CHAPTER 3

ENERGY EFFICIENT KEY DISTRIBUTION IN SENSOR NETWORK

If the sensors are deployed via random scattering (e.g. from an airplane), the network protocols cannot know beforehand, that the after deployment which nodes will be within communication range of each other. Even Deployment of nodes by hand, the large number of nodes concerned to predetermine the location of every individual nodes, makes it costly. Hence, any security protocol, should priory not suppose the knowledge of which nodes will be neighbors in a network.

Secure symmetric encryption will be extensively available in the Sensor Networks. Effective use of that secures symmetric encryption capability is a critical problem. As is always the case with symmetric encryption, proper key management is a fundamental concern.

So to distribute the keys among the sensor nodes in WSN symmetric approaches have been widely used or proposed, but usage of Public key approaches is not preferred due to its complex, power hungry and slow nature and not suitable for use in low power environment like wireless Sensor Network. Some work has been proposed using public key approaches like RSA, Diffie-Hellman and ECC, but they are not widely accepted. Due to its inherent characteristic with variant of services with confidentiality, the researcher continued and proposed so many other light weighted solutions for WSN using public key approaches.
The future in sensor security is the public key cryptography [10], because it is easy to distribute keys in public key cryptography than symmetric key cryptography. As well as it is also difficult to prove authentication for adversary in public key cryptography.

In this chapter we have presented and analyzed the key distribution techniques using public key methods in such a way that total energy consumption decreases as compared to conventional public key methods for secure distribution of keys among the sensor nodes. Our proposed scheme is based on preprocessing or offline processing [28] of the key distribution algorithms partially at base station. Here we are presenting a Public key method using different variants of Diffie-Hellman key exchanges; with Elliptic curve cryptography for distribution of symmetric keys among sensors for data confidentiality.

Here we have shown that the key distribution method not only successfully establish secure channel between peers nodes but also prove authentication to other nodes. The proposed method not only facilitates the required security services, but it also minimizes the energy consumption at sensor node. The preprocessing approach, pre-calculates the part of the security algorithm at base station and minimizes the computation effort on the sensor node.

The provided authentication during above said key distribution approach may be treated as first level of authentication (in case of multilevel authentication) discussed in the next chapter.
3.1 Issues of Key Distribution

These networks are frequently deployed in unattended environments, leaving these networks vulnerable to passive and active attacks by the adversary. The chat between sensors nodes can be eavesdropped by the adversary. The adversary can be aware of the conversation between the sensors and can forge the transmitted data. Sensor nodes should be resilient to these attacks.

To provide security for wireless sensor networks, communication should be encrypted and authenticated. An open research problem is how to secure the communications among sensor nodes in wireless sensor networks [8], in other words, how to set up faith among the participating entities? Cryptographic key computations can be used to set up trust between communication nodes, either using secret key or public key techniques. There are three types of general key agreement schemes: trusted-server scheme, self-enforcing scheme, and key pre-distribution scheme discussed later in this chapter.

Due to the resource constraints of wireless sensor networks and absence of TTL as KDC, trusted server scheme is not favored as compared to self-enforcing scheme and key pre-distribution system. The trusted-server scheme depends on a trusted server for key agreement among nodes, e.g., Kerberos [28]. However, the trusted-server scheme is not suitable for wireless sensor networks since there is usually no trusted infrastructure in sensor networks.

In key pre-distribution protocols, key information is distributed among all sensor nodes prior to deployment. Sensor nodes can communicate with
each other if they share a common key [27]. There exist many key pre-
distribution schemes discussed later.

The self-enforcing scheme depends on asymmetric cryptography, such
as key agreement using public key certificates. Though, limited computation
and energy resources of sensor nodes in wireless sensor networks often make
it unwanted to use public key algorithms, such as Diffie-Hellman key
agreement [25] or RSA [20].

Since Sensor nodes are resource constrained and run on battery, so
energy consumption should be low to make it operate for many days. In sensor
network security, the challenge is the design of protocols to bootstrap the
establishment of a secure communications infrastructure of sensor nodes with
some secret information, but have no prior direct contact with each other,
referring to this problem as the bootstrapping problem.

The next issue is the security in sensor networks. There can be many
types of attacks are possible in sensor network. The presented security issues
for sensor networks have not been addressed at all. It does not provide
assurance for replay attack, authentication, and confidentiality.

For implementing public key cryptography care should be taken in
sensor networks, because of the constrained of sensor network devices. The
symmetric methods are preferred to provide confidentiality, because it
consumes less energy for the generation of cipher text as compared to
asymmetric methods, like RSA, DHA that involve power function calculation
for the generation of key or cipher text.
3.2 Key Distribution and Management

This section discusses about key distribution techniques [11], [18]. The general key distribution problem refers to the task of distributing secret keys among communicating parties and to provide security properties such as secrecy and authentication.

In sensor networks, key distribution is typically combined with initial communication establishment to bootstrap a secure communication infrastructure from a collection of deployed sensor nodes. In this section, nodes have been pre-initialized with some secret information before deployment, but only after network setup, location of nodes can be determined. The node location determines, which nodes need to establish cryptographic keys with which other nodes, so we cannot set up these keys before deployment.

A bootstrapping protocol must not only enable a newly deployed sensor network to initiate a secure infrastructure, but it also allows nodes deployed at a later time to join the network securely [21]. Due to the many limitations this is a challenging problem of sensor networks hardware and software. The characteristics of sensor networks make difficult the design of secure protocols for sensor networks, and also make the bootstrapping problem difficult.

- **Requirements for bootstrapping security in sensor networks:** A bootstrapping scheme for sensor networks needs to satisfy the following requirements [11] [18].
• Deployed nodes must be able to establish secure node-to-node communication.
• Additional legitimate nodes deployed at a later time can form secure connections with already-deployed nodes.
• Unauthorized nodes should not be able to gain entry into the network, either through packet injection or masquerading as a legitimate node.
• The scheme must work without prior knowledge of which nodes will come into communication range of each other after deployment.
• The computational and storage requirement of the scheme must be low, and the scheme should be robust to DoS attacks from out-of-network sources.

3.3 Literature Survey

For Key distribution among sensor nodes several symmetric & asymmetric approaches have been developed. Symmetric approaches takes less energy for the establishment of the key compared to the asymmetric approaches [14]. Today the favored method of key distribution in most modern computer systems is via asymmetric cryptography, also known as public key methods.

But public key approaches are power hungry in nature. The traditional approaches take more energy for the establishment of symmetric keys among the sensor nodes. But due to its inherent features, now days it preferred over symmetric one.

Public key systems are generally recognized to have an upper hand in key distribution. A Certificate Authority is used for key management and has
its own public/private key pair. The CA's public key is known to every network node. The trusted CA is responsible to sign certificates, binding public keys to nodes, and has to stay online to verify the current bindings. The public key of a node should be revoked if this node is no longer trusted or leaves the network.

A single key management service for an ad hoc network is probably not an acceptable solution [21]. If the CA is unavailable, nodes cannot get the current public keys of other nodes to establish secure connections. In addition, if a CA is compromised, the attacker can sign erroneous certificates using the database of the private keys. Naive replication of CAs can make the network more vulnerable, since compromising of any single replica can cause the system to fail’. Hence, it may be more prudent to distribute the trust to a set of nodes by letting these nodes share the key management responsibility [3].

### 3.3.1 Key pre-distribution schemes

The key pre-distribution schemes are based on symmetric key cryptography. Following are the popular symmetric key distribution approaches discussed.

- **Using a single network-wide key:** The simplest method of key distribution is to pre-load a single network wide key onto all nodes before deployment. After deployment, nodes establish communications with any neighboring nodes that also possess the shared network key. This can be achieved simply by encrypting all communications in the shared network-wide key and appending a message authentication code (MAC) to ensure integrity [15].
The properties of the single network-wide key approach are as follows:

- It requires a small amount of memory.
- No additional protocol steps are required to set up the keys.
- It provides resistance against DoS and packet injection due to using MAC.

The main drawback of the network-wide key approach is that the compromise of a single node causes the compromise of the entire network.

**Using pair wise-shared keys:** In this approach, every node in the sensor network shares a unique symmetric key with every other node in the network. Hence, in a network of n nodes, there are a total of \( n(n-1)/2 \) unique keys. Every node stores \( n-1 \) keys, one for each of the other nodes in the network.

After deployment, nodes must perform key discovery to verify the identity of the node that they are communicating with.

The properties of this approach are as follows [7]:

- It provides Perfect resilience to node capture.
- Compromised keys can be revoked by simply broadcasting its entire set of \( n-1 \) pair wise keys to the network. No authentication is necessary.
- The pair wise keys scheme achieves many of the benefits of using symmetric cryptography without needing dedicated hardware or software to before the more complex asymmetric cryptographic primitives.
The main problem with the pair wise keys scheme is poor scalability [17].

- **Bootstrapping security off a trusted base station:** This method of key distribution uses a trusted, secure base station as an arbiter to provide link keys to sensor nodes. The sensor nodes authenticate themselves to the base station, after which the base station generates a link key and sends it securely to both parties. An example of such a protocol is part of the SPINS security infrastructure.

  Prior to deployment, a unique symmetric key is generated for each node in the network. This node key is stored in the node's memory and will serve as the authenticator for the node as well-as facilitate encrypted communications between the node and the base station. The base station has access to all the node keys either directly or indirectly.

  This method, unlike the other methods mentioned previously, assumes some level of reliable transport is available between the node and the base station before any key establishment has taken place. Since this transport occurs before any security primitives are in place, it will necessarily have to be assumed as insecure, however, as long as it is reliable in a way such that a small number of malicious nodes are unable to prevent the transmission of messages to and from the base station then the protocol presented here is viable. The properties of this method of key establishment are as follows.

  - Small memory requirement
  - Perfect resilience to node capture:
3.3.2 Random key pre-distribution scheme

Eschenauer and Gligor first proposed a random key pre-distribution scheme [12] [18]. Let \( m \) denote the number of distinct cryptographic keys that can be stored on a sensor node. Before sensor nodes are deployed, an initialization phase is performed. In the initialization phase, the basic scheme picks a random pool (set) of keys \( S \) out of the total possible key space. For each node, \( m \) keys are randomly selected from the key pool \( S \) and stored into the node's memory. This set of \( m \) keys is called the node's key ring. The number of keys in the key pool, \( S \), is chosen such that two random subsets of size \( m \) in \( S \) will share at least one key with some probability \( p \).

After the sensor nodes are deployed, a key-setup phase is performed. The nodes first perform key-discovery to find out with which of their neighbors they share a key. Such key discovery can be performed by assigning a short identifier to each key prior to deployment, and having each node broadcast its set of identifiers. Nodes which discover that they contain a shared key in their key rings can then verify that their neighbor actually holds the key through a challenge response protocol. The shared key then becomes the key for that link.

- **SPINS: SNEP AND \( \mu \)TELSA**

  **SPINS:** Security protocols for Sensor Networks comprised of Sensor Network Encryption Protocol (SNEP) and \( \mu \)TELSA [2]. The functions of SNEP are to provide confidentiality (privacy), two party data
authentication, integrity and freshness. µTELSA is to provide authentication to data broadcasts.

**SNEP:** In SNEP each node j share a unique master key $K_j$ with the base station. This master key is used to derive all other keys. For data encryption SENP employs a onetime key produced by using a key $K_j$ and an incremental counter as input to the RC5 cryptographic algorithm. The RC5 algorithm outputs a binary string that is used as the one time key. The message is XORed with the Onetime Key, transmitted and the counter is incremented in preparation for the next message. The base station, aware of the node’s counter value and the derived key, produces the identical one time key, XORs the encrypted message with the onetime key to produce the clear text.

The first drawback of this protocol is the distribution of the master key and management of the key at base station. Secondly if the master key is compromised there will be no security.

**µTELSA:** It is a broadcast authentication technique that achieves asymmetric properties and thus enables low powered nodes to perform source authentication. µTELSA requires shared symmetric key between the base station and each of the nodes. That key work as a seed value for generation of MAC.

µTELSA has the drawback of requiring a shared symmetric key between a base station and each of the sensor nodes.
- **Eschenauer-Gligor's Random Key Pre-Distribution Scheme**: It contains following-
  - `q-Composite random pre-distribution [12]` – It improves the flexibility of the original scheme by requiring that nodes share q keys to set up secure channels.
  - **Bivariate polynomial scheme [7]** - A bivariate, commutative, polynomial \( f(x; y) = f(y; x) \) is defined for the network and each node is preloaded with \( f(i; y) \), where \( i \) is the node's identity. To establish shared key nodes must compute the value of the polynomial by replacing \( y \) with the destination's identity. For the attacker to compromise the network the polynomial must be solved which requires \( t + 1 \) node to be compromised with \( t \) being the polynomial's degree. A more thorough review of key pre-distribution in WSNs is provided in [20].

- **Blom's X-Secure Key Pre-Distribution Scheme**: This key pre-distribution method allows every pair of nodes in a network to be able to find a pairwise secret key by generating a \((X+1)\times N\) matrix \( G \) over a finite field \( GF(g) \), where \( N \) is the size of the network. Providing no more than \( X \) nodes are compromised, the network is perfectly secure [29].

- **Polynomial-Based Key Pre-Distribution Scheme**: A polynomial-based key pre-distribution technique was developed for group key pre-distribution scheme [23]. Polynomial-based key pre-distribution schemes allow any group of \( t \) parties to compute a common key. These schemes focus on saving communication costs while memory constraints are not placed on group members.
3.3.3 Limitations of Current Key Pre-Distribution Schemes

Current schemes only think parts of system parameters, such as memory size, battery power, and computation capacity generally. Some important application based parameters, such as lifetime of network, requisite security level of different applications are not taken into account and not discussed sufficiently and charily. There are still no organized studies of all the different models for security of wireless sensor networks.

3.4 Perception of Public key distribution scheme

The common perception of public key cryptography is that it is complex, slow and power hungry, and as not at all suitable for use in low power environment like wireless Sensor Network. But in this chapter as well as in this thesis, we challenge the basic assumption about public key cryptography in Sensor Networks that are based on the traditional software based approach [10]. We can employ public key cryptography in Sensor Network for security, provided we use careful optimization and low power design techniques.

A pleasing implementation of asymmetric algorithms giving satisfying performance together with minimal memory consumption is yet missing. As cryptographic primitives are the fundamental building blocks of every secure protocol, the knowledge of algorithm usability is crucial for the design of new protocols for sensor networks [5].

However, asymmetric cryptography has following disadvantages:

- **Reliance on asymmetric key cryptographic hardware or software:** All known asymmetric cryptographic schemes often engage computationally
intensive mathematical functions, such as the modular exponentiation of large numbers. Basic sensor node CPU hardware, though, is extremely limited, some even lack an integer multiply instruction. Sequentially to implement asymmetric cryptography on sensor nodes, it is necessary to either incorporate dedicated cryptographic hardware on a sensor node, thus increasing its hardware cost, or encode the mathematical functions in software, which is usually 3 to 5 times slower than symmetric cryptographic functions. Agreed that asymmetric cryptography is only used for key setup upon node deployment, this represents a tiny fraction of the sensor node's lifetime. Significantly increasing the cost of a node is thus difficult to justify.

- **Vulnerability to denial-of-service:** Asymmetric cryptographic operations engage significant amounts of computation and it can consume few seconds up to minutes of processing for a sensor node to complete one signature generation or verification. Thus, the nodes are vulnerable to a battery exhaustion denial of service attack where they are continuously flooded with illegal signatures, or requested to generate a signature. Since this denial-of-service can only occur during the key establishment phase, the sensor network is only vulnerable when it is newly deployed or when additional nodes are being added to the network. Increased monitoring of the site for adversarial transmissions during those times could alleviate the DoS problem.

- **No resistance against node replication:** Once a node is captured, its key pair can be used to set up links with every single one of the nodes in the network, effectively making the node omnipresent. This could potentially put it in a position to subvert the routing infrastructure. We could prevent such an attack with some added countermeasures. For example, each time a node forms a connection with another node, they could both report the event to a base
station encrypted using their master public key. Any node that has an exceptionally high degree could then be immediately revoked.

Following result shows the energy consumption of public key algorithm compared with the symmetric one for same security level [24][9].

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Energy Consumed</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA-1024 (Public key approach)</td>
<td>397.7 µJ</td>
</tr>
<tr>
<td>AES-128 (symmetric key approach)</td>
<td>2.49 µJ</td>
</tr>
</tbody>
</table>

Above table shows the power hungry nature of public key approaches. Following are the popular approaches for distribution of keys in Public key environment.

### 3.5 Key establishment using public key method

A brief outline of a possible public-key method for sensor networks is as follows. Proceeding to the deployment, a master public/private key pair is first generated [5]. Then, for every node A, its public/private key pair is generated. This key pair is stored in node A's memory along with the master public key and the master key's signature on A's public key. Once all the nodes are initialized in this fashion, they are ready for deployment.

Once the nodes have been deployed, they perform key exchange. Nodes exchange their respective public keys and master key signatures. Each node's public key is verified as legitimate by verifying the master key's signature using the master public key. Once the public key of a node has been received,
a symmetric link key can be generated and sent to it, encrypted by its public key. Upon reception of the session key, key establishment is complete and the two nodes can communicate using the symmetric link key.

The properties of this approach are as follows:

- **Completely resilient against node capture**: Capture of any number of nodes does not represent any additional communications in the network, since these nodes will have no knowledge of any secret keys besides the ones that they are actively using.

- **Probable to revoke known compromised key pairs**: This can be performed by broadcasting a revocation message, signed by the master key. Nodes receiving the broadcast can authenticate it as coming from the legitimate authority and ignore any future communications purporting to originate from the revoked key pair.

- **Fully scalable**: Signature schemes function just as efficiently regardless of the number of nodes in the network.

### 3.6 Distributed key management Schemes

Centralized key management techniques are frequently effective in the context of static group membership with the underlying transfer medium being dependable. Although quite simple, centralized approach to key management has a single point of failure and attack. That is, an active adversary needs only to compromise the key manager to influence the security of the entire group. Another drawback of centralized techniques is that they regularly require a secure channel to transmit the group key from the group controller to each
member. Given the need of infrastructure and public transfer medium of ad hoc networks, this requirement is excessive.

Distributed key management techniques need that two or more group members contributes for the generation of the group key. After receipt of the public value(s), each group member uses its own secret to generate the actual group key. Most forms of distributed key management and agreement are based on a generalization of the well-known Diffie-Hellman algorithm and Elliptic Key Cryptography, which are discussed in this section.

### 3.6.1 Diffie-hellman key agreement

The Diffie-Hellman key agreement protocol is proposed by Whitfield Diffie [25] gets through of the twentieth century having distributive key management. Distinct from other cryptosystems, the Diffie-Hellman protocol provided a way for two parties to agree on a secret key/ group key and use it to communicate over an unconfident medium in an ad hoc fashion.

- **Diffie-Hellman Key Agreement**
  1. A chooses its secret $K_R$, and generates $K_u = \alpha^{KR} \mod p$ and sends it to B. where A and B are the protocol participates sensors, $K_R$ and $K_U$ are the respective private and public keys and $\alpha$ and $p$ are the global public elements.
  2. Like-wise B chooses its private key, generate and send its public key to A.
  3. A computes $K = K_{AB} = (private\ key\ of\ A\ (K_R))^{public\ key\ of\ B\ (K_u)}$
4. In the same way B compute the same key $K = K_{BA} = (\text{private key of } B (K_R))^{\text{public key of } A (K_a)}$

- **Analysis of Diffie-Hellman Key Agreement**

Despite the widespread acceptance of Diffie-Hellman, it is vulnerable to the man-in-the-middle attack. The above attack is made feasible by the need of authentication in Diffie-Hellman. Following table shows the complexity of DHA where n is number of participating nodes.

**Table 3.2: Complexity analysis of DHA**

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of round</td>
<td>1</td>
</tr>
<tr>
<td>Number of messages</td>
<td>$2n(n-1)/2$</td>
</tr>
<tr>
<td>Exponentiations per member</td>
<td>2</td>
</tr>
<tr>
<td>Total exponentiations</td>
<td>$n^2$</td>
</tr>
<tr>
<td>Total message size</td>
<td>$N$</td>
</tr>
</tbody>
</table>

### 3.6.2 N-Party Diffie-Hellman Key Agreement

The revised protocol, henceforth referred to as Generalized Diffie-Hellman (GDH) [3], is nearly identical to its predecessor: members agree on an a priori $G$ and $\alpha$; each member then generates its own secret $N_i \in G$.

- **The Protocol Notation**: Let $N$ is the number of protocols participants, that having; $i, j, k \in [1, n], M_i$ is the $i^{th}$ group member, $i \in [1, n], N_i$ is the random secret chosen by the member $M_i$, $q$ is the order of the algebraic group, $p$ is the large prime number, $G$ is the unique subgroup of $Z_p$ of order $q$ with $p, q$
prime, $\alpha$ is the exponentiation base - the generator in the group $G$, $K_n$ is the group key$^{[23]}$ shared by $n$ members.

- **Key Agreement:** The Generalized Diffie-Hellman consists of two stages - upflow and downflow. Each member's contributions are composed during the upflow stage, and the resultant intermediate values are broadcast in the downflow stage to the group.

- **Setup:** The setup for GDH is identical to that of two-party Diffie-Hellman: all participants, $M_1, \ldots, M_n$ choose a cyclic group, $G$, of order $q$, and a generator, or in $G$; each member then chooses a secret share, $N_i \in G$.

- **Upflow:** During the upflow, each member $M_i$ performs a single exponentiation, appends it to the flow, and forwards the flow to $M_{i+1}$.

$$M_i \stackrel{\alpha \prod_{k \in [1,i]} K_{\epsilon[i,k]}}{\longrightarrow} M_{i+1}$$

The upflow stage terminates and the downflow commences when $M_n = M_i$ - when the last member has received the upflow.

$$M_{n-1} \overset{(\alpha^{N_1}, \alpha^{N_1} N_2, \ldots, \alpha^{N_1}, \alpha^{N_1} N_2, \ldots, N_{n-1})}{\longrightarrow} M_n$$

Upon receipt of the upflow, $M_n$ calculates the new group key, $K_n$, by exponentiation of the last intermediate value in the flow:

$$K = K_n(\alpha^N 1^N 2, \ldots, n - 1)^N n$$

Once $K_n$ has been calculated, $M_n$ commences the downflow.

- **Downflow**

  The downflow is initially comprised of following $n-1$ intermediate values

  $$(\alpha^N 1^N n, \ldots, \alpha^N 1^N 2^N n, \ldots, \alpha^N 1^N 2^N n - 2^N n)$$
Exponentiated to the $n^{th}$ group member's secret $N_n$. $M_n$ sends the downflow to $M_{n-1}$

$$M_{n-1}^{(\alpha^{N_1N_n}, \ldots, \alpha^{N_1N_2N_n}, \ldots, \alpha^{N_1N_2, \ldots, N_{n-2}N_n})} M_n$$

- Upon receipt of the downflow, each member, $M_i$, removes its own intermediate value as follows

$$(\alpha^N 1^{N_2}, \ldots, i - 1^N i + 1 \ldots \ldots N_n)$$

- It calculates the group key, $K_n = (\alpha^{N_1N_2 \ldots, N_{i-1}N_{i+1} \ldots N_n})^N_i$, exponentiates the remaining $i-1$ intermediate values in the flow, and forwards the flow to its predecessor, $M_{n-1}$

$$(\alpha^N 1^N n^n n - 1, \ldots, ^N i + 1^N i, \alpha^N 1^N n^n n - 1, \ldots, ^N i + 1^N i)$$

$M_{i-1} \leftarrow M_i$

### 3.6.3 Analysis of N-Party Diffie-Hellman Key Agreement

Possibly the most notable strength of GDH is the ease with which it permit group members to generate and distribute a shared group key over an insecure medium. Unfortunately, the essential form of GDH is computationally complex, and does not scale to large group networks these requirements may be restrictive in some ad hoc networking environments. Following table shows the complexity of N-party DHA where $n$ is number of participating nodes.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of rounds</td>
<td>$n-1$</td>
</tr>
<tr>
<td>Number of messages</td>
<td>$n(n-1)$</td>
</tr>
<tr>
<td>Exponentiations per member</td>
<td>$N$</td>
</tr>
</tbody>
</table>

Table 3.3: Complexity analysis of N-GDH
### 3.6.4 The Ingemarsson Protocol

The previously discussed generalized version of the Diffie-Hellman algorithm [13], Prior to its publication, a number of more limiting variations on Diffie-Hellman have been presented; the Ingemarsson et al, protocol was the first of these attempt. The performance and applicability of this protocol has since been surpassing by more effective protocols.

- **Design**: The design of the Ingermasson protocol requires that the group members be organized as a logical ring. Each members of the group receives the intermediate value from its predecessor node, exponentiates using its own share, and forwards the result to its successor. After n-rounds, the protocol is complete, and each member calculates the group key.

- **The Protocol Setup**: Earlier to the first round of the protocol, all group members must synchronize in a ring. This step requires each member to be apprised of its successor and predecessor, in addition to the start time of the first round.

- **Key Agreement**

\[
M_i \xrightarrow{\alpha(I^N_j, k)} M_{i+1} \mod n
\]

- **Analysis**: The complexity analysis indicates, the Ingemarsson protocol is quite slow - the slowest of all protocols. As well, many of the design requirements make the resulting protocol quite restrictive, especially for ad hoc networks.

<table>
<thead>
<tr>
<th>Total exponentiations</th>
<th>(n^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total message size</td>
<td>(n(n-1))</td>
</tr>
</tbody>
</table>
• **Complexity:** The Ingemarsson protocol requires n-1 rounds. At each round, n messages are sent - one by each member node. This yields the complexity of this process (the input parameter, n, denotes the number of group members) is

**Table 3.4:** Complexity analysis of Ingemarsson Protocol

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of round</td>
<td>n-1</td>
</tr>
<tr>
<td>Number of messages</td>
<td>n(n-1)</td>
</tr>
<tr>
<td>Exponentiations per member</td>
<td>N</td>
</tr>
<tr>
<td>Total exponentiations</td>
<td>n^2</td>
</tr>
<tr>
<td>Total message size</td>
<td>n(n-1)</td>
</tr>
</tbody>
</table>

### 3.6.5 The Burmester and Desmedt Protocol

The Burmester and Desmedt protocol [16] presents a much faster variation of the GDH. The setup phase of the Burmester and Desmedt protocol is the same to that of basic GDH; however, the group key construction is significantly different.

#### The Protocol

Each participant, \( M_i, i \in [1, n] \) executes the following rounds:

1. \( M_i \) generates a secret, \( N_i \), and broadcasts the new key, \( z_i = \alpha^{ni} \), to all group members.
2. Each member, \( M_i \) computes and broadcasts
   \[
   X_i = \left( \frac{Z_{i+1}}{Z_{i-1}} \right) N_i
   \]
$M_i \mod n$ computes the new group key, $K_n$ as:

$$K = K_n = Z_{i-1}nN_i \times X_i^{n-1} \times X_{i+1}^{n-2} \ldots X+i-2 \mod p$$

### Complexity

**Table 3.5:** Complexity analysis of Burmester and Desmedt Protocol

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of round</td>
<td>2</td>
</tr>
<tr>
<td>Number of messages</td>
<td>$2n$</td>
</tr>
<tr>
<td>Exponentiations per member</td>
<td>$N$</td>
</tr>
<tr>
<td>Total exponentiations</td>
<td>$N(n+1)$</td>
</tr>
<tr>
<td>Total message size</td>
<td>$2n$</td>
</tr>
</tbody>
</table>

- **Analysis:** Like the Ingemarsson protocol, the Burmester protocol requires $n+1$ exponentiations per group members; however, each exponentiation is considerably less complex.

#### 3.6.6 GDH.2

GDH.2 is a refinement of GDH.1. Like all CLIQUES protocols, GDH.2 consists of an upflow and downflow. $N-1$ contributions are collected during the upflow stage; and the resulting intermediate values are broadcast to the group during the down flow.

- **Setup**

Members of the group select $p$, $q$, $G$, and $a$. In addition, each member is assigned a sequential identifier in $1...n$. $M_n$ assumes the role of the group controller.
• **Key Agreement**

Each $M_i$ selects a random secret, $r_i \in \mathbb{Z}_q^*$. 

$M_i$ receives the upflow, exponentiates each intermediate value, adds a new intermediate value that excludes its own contributions, updates the cardinal value, and forwards the upflow to $M_{i+1}$:

$$M_i \xrightarrow[\alpha^{n\ldots r_j/r_j\ldots j\in[1,i]\alpha^{\eta\ldots \eta}}]{} M_{i+1}$$

$M_n$ receives the upflow from $M_{n-1}$, calculates the group key from the cardinal value, exponentiates all intermediate values, and broadcast the revised intermediate values to the group:

$$\text{ALL} \xrightarrow[\alpha^{1\ldots n\mid i\in[1,n]\alpha^{\eta\ldots \eta}}]{} M_n$$

Each member receives the broadcast message from $M_n$ extracts its intermediate value, and exponentiates it using its secret, $r_i$ to calculate the group key, $K$:

$$K = (\alpha^{r_i} \cdot 2^{r_i} - 1^{r_i} + 1 \ldots n^{r_i} - 1^{r_i} n^{r_i} - 1)^{r_i}$$

• **Complexity**

**Table 3.6**: Complexity analysis of GD.2 Protocol

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Number of round</em></td>
<td>$N$</td>
</tr>
<tr>
<td><em>Number of messages</em></td>
<td>$N$</td>
</tr>
</tbody>
</table>
3.6.7 GDH.3

The GDH.3 protocol targets at reducing the number of exponentiations required by each member. Reducing the number and relative expense of exponentiations is of paramount importance in ad hoc networking context wherein the nodes have constrains on their computational capabilities.

- **Setup**

Members of the group select p, q, G, and a. In addition, each member is assigned a sequential identifier in 1...n. M_n assumes the role of the group controller.

- **Key Agreement**

1. The first state is identical to the upflow of GDH. 1: each member receives the upflow, adds its contribution, and forwards it to its successor.

\[
M_i \xrightarrow{\alpha^{\Pi_k \cdot N_k | k_{e[1,i]}}} M_{i+1}
\]

2. M_n receives the upflow from M_{n-1}, calculates the group key from the cardinal value, exponentiates all intermediate values, and broadcast the revised intermediate values to the group:

<table>
<thead>
<tr>
<th>Exponentiations per member</th>
<th>((i + 1) - O(n))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total exponentiations</td>
<td>(n(n + 2)/2 - 1)</td>
</tr>
<tr>
<td>Total message size</td>
<td>((n - 1)\left(\frac{n}{2} + 2\right) - 1)</td>
</tr>
</tbody>
</table>
\[ M_i \xlongequal{\alpha / (\tau_k \mid k \in [1, n-1])} M_{n-1} \]

3. The third stage is the \( n \)th round. It entails a response from every group member.

\[ M_i \xlongequal{\alpha / (\tau_k \mid k \in [1, n-1] : k \neq 1)} M_n \]

4. In the final stage, \( M_n \) collects all responses, exponentiates them to \( \tau_n \) and broadcasts all intermediate values to the group.

\[ M_i \xlongequal{\alpha / (\tau_k \mid k \in [1, n-1] : k \neq i \mid i \in [1, n-1])} M_n \]

**Complexity**

**Table 3.7:** Complexity analysis of GD.3 Protocol

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of round</td>
<td>( n+1 )</td>
</tr>
<tr>
<td>Number of messages</td>
<td>( 2n-1 )</td>
</tr>
<tr>
<td>Exponentiations per member</td>
<td>4</td>
</tr>
<tr>
<td>Total exponentiations</td>
<td>5( n-6 )</td>
</tr>
<tr>
<td>Total message size</td>
<td>3(n-1)</td>
</tr>
</tbody>
</table>

**Analysis:** The GDH.3 protocol gives considerable enhancements over GDH.1 and GDH.2 with its constant message sizes and reasonably fewer exponentiations per member. It should be noted, though, that GDH.3 does require the group controller, \( M_n \), to perform \( n \) exponentiations.

**3.6.8 Comparison of Message complexity of different variants of GDHA**

Following graph shows the complexity of messages for different Diffie-Hellman key distribution techniques.
3.6.9 The Tree-Based Generalized Diffie-Hellman Protocol

The Tree based Generalized Diffie-Hellman (TGDH) key agreement protocol seeks to improve this performance by structuring the key generation hierarchically rather than linearly. The tree-based key agreement algorithm provides an interesting contrast to CLIQUES because the actual algorithm used to generate the keys (Diffie-Hellman) is identical; the only major difference between the two is the structure of the contributing nodes.

- **Design**

Each node in the key tree, \( <l, v> \), has a key, \( K_{l,v} \), a blinded key, \( BK_{l,v} \), and member, \( M_l \), associated with it. A blinded key is simply the modular exponentiation of the key in prime order groups (i.e., \( BK_{l,v} = \alpha^k \mod p \)). Each \( K_{l,v} \) is computed recursively as follows:

\[
C_{l,v} = \alpha^K < 1 + 1,2v > *K < 1 + 1,2v + 1 > \mod p
\]

The co-path is composed of all of the siblings of nodes that appear in the key-path. A node, \( <l, v> \), uses the blinded keys in its co-path and its own key to derive the group secret. In order to perform this computation, each
node must, at a minimum, know all of the keys in its key-path and all of the blind keys in its co-path.

3.7 Problem Statement

Distribution of secret key for establishment of the secure channel in sensor networks is a challenging task. Due to the absence of availability of any kind of infrastructure in sensor networks, the key distribution process became tricky. For distributing the keys among the sensor nodes or with the base station, various symmetric and public key approaches have been presented as discussed above, as key pre-distribution and self enforcing schemes. Asymmetric or public key approaches are more adoptable and acceptable solution for distribution of the keys as compared to symmetric approaches.

But a care to be taken in the implementation of public key approaches in sensor networks due to its power hungry nature. Key distribution using asymmetric approaches consumes more power for generation of secret keys, due to its slow nature and power function calculation like RSA, DHA. The next section contains proposed energy efficient public key approaches for the generation of secret keys in sensor networks.

3.8 Proposed solution for key distribution

So considering the above problem, we have proposed energy efficient key distribution approaches using public key cryptography in this chapter. In this chapter we have presented different variants of Energy efficient Diffie-Hellman key distribution techniques along with Energy efficient Elliptic curve cryptography algorithm for establishment of keys between the nodes in Adhoc sensor network.
Here we have shown that by identifying and shifting the key generation steps from sensor node to base station saves significant amount of time/energy of nodes. Also by choosing the public key approaches for key distribution provides the authentication in our networks.

Our proposed solution is based on public key algorithm because Public key approaches are more suitable and acceptable as compared to symmetric one for key distribution [8] as stated. The preprocessing or offline processing [28] of the key distribution algorithm partially minimizes the energy consumption at sensor node after deployment. Here we have chosen various versions of Diffie-Hellman key exchange methods discussed above and elliptic curve cryptography because of having effectiveness on the difficulty of the computing discrete logarithm property [26]. This property ensures that by getting the values of GPE and public Key, it is not feasible for an opponent to find the private key. This can be treated as first level of authentication (in case of multilevel authentication) achieved by using the above said key distribution.

Our method is proposed for light weighted and resource constrained sensor nodes. Here each node is limited to broadcast the message to only its neighbors. Our algorithm has three phases

(1) Before deployment of the Sensor Nodes,

(2) After deployment of the Sensor Nodes,

(3) Addition of a new node in exiting network.
3.8.1 Flow Diagram of proposed Key Distribution

In the proposed diagram the security algorithm is broken to process offline and online on base station and sensor node respectively.

![Flow Diagram of proposed Key Distribution]

**Operation at Base Station (Before Deployment)**

- Exchange of Public keys to its Neighbor sensor
- Calculation of different secret key between neighboring sensors
- Is Network Scalable

**Operation at Each Sensor Node (After Deployment)**

- Keep the Global element
- Discard the Global element

Figure 3.2: Proposed Key distribution Process

3.8.2 Energy Efficient Key distribution using Diffie-Hellman

For Key distribution, we have proposed different variations of key distribution techniques using Diffie-Hellman in such a way that total energy consumption decreases for distribution of the symmetric key. The proposed scheme discussed as follows named

The proposed scheme for key distribution is based on variants of Diffie-Hellman key agreement as discussed in above section. The following three phases of the proposed method.
i **Before deployment of the Sensor Nodes:** Step 1 & 2 of GDHA is performed at base station for each sensor consist of following

- Selection of Private Key., Generation of public keys and Loading the credentials containing corresponding private and public keys with Global public element at each sensor.

ii **After deployment of each Sensor Nodes:** It consists of establishment of secret key between any pair of the neighboring sensor nodes. It consist of following steps

- Exchanging of public keys between neighboring nodes.
- Calculation of secret key (that will be different for each pair) as discussed in step 3 & 4.

iii **Addition of a new node in exiting Network:** Now if a new Sensor Node is deploys to the exiting one having the same credentials as mentioned above, it exchange its public key to its neighbor and establish a new secret key.

Following are the implementation assumption taken.

- Experiments are carried out at 8051 microcontroller as sensor processor with XLAT=11.0592 MHz.
- Continuous power supply i.e. resistance, capacitance & inductance are static 12 clock per machine cycle.

### 3.8.3 Result & Analysis of EEDHA

Following table shows the time for the generation of public key involved in power function calculation for DHA. Here this power function calculation at node is not necessary to compute, so shifted it to the base station.
Following table shows the total time taken by the sensor node to generate the public key with machine cycle required and time taken in µs.

**Table 3.8:** 8051 pseudo code of generation of key using exponentiation function in DHA

<table>
<thead>
<tr>
<th>CODE</th>
<th>Machine Cycle required</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Power_function:</strong> ORG 500</td>
<td>gre</td>
<td></td>
</tr>
<tr>
<td>MOV A, #01</td>
<td>1</td>
<td>1.085</td>
</tr>
<tr>
<td>MOV R1, # Private key of sender/Receiver</td>
<td>1</td>
<td>1.085</td>
</tr>
<tr>
<td>Again: MOV B, # Global public key component( primitive root of q)</td>
<td>1</td>
<td>1.085</td>
</tr>
<tr>
<td>MUL AB</td>
<td>4</td>
<td>4.34</td>
</tr>
<tr>
<td>MOV, # Global public key component q (a prime number)</td>
<td>4</td>
<td>4.34</td>
</tr>
<tr>
<td>DIV AB</td>
<td>1</td>
<td>1.085</td>
</tr>
<tr>
<td>MOV A,B</td>
<td>2</td>
<td>2.17</td>
</tr>
<tr>
<td>DJNZ R1, Again</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The total Time taken for executing the above code on the node is \((12/11.0592) * 13\) private keys of neighboring. Following figure shows the saved time of processor in a varied network of sensor nodes in our proposed method.
3.8.4 Complexity of Proposed EEDHA

Following table shows the complexity of EEDHA where n is number of participating nodes.

**Table 3.9: Complexity analysis of EEDHA**

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of rounds</td>
<td>1</td>
</tr>
<tr>
<td>Number of messages</td>
<td>(n(n-1)/2)</td>
</tr>
<tr>
<td>Exponentiations per member</td>
<td>1</td>
</tr>
<tr>
<td>Total exponentiations</td>
<td>(N)</td>
</tr>
<tr>
<td>Total message size</td>
<td>(N/2)</td>
</tr>
</tbody>
</table>

Following figure shows the public key calculation and secret key calculation time with increasing key length. Here the blue line shows the public key calculation time and the red show the time taken for the secret key generation.
Figure 3.4: Comparison between Time taken by secret and public key generation by DHA

The above analysis shows that by shifting the power function calculation step from the sensor node to base station, we can save a significant amount of time.

3.8.5 Energy efficient Key Agreement for the other variants of Diffie-Hellman

The proposed algorithm is based on N-Party Diffie-Hellman Key agreement key agreement as discussed in above section. This method will generate a group key (start from a group of 2 to n nodes). The following three phases’ shows the proposed method.

Before deployment of the Sensor Nodes: For key establishment between the sensor the upflow and downflow process can be carried out at the base station because the group key formed between the sensor nodes is irrespective of their position after the deployment. Because the upflow process of group key generation does not depend, which nodes will be
neighbor of each other, so the whole up flow process of group key establishment can be carried out at base station as follows.

\[ M_{n-1} \overset{(\alpha^{N_1}, \alpha^{N_1N_2}, ..., \alpha^{N_1N_2\cdots N_n})}{\longrightarrow} M_n \]

This is carried out until the last node let \( M_n \) calculates the new group key, \( K_n \), by exponentiation of the last intermediate value in the flow:

\[ K = K_n(\alpha^N 1^{N-2}, \ldots, n - 1)^N \]

**ii After deployment of the Sensor Nodes:** After up-flow is completed, the down-flow starts. \( M_n \) node starts the down flow process, which propagates the down-flow message to its neighboring nodes that calculates a new group key as follows. This is carried out through the whole network until \( M_1 \) receives the message and calculate the key.

Exponentiated to the \( n^{th} \) group sensor node secret \( N_n \). \( M_n \) Sends the downflow to \( M_{n-1} \)

\[ M_{n-1} \overset{(\alpha^{N_1N_n}, \alpha^{N_1N_2N_n}, ..., \alpha^{N_1N_2\cdots N_{n-2}N_{n}})}{\longrightarrow} M_n \]

Upon receipt of the downflow, each sensor node, \( M_i \), removes its own intermediate value as follows

\[ (\alpha^N 1^{N-2}, \ldots, \alpha^N i - 1^N, i + 1 \ldots \ldots n) \]

**iii** It calculates the group key, \( K_n = (\alpha^{N_1N_2\ldots N_{i-1}N_{i+1}\ldots N_n})^{N_i} \), exponentiates the remaining \( i-1 \) intermediate values in the flow, and forwards the flow to its predecessor, \( M_{n-1} \)

\[ \overset{(\alpha^{N_1N_2\cdots N_{i-2}N_{i-2}\cdots N_{i+1}N_{i}N_n})}{\longleftarrow} M_i \]
iv **Addition of a new node in exiting Network**: A new node can be added easily to the existing network, by calculating a new group key again at the base station, so addition of new node does not take any sensor energy/time.

The rest of the different variations of Diffie-Hellman i.e. The Ingemarsson Protocol, The Burmester and Desmedt Protocol, GDH.2, GDH.3, The Tree-Based Generalized Diffie-Hellman Protocol also work in sensor network at the same pattern stated in EEGDH key agreement. So this key agreement protocol can also be used in energy efficient way, by performing key agreement process partially or completely at base station. So it can also save the time and energy at sensor nodes and increases the lifetime of network.

### 3.8.6 Energy efficient Elliptic Curve Cryptography using DHA

The motive behind choosing ECC for key distribution is that it consumes less energy as compared to other public key methods [1][22]. Following result shows that the energy consumption using Elliptic curve cryptography is less than the RSA [24] for providing the same security level.

**Table 3.10**: Energy comparison between RSA & ECC algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Energy Consumed for encryption</th>
<th>Energy Consumed for decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA 1024</td>
<td>397.7 µJ</td>
<td>390.3 µJ</td>
</tr>
<tr>
<td>ECC 160</td>
<td>93.7 µJ</td>
<td>93.3 µJ</td>
</tr>
</tbody>
</table>

The proposed method of key distribution using EEECC has following phases as before
i  **Phase 1: Before deployment of the Sensor Nodes**

- The Base Station select a large integer q, which is either prime number p or an integer of the form 2m and elliptic curve parameter a and b for following equation.

\[ Y^2 + xY = x^3 + ax^2 + b \]

This defines the elliptic group of points on X and Y axis. The base station also picks a base point G from the above points whose order is a very large value n.

- The base station select private value \( n_1, n_2, \ldots, n_N \) for sensor nodes 1, 2, \ldots, n respectively, which is less than n for every station and for itself also. These are the private keys for each of the sensor nodes and base station.

- The base station generates public keys for each of the sensor nodes and for itself by following equation.

\[ P = n * G \]

Where n and P are the private and public values of the nodes.

ii  **Phase 2: After deployment of each Sensor Nodes**

- Now every node of our static network broadcast their public value P to its neighboring nodes.

- Now every node calculate its secret key (that will be different for each pair) as follows

  let second node is the neighbor of the first node, so by using following equation 1 and 2 generate same key K (symmetric Key) at both the end.

\[ K = n_1 * P_2 \text{ (at node 1)} \text{ and } K = n_2 * P_1 \text{ (at node 2)} \]
Where $n_1$ & $n_2$ is the private key of node 1 and 2 respectively & $P_1$ and $P_2$ is the public key of node 1 and 2 respectively

- Now every node has a secret key to exchange the message to each other with its id. This show confidentiality, data integrity and authentication to each other.

iii Phase 3: Addition of a new node in exiting Network

- Now if a new Sensor Node is deploys to the exiting one with the same values that is $n_R$, $P_R$ and G It exchanges the public value $P_R$ and G to its neighbors.
- By using above method the neighbors generate the corresponding keys by using its previous value that is encrypted with its symmetric key.

3.8.7 Result and Analysis of EEECC

Following table shows the total time taken by the sensor node to generate the public key using a multiplication function. Assumption is same for the implementation as taken of Diffie- Hellman key distribution.
Table 3.11: 8051 pseudo code of generation of key using multiplication function in ECC

<table>
<thead>
<tr>
<th>Code</th>
<th>Machine cycle</th>
<th>Time taken of executing the Instruction (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power function: ORG 500</td>
<td>1</td>
<td>1.085</td>
</tr>
<tr>
<td>MOV A, #00</td>
<td>1</td>
<td>1.085</td>
</tr>
<tr>
<td>MOV R1, #</td>
<td>1</td>
<td>1.085</td>
</tr>
<tr>
<td>Private key of sender/Receiver</td>
<td>1</td>
<td>1.085</td>
</tr>
<tr>
<td>MOV B, # Global public key component G</td>
<td>2</td>
<td>2.17</td>
</tr>
<tr>
<td>Again: ADD A, B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DJNZ R1, Again</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The total Time taken for executing the above code at node is \((12/11.0592)\) * 3* private keys of sensors.

So above is the total execution time saved at sensor nodes by shifting that multiplication function to the base station. And if the sensor nodes are large enough, we can save substantial amount of energy at the node, as shown in the following graph. Let the private key is 255 at both the end. Following Figure shows the time saved by the sensor for running above code as the number of sensors increases.
The above results of proposed EEDHA (with its all variants) and EEECC saves a significant amount of time/energy at node by pre-computation of the security algorithm at base station partially.

3.9 Conclusion of the chapter

In this chapter we have presented different variants of Energy efficient Diffie-Hellman key distribution techniques along with Energy efficient Elliptic curve cryptography schemes for establishment of keys between the nodes in Adhoc sensor network in energy efficient manner.

Here we have shown that by identifying and shifting the key generation steps from sensor node to base station saves significant amount of time/energy of nodes. Also by choosing the public key approaches for key distribution provides the authentication in our networks.

The advantage of using proposed method over existing method is that the energy consumption decreases significantly as compared to traditional approaches due to pre-processing or offline computation [28] of security