ISSN:0975-9646

CEBKST: The Cost Efficient Based On Keying and Secure Data Transmission for Wireless Sensor Network

Narahari A, M. Preethi
Computer Science and Engineering
Kakatiya Institute of Technological Sciences Warangal.Andhra Pradesh, India.

***Abstract*--The project deals with Designing secure network protocols and cost-efficient, for Wireless Sensor Networks (WSNs). It is a problem because sensors are resource-limited wireless devices. Since the communication cost is the most dominant factor in a sensor's energy consumption, Here we introduce a cost efficient based on keying and secure data transmission (CEBKST) scheme for Wireless sensor network that significantly reduces the number of transmissions needed for rekeying to avoid out of dated keys. In addition to the goal of saving energy, minimal transmission is imperative for some military applications of Wireless sensor networks where an adversary could be monitoring the wireless spectrum. CEBKST is a secure communication framework where sensed data is encoded using a scheme based on a permutation code generated via the RC4 encryption mechanism. The key to the RC4 encryption mechanism dynamically changes as a function of the residual virtual energy of the sensor. Thus, a one-time dynamic key is employed for one packet only and different keys are used for the successive packets of the stream. The intermediate nodes along the path to the sink are able to verify the authenticity and integrity of the incoming packets using a predicted value of the key generated by the sender's virtual energy, thus requiring no need for specific rekeying messages. CEBKST is able to efficiently detect and filter false data injected into the network by malicious outsiders. The CEBKST framework consists of two operational modes, each of which is optimal for different scenarios. In CEBKST-I, each node monitors its one-hop neighbors where CEBKST-II statistically monitors downstream nodes. We have evaluated CEBKST's feasibility and performance analytically and through simulations. Our results show that CEBKST, without incurring transmission overhead (increasing packet size or sending control messages for rekeying), is able to eliminate malicious data from the network in an energy efficient manner.**

Key terms- Security, WSN Security, CEBKST, resource constrained devices.

I. INTRODUCTION

The WSN technology is no longer nascent and will be used in a variety of application scenarios. Typical application areas include environmental, military, and commercial enterprises. For example, in a battlefield scenario, sensors may be used to detect the location of enemy sniper fire or to detect harmful chemical agents before they reach troops. In another potential scenario, sensor nodes forming a network under water could be used for oceanographic data collection, pollution monitoring, assisted navigation, military surveillance, and mine reconnaissance operations. Future improvements in technology will bring more sensor applications into our daily lives and the use of sensors will also evolve from merely capturing data to a system that can be used for real-time compound event alerting .From a security standpoint, it is very important to provide authentic and accurate data to surrounding sensor nodes and to the sink to trigger time-critical responses (e.g., troop movement, evacuation, and first response deployment). Protocols should be resilient against false data injected into the network by malicious nodes. Otherwise, consequences for propagating false data or redundant data are costly, depleting limited network resources and wasting response efforts. However, securing sensor networks poses unique challenges to protocol builders because these tiny wireless devices are deployed in large numbers, usually in unattended environments, and are severely limited in their capabilities and resources (e.g., power, computational capacity, and memory). For instance, a typical sensor operates at the frequency of 2.4 GHz, has a data rate of 250 Kbps, 128 KB of program flash memory, 512 KB of memory for measurements, transmit power between 100 _W and 1 mW, and a communications range of 30 to 100 m. Therefore, protocol builders must be cautious about utilizing the limited resources onboard the sensors efficiently. In this paper, we focus on keying mechanisms for WSNs. There are two fundamental key management schemes for WSNs: static and dynamic. In static key management schemes, key management functions (i.e., key generation and distribution) are handled statically. That is, the sensors have a fixed number of keys loaded either prior to or shortly after network deployment. On the other hand, dynamic key management schemes perform keying functions (rekeying) either periodically or on demand as needed by the network. The sensors dynamically exchange keys to communicate.

LITERATURE SURVEY AND MOTIVATION

One significant aspect of confidentiality research in WSNs entails designing efficient key management schemes. This is because regardless of the encryption mechanism chosen for WSNs, the keys must be made available to the communicating nodes (e.g., sources and sink(s)). The keys could be distributed to the sensors before the network deployment or they could be redistributed (re-keying) to nodes on demand as triggered by keying events. The former is static key management and the latter is dynamic key management. There are myriads of variations of these basic schemes in the literature. In this work, we only consider dynamic keying mechanisms in our analysis since CEBKST uses the dynamic keying paradigm. The main motivation behind CEBKST is that the communication cost is the most dominant factor in a sensor's energy consumption. Thus, in this section, we present a simple analysis for the re-keying cost with and without the

transmission of explicit control messages. Re-keying with control messages is the approach of existing dynamic keying schemes whereas re-keying without extra control messages is the primary feature of the CEBKST framework.

Dynamic keying schemes go through the phase of re-keying either periodically or on demand as needed by the network to refresh the security of the system. With re-keying, the sensors dynamically exchange keys that are used for securing the communication. Hence, the energy cost function for the keying process from a source sensor to the sink while sending a message on a particular path with dynamic key-based schemes can be written as follows (assuming computation cost, E-comp, would approximately be fixed):

$$E_{Dyn} = (E_{Kdisc} + E_{comp}) * E[\eta h] * \frac{M}{\tau}$$

Where _ is the number of packets in a message, _ is the key refresh rate in packets per key; E_{Kdisc} is the cost of shared key discovery with the next hop sensor after initial deployment, and $E[\eta h]$ the expected number of hops. In the dynamic key-based schemes, _ may change periodically, on demand, or after a node-compromise. A good analytical lower bound for E½/h is given $E[\eta h] = \frac{D-tr}{E[dh]} \mid 1$

Where D is the end-to-end distance (m) between the sink and the source sensor node, tr is the approximated transmission range (m), and E½dh_ is the expected hop distance.

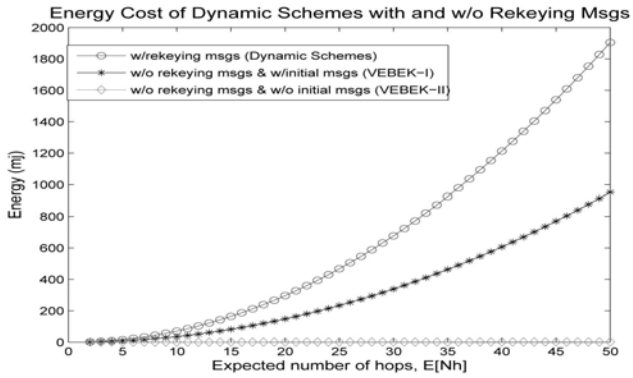

Fig1.keying cost of dynamic key-based schemes based on E(ηh)

$$E_{Kdisc} = \{E[N_e] * E_{node}\} * M - 2 * E_{node}\}$$

$$E_{node} = E_{tx} + E_{rx} + E_{comp},$$

Where E_{node} is the approximate cost per node for key generation and transmission, E½Ne_ is the expected number of neighbors for a given sensor, M is the number of key establishment messages between two nodes, and Etx and Erx are the energy cost of transmission and reception, respectively. Given the transmission range of sensors (assuming bidirectional communication links for simplicity), tr, total deployment area, A, total number of sensors deployed, N, E½Ne_ can be computed as

$$E[N_e] = \frac{N * \pi * tr^2}{A}$$

BLOCK DIAGRAM

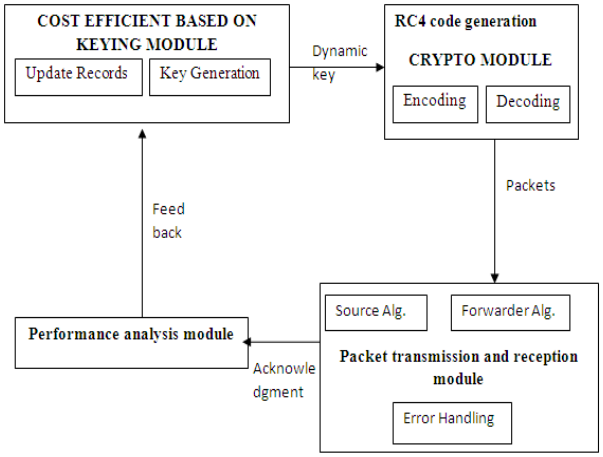

Fig2. Structure of CEBKST

II. MODULE DESCRIPTION

This project contains four major module .those modules are follow

1. Cost efficient based on keying module.
2. Crypto module.
3. Packet transmission and reception module.
4. Performance analysis module.

Cost efficient based on keying module:

The Cost Efficient Based on Keying process involves the creation of dynamic keys. Contrary to other dynamic keying schemes, it does not exchange extra messages to establish keys. A sensor node computes keys based on its residual virtual energy of the sensor. The key is then fed into the crypto module.

Algorithm 1. Compute Dynamic Key

1: ComputeDynamicKey(E_{vc}; ID_{dr})
2: begin
3: j ←tx^{ID}clr
4: if j=1 then
5: K_j ←F(E_{ini},IV)
6: else
7: Kj ←F($k_{(j-1)}$,E_{vc})
8: end if
9: return Kj
10: end

Deciding which nodes to watch and how many depends on the preferred configuration of the CEBKST authentication algorithm, which we designate as the operational mode of the framework. Specifically, we propose two operational modes CEBKST-I and CEBKST-II and they are discussed in the next section. When an event is detected by a source sensor, that node has remained alive or t units of time since the last event (or since the network deployment if this is the first event detected). After detection of the event, the node sends the l-bit length packet toward the sink. In this case, the following is the virtual cost associated with the source node:

$$E_{vc} = l * (e_{tx} + e_{enc}) + t * e_a + E_{synch}$$

In the case where a node receives data from another node, the virtual perceived energy value can be updated by

decrementing the cost associated with the actions performed by the sending node using the following cost equation. Thus, assuming that the receiving node has the initial virtual energy value of the sending node and that the packet is successfully received and decoded associated with a given source sensor, k, the virtual cost of the perceived energy is computed as follows:

$$E_p^k = l * (e_{rx} + e_{dec} + e_{tx} + e_{enc}) + 2 * e_a * t$$

Where in both the equations, the small es refer to the one bit energy costs of the associated parameter. However, E_{synch} in (6) refers to a value to synchronize the source with the watcher-forwarders toward the sink as watcher-forwarder nodes spend more virtual energy due to packet reception and decoding operations, which are not present in source nodes. Hence,

$$E_{synch} = l * (e_{dec} + e_{tx}) + e_a * t$$

Crypto module:

The crypto module in CEBKST employs a simple encoding process, which is essentially the process of permutation of the bits in the packet according to the dynamically created permutation code generated via RC4. The encoding is a simple encryption mechanism adopted for CEBKST. However, CEBKST's flexible architecture allows for adoption of stronger encryption mechanisms.

The resource constraints of WSNs, traditional digital signatures or encryption mechanisms requiring expensive cryptography are not viable. The scheme must be simple, yet effective. Thus, in this section, we introduce a simple encoding operation similar to that used in the encoding operation is essentially the process of permutation of the bits in the packet, according to the dynamically created permutation code via the RC4 encryption mechanism. The key to RC4 is created by the previous module (Cost efficient based on keying module). The purpose of the crypto module is to provide simple confidentiality of the packet header and payload while ensuring the authenticity and integrity of sensed data without incurring transmission overhead of traditional schemes. However, since the key generation and handling process is done in another module, CEBKST's flexible architecture allows for adoption of stronger encryption mechanisms in lieu of encoding. The packets in CEBKST consists of the ID (i-bits), type (t-bits) (assuming each node has a type identifier), and data (d-bits) fields. Each node sends these to its next hop. However, the sensors' ID, type, and the sensed data are transmitted in a pseudorandom fashion according to the result of RC4. More specifically, the RC4 encryption algorithm takes the key and the packet fields (byte-by byte) as inputs and produces the result as a permutation code as depicted in Fig. 3. The concatenation of each 8-bit output becomes the resultant permutation code. As mentioned earlier, the key to the RC4 mechanism is taken from the core virtual energy-based keying module.

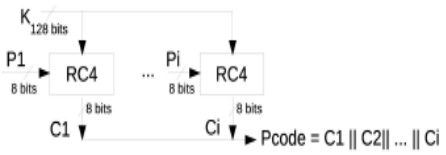

Fig 3. Rc4 Mechanism in CEBKST.

This is responsible for generating the dynamic key according to the residual energy level. The resultant permutation code is used to encode the <ID|type|data> message. Then, an additional copy of the ID is also transmitted in the clear along with the encoded message. The format of the final packet to be transmitted becomes Packet [ID, {ID, type, data} k] where {x}k constitutes encoding x with key k. Thus, instead of the traditional approach of sending the hash value (e.g., message digests and message authentication codes) along with the information to be sent, we use the result of the permutation code value locally. When the next node along the path to the sink receives the packet, it generates the local permutation code to decode the packet.

Packet transmission and reception module:

The Packet transmission and reception module handles the process of sending or receiving of encoded packets along the path to the sink. And also get the acknowledgement from the receiver side to conform the delivery status of the node.

Source Node Algorithm

When an event is detected by a source node, the next step is for the report to be secured. The source node uses the local energy value and an IV (or previous key value if not the first transmission) to construct the next key. As discussed earlier, this dynamic key generation process is primarily handled by the CEBKST module. The source sensor fetches the current value of the energy from the CEBKST module. Then, the key is used as input into the RC4 algorithm inside the crypto module to create a permutation code for encoding the <ID|type|data> message. The encoded message and the clear text ID of the originating node are transmitted to the next hop (forwarding node or sink) using the following format: [ID, {ID, type, data} Pc], where $\{x\}_{Pc}$ constitutes encoding x with permutation code Pc. The local energy value is updated and stored for use with the transmission of the next report.

Forwarder Node Algorithm

Once the forwarding node receives the packet it will first check its watch-list to determine if the packet came from a node it is watching. If the node is not being watched by the current node, the packet is forwarded without modification or authentication. Although this node performed actions on the packet (received and forwarded the packet), its local virtual perceived energy value is not updated. This is done to maintain synchronization with nodes watching it further up the route. If the node is being watched by the current node, the forwarding node checks the associated current virtual energy record (Algorithm 2) stored for the sending node and extracts the energy value to derive the key. It then authenticates the message by decoding the message and comparing the plaintext node ID with the encoded node ID. If the packet is authentic, an updated virtual energy value is stored in the record associated with the sending node. If the packet is not authentic it is discarded. Again, the energy value associated with the current sending node is only updated if this node has performed encoding on the packet.

Algorithm2. Forwarding Node Algorithm with Communication Error Handling

```
: Forwarder(currentNode, WatchedNode, UpstreamNode)
: begin
: i ← currentNode; enc ← 0; WL_i ← WatchList
: k ← WatchedNode; src ← 0; j ← 0
: E_rx_i, ⟨ID_clr, {msg}_K⟩ ← ReceivePacket()
: if ID_clr ∈ WL_i then
:   while (keyFound = 0) and (j <= thresHold) do
:     E^k_p_i ← FetchVirtualEnergy(i, ID_clr, enc, src)
:     K ← ComputeDynamicKey(E^k_p_i, ID_clr)
      Pc ← RC4(K, ID_clr)
      E_dec_i, Msg_ID ← decode(Pc, {msg}_K)
      if ID_clr = Msg_ID then
         keyFound ← true
      else
         j++
         E^k_p_i ← E^k_p_i − E_tx_i − E_enc_i − E_rx_i − E_dec_i − 2 * E_a_i
      end if
    end while
    if keyFound = true then
      if j > 1 then
         reEncode ← true
      else
         if E_b_i > 0 then
            reEncode ← true
         else
            reEncode ← false
         end if
      end if
      if reEncode = true then
         enc ← 1
         E_b_i ← FetchVirtualEnergy(i, ID_clr, enc, src)
         K ← ComputeDynamicKey(E_b_i, ID_clr)
         Pc ← RC4(K, ID_clr)
         E_enc_i, {msg}_Pc ← encode(Pc, msg)
         packet ← ⟨ID_clr, {msg}_Pc⟩
         E_tx_i ← ForwardPacket()
         E_b_i ← E_b_i − E_tx_i − E_enc_i − E_rx_i − E_dec_i − 2 * E_a_i
      else
         ForwardPacket() //Without any modification
      end if
    else
      DropPacket() //Packet not valid
    end if
  else
    ForwardPacket() //Without any modification
  end if
  end
```

Performance analysis module:

In this module we are going to consider the false injection and eavesdropping of messages from an outside malicious node. And also check a routing path is established from the sources in the event region to the sink. We assume that the path is fixed during the delivery of the data and the route setup is secure. So the sensor network is densely populated generate reports for the same event. This module will help to analyze the performance of the nodes.

Simulation Parameters

We use the Georgia Tech Sensor Network Simulator (GTSNetS) [16], which is an event-based object-oriented sensor network simulator with C++, as our simulation platform to perform the analysis of the CEBEKST communication framework. The topology used for the simulation is shown in Fig. 6, while the parameters used in the simulation are summarized in table. Nodes were distributed randomly in the deployment region and on average, the distance between the source nodes and the sink was around 25-35 hops. The Key Search Threshold value was 15. The energy costs for different operations in the table are computed based on the values given in [4]. However, the costs for encoding and decoding operations are computed based on the reported values of the implementation of RC4 [18] on real sensor devices.

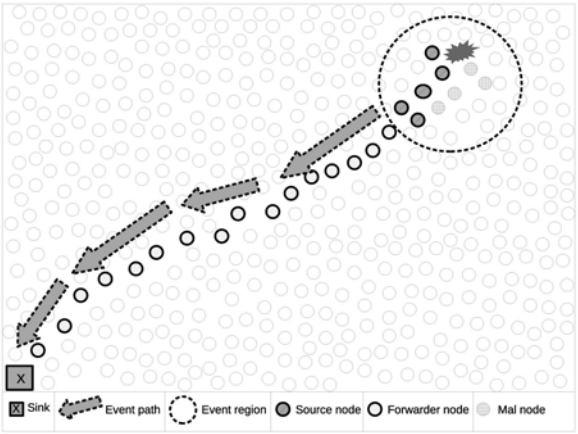

Fig 3. simulation topology with GTSNetS

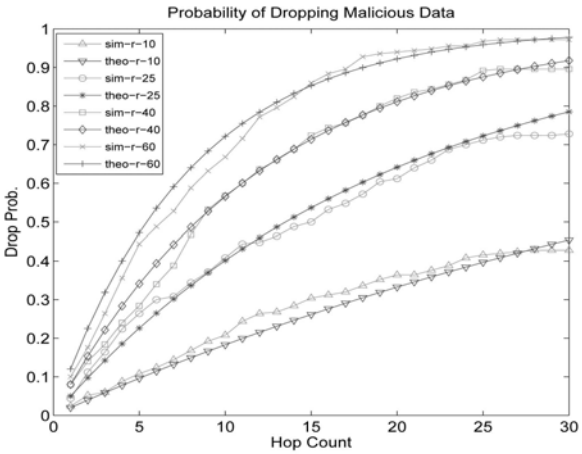

Fig 4. Theoretical and simulation result with varying no.of watching nodes.

TABLE I
General simulation parameters

# of Nodes	500	SensSize	32 bytes
Area	1000x1000 m	RecvInterval	50s
# of Watched	(0..60)	SensRate	30s
Link Rate	250Kbps	SimTime	3000s
Range	75 m	#of Mal Node	(0..10)

III. OPERATIONAL MODES OF CEBKST

The CEBKST protocol provides three security services:

Authentication, integrity, and non-repudiation the fundamental notion behind providing these services is the watching mechanism described before. The watching mechanism requires nodes to store one or more records (i.e., current energy level, bridge energy values, and Node-Id) to be able to compute the dynamic keys used by the source sensor nodes, to decode packets, and to catch erroneous packets either due to communication problems or potential attacks. However, there are costs (communication, computation, and storage) associated with providing these

services. In reality, applications may have different security requirements. For instance, the security need of a military WSN application (e.g., surveying a portion of a combat zone) may be higher than that of a civilian application (e.g., collecting temperature data from a national park).

The CEBKST framework also considers this need for flexibility and thus, supports two operational modes: CEBKST-I and CEBKST-II. The operational mode of CEBKST determines the number of nodes a particular sensor node must watch. Depending on the vigilance required inside the network, either of the operational modes can be configured for WSN applications. The details of both operational modes are given below.

***CEBKST-I*:**

In the CEBKST-I operational mode, all nodes watch their neighbors; whenever a packet is received from a neighbor sensor node, it is decoded and its authenticity and integrity are verified. Only legitimate packets are forwarded toward the sink. In this mode, we assume there exists a short window of time at initial deployment that an adversary is not able to compromise the network, because it takes time for an attacker to capture a node or get keys. During this period, route initialization information may be used by each node to decide which node to watch and a record r is stored for each of its one-hop neighbors in its watch-list. To obtain a neighbor's initial energy value, a network-wise master key can be used to transmit this value during this period similar to the shared-key discovery phase of other dynamic key management schemes.

Alternatively, sensors can be preloaded with the initial energy value. When an event occurs and a report is generated, it is encoded as a function of a dynamic key based on the energy of the originating node and transmitted. When the packet arrives at the next-hop node, the forwarding node extracts the key of the sending node (this could be the originating node or another forwarding node) from its record. (The perceived energy value associated with the sending node and decodes the packet.) After the packet is decoded successfully, the plaintext ID is compared with the decoded ID. In this process, if the forwarding node is not able to extract the key successfully, it will decrement the predefined virtual energy value from the current perceived energy and tries another key before classifying the packet as malicious (because packet drops may have occurred due to communication errors). This process is repeated several times; however, the total number of trials that are needed to classify a packet as malicious is actually governed by the value of Key Search Threshold.

If the packet is authentic, and this hop is not the final hop, the packet is re encoded by the forwarding node with its own key derived from its current virtual bridge energy level. If the packet is illegitimate, the packet is discarded. This process continues until the packet reaches the sink. Accordingly, illegitimate traffic is filtered before it enters the network.

CEBKST-II:

In the CEBKST-II operational mode, nodes in the network are configured to only watch some of the nodes in the network. Each node randomly picks r nodes to monitor and stores the corresponding state before deployment. As a packet leaves the source node (originating node or forwarding node) it passes through node(s) that watch it probabilistically. Thus, CEBKST-II is a statistical filtering approach like SEF and DEF. If the current node is not watching the node that generated the packet, the packet is forwarded. If the node that generated the packet is being watched by the current node, the packet is decoded and the plaintext ID is compared with the decoded ID. Similar to CEBKST-I, if the watcher-forwarder node cannot find the key successfully, it will try as many keys as the value of virtual Key Search- Threshold before actually classifying the packet as malicious. If the packet is authentic, and this hop is not the final Destination, the original packet is forwarded unless the node is currently bridging the network. In the bridging case, the original packet is re encoded with the virtual bridge energy and forwarded. Since this node is bridging the network, both virtual and perceived energy values are decremented accordingly.

If the packet is illegitimate, which is classified as such after exhausting all the virtual perceived energy values within the virtual Key Search Threshold window, the packet is discarded. This process continues until the packet reaches the sink.

IV. CONCLUSION

Communication is very costly for wireless sensor networks (WSNs) and for certain WSN applications. Independent of the goal of saving energy, it may be very important to minimize the exchange of messages (e.g., military scenarios). To address these concerns, we presented a secure communication framework for WSNs called cost efficient based on Keying. In comparison with other key management schemes, CEBKST has the following benefits: 1) it does not exchange control messages for key renewals and is therefore able to save more energy and is less chatty, 2) it uses one key per message so successive packets of the stream use different keys—making CEBKST more resilient to certain attacks (e.g., replay attacks, brute-force attacks, and masquerade attacks), and 3) it unbundled key generation from security services, providing a flexible modular architecture that allows for an easy adoption of different key-based encryption or hashing schemes.

FUTUERE ENHANCEMNET

The project has covered almost all the requirements. Further requirements and improvements can easily be done since the coding is mainly structured or modular in nature. Improvements can be appended by changing the existing modules or adding new modules. Our future work will address insider threats and dynamic paths.

REFERENCES

1. S. Uluagac, R. Beyah, and J. Copeland, "Secure Source-Based Time Synchronization (SOBAS) for Wireless Sensor Networks," technical report, Comm. Systems Center, School of Electrical and Computer Eng., Georgia Inst. of Technology, http://users.ece. gatech.edu/selcuk/sobas-csc-techreport.pdf, 2009.
2. R. Venugopalan et al., "Encryption Overhead in Embedded Systems and Sensor Network Nodes: Modeling and Analysis," Proc. ACM

Int'l Conf. Compilers, Architecture, and Synthesis for Embedded Systems (CASES '03), pp. 188-197, 2003.
3. C. Kraub, M. Schneider, K. Bayarou, and C. Eckert, "STEF: A Secure Ticket-Based En-Route Filtering Scheme for Wireless Sensor Networks," Proc. Second Int'l Conf. Availability, Reliability and Security (ARES '07), pp. 310-317, Apr. 2007.
4. C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," Proc. ACM MobiCom, pp. 56-67, Aug. 2002.
5. K. Akkaya and M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks," Ad Hoc Networks, vol. 3, pp. 325-349,May 2005.
6. Georgia Tech Sensor Network Simulator (GTSNetS), http:// www.ece.gatech.edu/research/labs/MANIACS/GTNetS, 2007.
7. M. Passing and F. Dressler, "Experimental Performance Evaluation of Cryptographic Algorithms on Sensor Nodes," Proc. IEEE Int'l Conf. Mobile Ad-hoc and Sensor Systems, pp. 882-887, Oct. 2006.

AUTHORS

- **Mr. Narahari A** Graduates in Computer Science and Engineering recieved from JNT University of Hyderabad and Master of Technology in software engineering in kakatiya university.
- **Smt. M. Preethi** Assistant Professor in Computer Science and Engineering Department in kakatiya institute of technology and science.

.