CHAPTER 5

HIGH SPEED AND LESS COMPLEX ECC OVER BINARY FIELDS FOR WIRELESS SENSOR NETWORKS

5.1 INTRODUCTION

Wireless Sensor Networks (WSNs) [47] consists of large number of sensor nodes and few powerful control nodes. The sensor nodes sense the environmental changes and report them to the control nodes over flexible network architecture. Sensor nodes are generally composed of one or more sensing components, a processing component and a communication component. Control nodes process the data collected from the sensor nodes, disseminate control commands to sensor nodes and connect the network to a traditional wired network. Three major security issues concerning sensor networks are Localization, Time Synchronization and Routing [95-96].

Secure Localization in WSNs is crucial for applications such as target detection and tracking, precision navigation, search and rescue, geographic routing and security surveillance. Some of the existing Localization techniques are Received Signal Strength Indicator (RSSI), Time of Arrival (ToA), Time Difference of Arrival (TDoA) and Angle of Arrival (AoA) [48-51]. These existing techniques are vulnerable to both the internal and external attackers. While the internal attackers report false distance information in order to cheat on their positions, the external attackers modify the positions of the other sensor nodes. Thus the need for the development of a Secure Localization technique arises.

In WSNs, Secure Time Synchronization assumes significance by way of providing a common clock among the sensor nodes. A group of sensor nodes needs a common synchronized clock to synchronize their behaviours on switching between
wake-up and sleep modes at the same time [55]. Assignment of time slots to sensor nodes facilitates multiple accesses in the shared communication medium of Medium Access Control protocols. By overcoming attacks, sensor nodes provide a synchronized clock to access their time slots without colliding with other nodes. Sensor nodes usually contain inexpensive oscillators with clock drifts which are intolerable for WSNs applications. Therefore, time synchronization becomes indispensable for many WSNs applications.

Finally, WSNs lack a fixed infrastructure where the mobile nodes are moving freely causing the network topology to change dynamically. However, the mobile nodes are vulnerable to physical capture and compromise. Hence, the routing algorithms need to manage node mobility in a secured manner apart from providing data reliability.

5.2 SECURE LOCALIZATION IN SENSOR NETWORKS

In sensor networks, nodes are deployed into an unplanned infrastructure in a dynamic manner without prior knowledge of their location. Anchor nodes are nodes whose positions are already determined by GPS or manual configuration. Determining the position of all the nodes from a few anchor nodes, relative distance and angle information among the nodes is termed as position estimation or Localization. In order to use the data collected by sensors, it is often necessary to have their position information stamped. For example, to detect and track objects with sensor networks, the physical position of each sensor should be known in advance for identifying the position of the detected objects.

Once the spatial co-ordinate of the sensors has been determined, the greatest challenge imposed by the sensor nodes is in exchanging these spatial co-ordinates in a secured manner with their authenticated neighbours. Attackers can easily inject bogus messages into the network to interrupt normal network functionalities. Therefore, it is appropriate to employ public-key cryptographic techniques for Secure Localization.
5.2.1 Proposed Secure Localization Technique

This module of the research work proposes a public-key cryptography technique for Secure Localization and authentication among sensor nodes. This proposed technique integrates high speed and less complex ECC algorithm developed in chapter four for ToA Localization scheme. The key exchange among the nodes is done by using ECC key exchange. This proposed new technique is compared for its timing performance with RSA and Mean Power with Rivest-Shamir-Adelman (MPRSA) for Secure Localization. The simulation results indicate that ECC is well suited for Secure Localization in sensor networks as it satisfies the constraints of the sensor networks which include minimum bandwidth, power, energy and computational speed.

5.2.2 Implementation of ECC for ToA Localization

Localization schemes can be divided into two categories viz., range based and range free. The former are characterized by using absolute point-to-point distance (range) or angle estimates in location derivations, while the latter depend on messages from neighbouring sensors and/or anchors. Range based solutions provide more accurate locations by employing higher hardware resource. However, range free approaches guarantee coarse grained location accuracy by employing lesser hardware resource. It is observed that almost all existing range based proposals were designed for benign scenarios where nodes co-operate to determine their locations. As a result, they are not suited for unattended and often hostile settings such as tactical military operations and homeland security monitoring. Under such circumstances, attackers can easily subvert the normal functionalities of WSNs by exploiting the weakness of Localization algorithms. To overcome these limitations, ECC-ToA Localization technique has been developed.

As shown in Figure 5.1, ECC-ToA technique is a range based Localization technique. The mutual authentication and key distribution among the sensor nodes was implemented using the developed high speed and less complex ECC algorithm.
The sensor node A is unaware of its position. The position of node A was determined by using the nodes B₁, B₂ and B₃ whose positions were already known. Each position aware node Bᵢ sends an ECC encrypted message to A and the trip time of the signal was computed. The trip time multiplied by the propagation speed of signals (i.e. the speed of light) yields a distance \( d_i \).

As shown in Figure 5.1, the distance \( d(A,Bᵢ) \) defines a circle around node \( d(A,Bᵢ) \). The position of A is on the circumference of the generated circle. In a two dimensional model, the position of A is unambiguously determined as the intersection of three such circles. Trip time measurement from each node Bᵢ to node A requires synchronized and accurate clocks at both locations.

Figure 5.1 ToA Technique
5.2.3 Simulation of ECC based ToA Localization Technique

The proposed ECC based ToA Localization technique was simulated with MATLAB 7.

![Diagram of Node Deployment](image)

**Figure 5.2 Deployment of Nodes**

The random deployment of 25 sensor nodes in an area of size $10 \times 10$ is shown in Figure 5.2. The position of these nodes was initially assumed. By using ECC based ToA technique the positions of the nodes were estimated and the performance characteristics of the proposed technique were analyzed.

5.2.3.1 Computation of ToA

To compute the time of arrival of the signal, encrypted data is sent from the anchor node to sensor node and the time of arrival is calculated by setting a timer.
Figure 5.3 Data Transmission from Anchor Node

The timer is set ON when the last bit of the assumed binary data 00110110 is sent from the anchor node and the timer is set OFF when the last bit of the data is received from the sensor node.
Figure 5.4 Data Reception by Anchor Node
The transmission of data by a Quadrature Phase Shift Keying (QPSK) transmitter after modulating with a pseudo random bit stream is shown in Figure 5.3. A serial to parallel conversion of the pseudo random bit stream is performed with mapping of two bits per symbol. The QPSK based system is used because it gives better bit error rate performance. A cosine and sine carrier was configured and the '1' and 'Q' symbols modulate these carriers via mixers. The binary data was divided into the odd bits and even bits. The even bits are the Q Channel Data and the odd bits are the I Channel Data where the bits are one half of the original serial bit rate.

Figure 5.4 illustrates the reception of data by the anchor node. After receiving, the data was decrypted and the ToA of the signal was calculated. At the receiver side a parallel to serial conversion of data was carried out. The binary data received is low pass filtered which consists of I Channel output and Q Channel output. For better Bit Error Rate (BER) performance, bandpass filters can be added at both the transmitter and receiver side. After computing the distance, location verification was done for the validation of the distance.

5.2.4 Results and Discussion

Comparison of the encryption and decryption time for ECC based ToA Localization technique using RSA, MPRSA and ECC was done. The simulations were carried out for both DH and ECC key Exchanges. The performance characteristics of RSA, MPRSA and ECC encryption and decryption timing for different key sizes are shown in Figures 5.5 and 5.6. Significant improvement in the time taken for encryption and decryption was achieved. For a key size of 128 bits the encryption time of ECC was 0.09 milliseconds. Thus a saving of 0.26 and 0.46 milliseconds was achieved when compared to MPRSA and RSA respectively.
Figure 5.5 RSA, MPRSA and ECC Encryption Time for Different Key Sizes

Figure 5.6 RSA, MPRSA and ECC Decryption Time for Different Key Sizes
Figure 5.7  Performance Comparison of RSA, MPRSA and ECC Encryption Time using DH Key Exchange.

Figure 5.8  Performance Comparison of RSA, MPRSA and ECC Decryption Time using DH Key Exchange.
RSA encryption  MPRSA encryption  ECC encryption

Figure 5.9  Performance of RSA, MPRSA and ECC Encryption Time using ECC Key Exchange

RSA decryption  MPRSA decryption  ECC decryption

Figure 5.10  Performance of RSA, MPRSA and ECC Decryption Time using ECC Key Exchange
Figure 5.11 Computed Positions of the Nodes

Figure 5.12 Time Measure Error versus Location Error Rate
Figures 5.7 and 5.8 illustrate the results of simulations performed using DH key exchange. The simulations were repeated for ECC key exchange and the results are shown in Figures 5.9 and 5.10. The simulation was performed for 25 sensor nodes and can extend to any number of nodes.

The computed positions of the nodes based on ECC based ToA technique is illustrated in Figure 5.11. The computed position of the nodes was then compared with the initial randomly deployed positions. The time measure error with location error rate for the nodes is illustrated in Figure 5.12. This error in the estimation of the position of the nodes occurs due to the error in the time of arrival of signal.

5.3 SECURE TIME SYNCHRONIZATION

The applications envisioned for sensor networks require collaborative execution of a distributed task amongst a large set of sensor nodes. This collaborative execution is realized by exchanging messages that are time stamped using the local clocks on the nodes. In general sensor networks are dense in large number. Also, energy efficiency is a major concern in these networks due to the limited battery capacity. This eliminates the use of external energy consuming equipments such as GPS receivers. To operate in such a large network densities with energy constraints, scalable and secured time synchronization algorithms are required.

5.3.1 Proposed Level Based Time Synchronization with ECC

Level Based Time Synchronization with ECC was proposed to provide redundant ways for each node to synchronize its clock with the common source. It can overcome partially missing or false synchronization information provided by compromised nodes. This technique constructs a level hierarchy initially and synchronizes the nodes, level by level. Eventually all nodes in the network synchronize their clocks to a reference node.

Level based time synchronization uses high speed and less complex ECC algorithm to provide efficiency in terms of scalability, energy and computational
cost. Even if certain number of malicious nodes disrupts clock synchronization, each normal node can still synchronize its local clock to the source node. The technique is also extended to cluster based sensor networks which further improves scalability and the time taken for synchronization. Hence, the proposed technique establishes a unique global timescale for all the nodes in a sensor network by way of providing secure clock synchronization using ECC.

5.3.2 Implementation of Level Based Time Synchronization with ECC

The proposed technique was implemented in two phases as follows:

Level Discovery Phase

This phase occurs when the network is deployed. Each node is assigned a level in this hierarchical structure. Only one node which has lowest identity is assigned to level 0 which is the source node. It is ensured that a node belonging to level \( i \) can communicate with at least one node belonging to level \( i-1 \).

![Hierarchical Sensor Network Diagram](image.png)

**Figure 5.13 Hierarchical Sensor Network**

The source node initiates this phase by broadcasting a level discovery message. The level discovery message contains level number and the sender's
identity authenticated with the pair-wise key shared between the sender and the receiver. The node after establishing its own level would then broadcast a new level discovery message. This process is continued and eventually every node in the network is assigned a unique level. On being assigned a level, a node neglects any such future messages.

In a sensor network, the nodes are usually deployed in a random fashion. Scenarios might exist where a sensor node may join the network after the completion of the level discovery phase. It is also possible to have another scenario where the nodes which are present at the onset of the network might not receive any level discovery messages owing to Medium Access Control layer collisions. In both the scenarios the nodes wait for some period of time for a level to be assigned. If the nodes are not assigned a level within that period, it broadcasts a level request message. The neighbours reply to this request by sending their own level. The new node assigns itself a level in a new phase termed local level discovery phase.

**Synchronization Phase**

Once the hierarchical structure has been established, the source node initiates the synchronization phase. A node belonging to a level $i$ has to synchronize to a node belonging to level $i-1$. Eventually every node is synchronized to the source node and global time synchronization is achieved. All the sensor nodes have unique identifier and bi-directional links that are used in the networks. The link level protocol ensures that each node is aware of the set of nodes (neighbour set) with which it can directly communicate.

As shown in Figure 5.14, pair-wise synchronization is achieved by the two way message exchange between nodes ‘A’ and ‘B’. $T_1$ and $T_4$ represent the time measured by local clock of node ‘A’. Similarly $T_2$ and $T_3$ represent the time measured by local clock of node ‘B’. At time $T_1$ node ‘A’ sends a synchronization pulse message to node ‘B’. Node B receives this message at $T_2$ and computes $T_2$ using equation (5.1):

$$T_2 = T_1 + D + d.$$  

(5.1)
Computation of clock drift and the propagation delay between the nodes are done using equation (5.2) and equation (5.3):

\[ D = \frac{(T2 - T1) - (T4 - T3)}{2} \]  

(5.2)

\[ d = \frac{(T2 - T1) + (T4 - T3)}{2} \]  

(5.3)

where,

D represents the clock drift between the two nodes and
d refers to the propagation delay between the two nodes.

\[ B \rightarrow T2 \rightarrow T3 \rightarrow T2, T3 \text{ are measured in } \]
\[ \text{Node B clock} \]

\[ A \leftarrow T1 \leftarrow T4 \rightarrow \]
\[ T1, T4 \text{ are measured in } \]
\[ \text{Node A clock} \]

**Figure 5.14 Pair-wise Synchronization**

At time T3, node 'B' sends back an acknowledgement message to node 'A'. The acknowledgement message consists of the level number of 'B' and the values of T1, T2 and T3. Node A receives the message at T4. Assuming that the clock drift and the propagation delay do not change in this small span of time, it is possible for node 'A' to compute the clock drift and propagation delay. This message exchange at the network level begins with the source node initiating the phase by broadcasting a time synchronization message. On reception of this message, nodes belonging to level 1 wait for some random time before initiating the two way message exchange with the source node. This randomization is meant to avoid contention in medium access. On receiving back an acknowledgment, these nodes adjust their clock to the source node.
The nodes belonging to level 2 would overhear this message exchange. This is based on the fact that every node in level 2 has at least one node of level 1 in its neighbour set. On hearing this message, nodes in level 2 back off for some random time, after which they initiate the message exchange with nodes in level 1. This randomization is meant to ensure that nodes in level 2 start the synchronization phase after nodes in level 1 have synchronized. After synchronization the node sends back an acknowledgement. This ensures that multiple levels of synchronization are not formed in the network. This process is carried out throughout the network and eventually every node is synchronized to the source node. In a sensor network, message collisions can take place quite often. To handle such situation, a node waiting for an acknowledgement timeouts after some random time and retransmits the synchronization pulse. This process is continued until a successful two way message exchange has been performed.

5.3.3 Cluster Based Time Synchronization

In level based time synchronization, the time required to synchronize a large sensor network to a single source node is high. This is due to message propagation and increased occurrences of message collisions. Due to clock drift, the nodes far away from the source node are not synchronized with a high precision. The synchronization error for time synchronization increases as the hop count or the level along the synchronization path increases. Clusters are deployed into the network. This cluster based time synchronization [97] improves scalability and reduces the time required for synchronization.

5.3.4 Results and Discussion

The efficacy of integrating ECC in Level Based Time Synchronization (LBTS) and Cluster Based Time Synchronization (CBTS) depends upon the selection of Elliptic Curve. The irreducible pentanomial used for 163 bit key was $x^{163} + x^7 + x^6 + x^3 + 1$. The field elements were isomorphed to ring representation by using the trinomial $x^{202} + x + 1$. 

106
5.3.4.1 Simulation Parameters

The simulation of the proposed technique was simulated by using the Parallel Simulation Environment for Complex (PARSEC) [98] System, an event driven parallel simulation language. PARSEC is a C-based discrete event simulation language adopting the process interaction approach to discrete event simulation. An object or set of objects in the physical system is represented by a logical process. Interactions among physical processes (events) are modeled by time stamped message exchanges among the corresponding logical processes. One of the important distinguishing features of PARSEC is its ability to execute a discrete event simulation model using several different asynchronous parallel simulation protocols on a variety of parallel architectures.

Table 5.1 Simulation Parameters

<table>
<thead>
<tr>
<th>PARAMETERS</th>
<th>MODEL/ VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Terrain</td>
<td>100m × 100m</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>50 – 200</td>
</tr>
<tr>
<td>Transmission range of a sensor node</td>
<td>20m</td>
</tr>
<tr>
<td>Packet size</td>
<td>40 bytes</td>
</tr>
<tr>
<td>MAC layer</td>
<td>CSMA</td>
</tr>
<tr>
<td>Elliptic Curve</td>
<td>Elliptic Curve over binary field, GF(2^m)</td>
</tr>
</tbody>
</table>

The Elliptic Curve \( y^2 + xy = x^3 + x^2 + 1 \) was chosen for simulation. Pairs of ECC public/private keys were generated and each pair was assigned an index. Figure 5.15 illustrates the deployment of the sensor nodes in a random fashion. The total number of nodes deployed was 200. The sensor node with the lowest identity number was chosen as the source node. The simulation parameters considered for simulation of the proposed secure time synchronization using ECC are presented in Table 5.1.
5.3.4.2 Synchronization Error for LBTS

The performance of LBTS with and without FCC is illustrated in Figure 5.16.
An increase in the number of nodes in the sensor network tends to increase the synchronization error. The sensor nodes encrypt the message with ECC during pair-wise time synchronization so as to tolerate the attacks due to malicious nodes. It is observed that LBTS with ECC shows better accuracy when compared to LBTS without ECC. As the number of nodes increase in the sensor network, there is an increase in the synchronization error.

5.3.4.3 Synchronization Time for LBTS

The time taken to synchronize all the nodes in the network for LBTS with and without ECC is illustrated in Figure 5.17.

![Figure 5.17 Synchronization Time for LBTS](image)

The synchronization time is the duration from the start of time synchronization to the time for which the last sensor node is synchronized. Each node carries out a message exchange independent of the other nodes in the network. Thus, it is anticipated that the time taken for synchronization increases linearly with the number of nodes. With the integration of ECC in the two way message...
exchange: there is minimum increase in synchronization time due to ECC encryption and decryption which is in terms of microseconds. As a result, the synchronization time increases to some extent when compared to the synchronization time without ECC.

5.3.4.4 Comparison of Synchronization Error for CBTS and LBTS

The performance comparison of synchronization error for LBTS and CBTS using ECC is illustrated in Figure 5.18. It is observed that CBTS outperforms the synchronization error in LBTS. The synchronization error of LBTS is about 14.7 μs and it increases as the number of levels in the hierarchy increases. However, the synchronization error for cluster based technique is around 10.5 μs.

![Figure 5.18 Comparison of Synchronization Error for CBTS and LBTS](image)

5.3.4.5 Comparison of Synchronization Time for CBTS and LBTS

In comparison to CBTS, the synchronization time for LBTS is high since synchronization starts from the source node and this process continues from one
level to another level, thereby synchronizing all the nodes in the network. In CBTS the nodes are clustered under cluster heads in the network. The cluster heads initiate the synchronization process at the same time leading to reduction in synchronization time. As shown in Figure 5.19, the synchronization time for CBTS is less when compared to LBTS.

![Figure 5.19 Comparison of Synchronization Time for CBTS and LBTS](image)

5.4 SECURE OLSR FOR DATA RELIABILITY

Optimized Link State Routing (OLSR) specified in IETF RFC 3626 [66] is a proactive, table driven routing protocol. The RFC specifies OLSR as a pure route maintenance protocol responsible for determining and maintaining routes. Each node via the route maintains topology information about the network by periodically exchanging link state messages. To update topological changes, control messages like HELLO messages in the form of packets have to be flooded through the entire network. Flooding means retransmission of every packet at every node. However, in large and dense networks, flooding leads to high number of unnecessary
retransmissions resulting in increased control message overhead and losses of control packets due to collisions of packets.

Also during the process of routing, WSNs are vulnerable to various attacks including Denial of Service (DoS) attack. Hence, the need for a secure routing protocol which enhances data reliability and overcomes DoS attacks is important. This module of the research work focuses in securing OLSR protocol in an optimized manner for a resource constraint environment like WSNs.

5.4.1 Proposed ECC-TC based OLSR-MPR Protocol

Optimization of OLSR protocol is obtained through a technique called Multipoint Relay (MPR). The key idea of MPR is to minimize the size of each control message and the number of nodes re-broadcasting a message thereby optimizing the bandwidth usage. The proposed protocol is best suitable for large and dense networks. Further data reliability is achieved by implementing ECC in OLSR protocol based on Threshold Cryptography (TC). TC involves the sharing of a key by multiple individuals engaged in ECC encryption or decryption or splitting of message either before or after ECC encryption. TC avoids trusting and engaging just one individual node for executing the operation. Hence, the objective is to share the key or message in such a way that each individual node performs computation on the message without revealing any secret information about its partial key or partial message. The developed high speed and less complex ECC algorithm over binary fields has been implemented for the cryptographic operation. From the simulated results, it was analyzed that this secure routing protocol overcomes DoS attacks and is well suited for sensor networks where both the sender and receiver have equal resource constraints.

5.4.2 OLSR-MPR Protocol

As shown in Figures 5.20(a) and 5.20(b), OLSR uses the MPR concept as an optimization to standard flooding. Instead of standard flooding, OLSR uses MPRs to reduce the size and the number of control messages throughout the network. Based upon its one hop and two hop neighbourhoods, a node selects one hop neighbours as
MPRs in such a way that each two hop neighbour can be reached by at least one member of the MPR set. The MPR set maintains the set of nodes which were elected MPRs by this node. The neighbour elected as MPR is marked MPR neighbour in the next HELLO message that is sent. The node being elected MPR upon receiving this HELLO message adds the node that selected it as MPR into its MPR selector set. The MPR selector set includes all neighbours that have selected this node as MPR. A node only forwards traffic of nodes that are included in its MPR selector set.

![Figure 5.20 (a) Standard Flooding](image)

![Figure 5.20(b) MPR Flooding](image)

Another advantage of using MPRs is the reduction in the number of nodes needed to broadcast the HELLO messages for updating the topology of the network. Only nodes elected as MPR announce those nodes that elected them as MPR (the members of its MPR selector set) to the entire network. This received topology information is sufficient to compute the routing table and the one and two hop neighbourhood in every node of the network.

The proposed algorithm for selecting MPR is executed in the following sequence:

i) From the source node, choose all the symmetric one hop neighbours which are willing to act as an MPR.

ii) Calculate for every neighbour node a degree which is the number of the symmetric neighbours that are two hops away from the
calculating source and does not include the source or its one hop neighbours.

iii) Add the symmetric neighbours to the two hop neighbour set.

iv) If there are more number of nodes for the two hop neighbour set, the selection of the MPR is done using the following rules:

- Choose the node with high willing value.
- Calculate the node with greater number of reachability.
- Select the node which has minimum delay.

5.4.3 Implementation of ECC-TC based OLSR-MPR Protocol

The proposed ECC-TC based OLSR-MPR protocol for WSNs operates in the following sequence:

i) As soon as a node joins the network, it begins its neighbour discovery by listening to HELLO messages. When all the neighbour nodes have been detected, the node defines its MPR list and in turn sends HELLO messages declaring its neighbours and its MPR list.

ii) Upon selection of a node as MPR, it sends topology control messages announcing MPR selector list. Before sending topology control messages, the node generates a pair of keys namely the private and public key using ECC and then floods the public key announcing its presence in the network.

iii) In the proposed scheme, topology control messages are actually sent through the selected trusted neighbours. These messages are signed with their respective attributed secret shares using Threshold Cryptography. To decide on the threshold value, Shamir secret sharing scheme [67] is used in threshold cryptography. For this purpose, 'k' one hop neighbours are selected according to their trust level. The source node sends to each of them a share of its secret key.
encrypted with the respective public key of the trusted neighbour. Upon completion of sending its share, the source node destroys the secret key so that its own key is not compromised.

iv) Finally, the source node sends its message encrypted with the respective neighbour’s public key to each of the ‘k’ one hop selected neighbours.

v) The ‘k’ selected neighbours flood the topology control message signed with their shares \((K, S_i)\). The sharing of the source secret key is completed by threshold cryptography mechanism.

vi) The destination node receives ‘k’ different messages signed with ‘k’ different shares of the secret key.

5.4.3.1 ECC-TC Algorithm

Figure 5.21 depicts the ECC-TC model where sender ‘S’ generates partial messages using Shamir’s Lagrange interpolation [67]. The steps in the ECC-TC model are executed in the following sequence:

![Figure 5.21 Block Model for ECC-TC](image-url)
1. Sender ‘S’ distributes the partial messages ‘C_i,s’ along with corresponding ‘x_i,s’ securely to all neighbouring nodes on distinct disjoint routes.

2. Available nodes on these routes perform the task of forwarding partial message packets till it reaches the Receiver ‘R’.

3. When ‘R’ receives ‘t’ or more ‘C_i,s’ and ‘x_i,s’, using first ‘tx_i’ values it calculates the corresponding cipher text ‘C’. For split after encryption, partial messages are first combined using Lagrange interpolation to recover original ‘C’. The cipher text ‘C’ is decrypted to recover the original message ‘M’.

**Share Split After Encryption (SAE)**

- Sender converts the secret ‘M’ to a point $P_M = (x, y)$ on Elliptic Curve and computes $P_C = n_A n_B G + P_M = (x_C, y_C)$.

- Sender uses Shamir’s secret sharing method to split ‘x_C’ and ‘y_C’ into ‘n’ shares of ‘x_{C1}’ and ‘y_{C1}’ respectively, where the threshold ‘t’ is $1 \leq t \leq n$ and then sends ‘n’ pieces of ‘x_{C1}’ and ‘y_{C1}’ to receiver.

- Receiver combines ‘t’ pieces of ‘x_{C1}’ and ‘y_{C1}’ separately to get $(x_C, y_C)$ and computes $P_M = P_C - n_A n_B G$. Finally the point $P_M$ is converted to secret ‘M’.

**Share Split Before Encryption (SBE)**

- Sender uses Shamir’s secret sharing method to split the secret ‘M’ into ‘n’ shares of secret ‘M_i’, where the threshold ‘t’ is $1 \leq t \leq n$ and converts each share ‘M_i’ to a point ‘P_i’ on the Elliptic Curve.

- Sender computes $P_1 + n_A n_B G$ and sends it to Receiver.

- Receiver recovers ‘P_1’ by subtracting $n_A n_B G$ from $P_1 + n_A n_B G$. With at least ‘t’ share of ‘P_M’, the receiver is able to recover ‘P_M’. Finally the point $P_M$ is converted to secret ‘M’.
5.4.3.2 ECC Menezes Vanstone-TC Algorithm

The proposed protocol was also implemented for ECC Menezes Vanstone (ECMV) which is simpler when compared to ECC-TC. ECMV solves the problem of encoding a message into a point on the Elliptic Curve. Let ‘H’ be a cyclic subgroup of $E_p(a, b)$ with the generator ‘G’. Receiver has a private key ‘$n_B$’, and a public key ‘$n_BG$’. The message ‘M’ is converted into a point $P_M = (x, y)$ in $GF(2^m)$. The ECMV-TC encryption and decryption algorithms are executed in the following sequence:

Encryption algorithm

- Sender select a random number $r < |H|$ and computes $rn_BG = (x_k, y_k)$.
- Sender sends $(rG, x_k \mod p, y_k \mod p)$ to the Receiver.

Decryption algorithm

- Receiver calculates $n_BRG = rm_BG = (x_k, y_k)$.
- Receiver recovers ‘x’ and ‘y’ by $(x_k^{-1}x_k \mod p)$ and $(y_k^{-1}y_k \mod p)$.
- Receiver converts the point $(x, y)$ to get the original plaintext ‘M’.

Share Split Before Encryption (SBE)

- Sender splits the message ‘M’ into n shares of secret ‘$M_s$’, where the threshold ‘t’ is $1 \leq t \leq n$ and converts each share ‘$M_s$’ into a point $P_t (x_t, y_t)$.
- Sender selects a random number $r < |H|$, and calculates $rn_BG = (x_k, y_k)$ and sends $(rG, x_kx_t \mod p, y_ky_t \mod p)$ to the Receiver.
- Receiver calculates $n_BRG = m_BG = (x_k, y_k)$ and recovers ‘$x_t$’ and ‘$y_t$’ by $(x_k^{-1}x_kx_t \mod p)$ and $(y_k^{-1}y_ky_t \mod p)$. With at least ‘t’ shares of ‘$P_M$’, the Receiver recovers ‘$P_M$’ and converts it to the secret ‘M’.
Share Split After Encryption (SAE)

- Sender converts the message 'M' into a point $P_M(x,y)$ and calculates $m_BG = (x_k, y_k)$, $z = x_k \pmod{p}$ and $w = y_k \pmod{p}$.
- Sender splits 'z', 'w' into 'n' shares of 'z_i' and 'w_i' respectively, where the threshold 't' $1 \leq t \leq n$ and sends $rG$ and 'n' pieces of 'z_i' and 'w_i' to the Receiver.
- Receiver combines 't' pieces of 'z_i' and 'w_i' separately to get (z,w) and calculates $m_BrG = m_BG = (x_k, y_k)$. It recovers $P_M$ by $x_k^{-1}z = x_k^{-1}x_k \pmod{p}$ and $y_k^{-1}w = y_k^{-1}y_k \pmod{p}$. Finally the point $P_M$ is converted to secret 'M'.

5.4.3.3 RSA-TC Algorithm

RSA-TC scheme was also simulated for the OLSR-MPR protocol and compared with the proposed ECC-TC scheme.

![Figure 5.22 Block Model for RSA-TC](#)

The steps in the RSA-TC model shown in Figure 5.22 are executed in the following sequence:
1. Sender 'S' distributes the shared keys along with 'x_i' values amongst its 'n' neighbours which acts as Share Holders (SH) with (f(x_i) mod φ, x_i, N) assigned to each SH.

2. 'S' sends message 'M' securely to all SHs for partial encryption.

3. SHs apply f(x_i) to 'M' and sends partial encrypted messages as C_i = M^{f(x_i)} mod N and 'x_i' to Receiver 'R'. A few SHs may not be available or a few messages from SH may be lost during the transmission.

4. 'S' notifies 'R' about threshold 't', 'N' and 'e'.

5. 'R' sends selected 'x_i' values to 'S' for 'x_i' values.

6. 'S' calculates 'x_i' values over mod φ (N) and sends them to 'R'. 'R' applies 'x_i' values to all 'C_i' and combines them to get the original 'C'. C^e mod N then gives the message 'M'.

5.4.4 Results and Discussion

The proposed protocol was simulated using NS-2 and MATLAB. The performance characteristics were analyzed.

5.4.4.1 Simulation Parameters

The simulation parameters considered for simulation of the proposed ECC-TC based OLSR-MPR protocol are tabulated in Table 5.2.

<table>
<thead>
<tr>
<th>PARAMETERS</th>
<th>MODEL/VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobility Model</td>
<td>Random Way Point Model</td>
</tr>
<tr>
<td>Pause Time</td>
<td>100 sec</td>
</tr>
<tr>
<td>Traffic Type</td>
<td>Constant Bit Rate</td>
</tr>
<tr>
<td>Simulation Time</td>
<td>100 sec</td>
</tr>
<tr>
<td>Speed</td>
<td>5-10-25-20 metres/second</td>
</tr>
<tr>
<td>MAC layer</td>
<td>CSMA</td>
</tr>
<tr>
<td>Packet Size</td>
<td>64 bytes</td>
</tr>
</tbody>
</table>
5.4.4.2 MPR Flooding Analysis

Figure 5.23 depicts the comparison results of ordinary flooding message with MPR flooding. It is observed that MPR minimizes the control traffic in terms of packet length and control message number thereby optimizing bandwidth usage.

![Graph showing comparison between flooding and MPR flooding]  
**Figure 5.23** Comparison Performance of Flooding with MPR Flooding

![Graph showing impact of mobility on the ATHR for a network of 50 nodes]  
**Figure 5.24** Impact of Mobility on the ATHR for Network of 50 Nodes
5.4.4.3 Average Throughput Received

The throughput of a connection between two nodes is measured as the number of bytes delivered per time unit. The impact of mobility on the Average Throughput Received (ATHR) for a network of 50 nodes was analyzed and is shown in Figure 5.24. The available bandwidth in the network and the protocol overhead influences the overall throughput in the network.

5.4.4.4 Performance Results for ECC-TC, ECMV-TC and RSA-TC Algorithms

The total encryption and decryption timings for ECC-TC based OLSR-TC protocol are shown in Figures 5.25 and 5.26.

![Total Encryption Timings for ECC-TC](image)

Figure 5.25 Total Encryption Timings for ECC-TC

For split after encryption, during the encryption process as ‘t’ and ‘n’ increases Lagrange timings contributes more than ‘n*K_t’ which is constant for any given key size irrespective of ‘t’ and ‘n’ values. For decryption, Lagrange timings are small but increases with ‘t’, ‘n’ and key sizes.
Figure 5.26 Total Decryption Timings for ECC-TC

The total encryption and decryption timings for ECMV-TC are shown in Figures 5.27 and 5.28. Total encryption timings increases gradually as ‘t’, ‘n’ and key sizes are increased.

Figure 5.27 Total Encryption Timings for ECMV-TC

Total decryption timings for split after encryption vary significantly and most of the timings are contributed by ‘rK_s’ calculation. Considering total
encryption timings for all ECC-TC algorithms, it is observed that with increase in key size and (t, n), the encryption timings increases gradually.

**Figure 5.28 Total Decryption Timings for ECMV-TC**

![Graph showing Total Decryption Timings for ECMV-TC](image)

**Figure 5.29 Total Encryption Timings for ECC-TC and RSA-TC**

![Graph showing Total Encryption Timings](image)

In comparison with ECMV-TC, ECC-TC is efficient for both split before and after encryptions and hence can be used when sender has resource constraints. As shown in Figures 5.29 and 5.30, RSA-TC is much expensive in terms of encryption and decryption timings irrespective of ‘n’ and ‘t’ values as compared to ECC-TC. With ECC-TC, the increase in the encryption and decryption timings is gradual as
the key size and 'n' are increased. As against ECC-TC, the timings in RSA-TC increases exponentially with increase in key size.

![Total Decryption Timings](image)

**Figure 5.30** Total Decryption Timings for ECC-TC and RSA-TC

### 5.5 CONCLUSION

By using high speed and less complex ECC algorithm over binary fields, the three main security issues for Localization, Time Synchronization and Routing with data reliability for WSNs has been resolved for a resource constraint environment. The ECC based ToA Localization technique for Secure Localization and authentication was simulated. This technique was compared with the other public-key cryptographic schemes like RSA and MPRSA. A further comparison was done by implementing both Diffie-Hellman key exchange and ECC key exchange. The simulation results clearly indicate that ECC based ToA Localization technique is compatible to be used in a resource constraint environment like WSN.

The LBTS using ECC was proposed for static sensor network which is based on simplistic approach of sender/receiver synchronization. The global time synchronization model is adopted where all the sensor nodes synchronize their time to a common source. This technique synchronizes all the nodes with significant improvement in the synchronization error as Elliptic Curve encryption and
decryption is 30 times faster due to ring representation of field elements. However, the synchronization time degrades significantly with the increase in the number of hop count along with the increase in the nodes being deployed. The CBTS approach guarantees that normal nodes can synchronize their time to the cluster heads with decrease in the synchronization error. This is done by reducing the average hop count from the reference node to each node. The simulation results indicate that this approach is promising and is more suitable for the current generation of sensor networks.

Finally, ECC-TC based OLSR-MPR protocol was proposed. The MPR is the key idea behind the OLSR protocol to reduce the information exchange overhead. Instead of pure flooding, the OLSR uses MPR to reduce the number of nodes, which broadcasts the information throughout the networks so as to achieve less delay in the process of routing with better available bandwidth. OLSR-MPR protocol provides better performance in terms of average packet delivery ratio, end-to-end delay and average throughput received. RSA-TC, ECC-TC and ECMV-TC were simulated for OLSR-MPR protocol and their performances were compared. It was observed that RSA-TC scheme requires larger storage capacity and computational power. A better timing response and better security level were achieved by using ECC-TC. For sender with resource constraints, ECC-TC split before encryption was found to be more suitable. But in the case where receiver has resource constraints, ECMV-TC split before encryption is preferred due to its low decryption time. It is inferred that the encryption and decryption timings differ significantly for ECC and ECMV in both split before and after encryption scenarios.