research interests include cyber security, computer networks, cryptography, algorithms etc.