Improved Leap Protocol with Experimental Results and Conclusion

[*]Sapna A. Aekre, [**]Prof. R. K. Krishna

[*]Department of Computer Technology, Rajiv Gandhi College of Engineering & research, Chandrapur (MH)
[**] Department of Electronics, Rajiv Gandhi College of Engineering & research, Chandrapur (MH)

***Abstract*-** Leap protocol offers many security benefits to WSNs. However, with much research it became apparent that LEAP only employs one base station and always assumes that it is trustworthy. It does not consist of defence against hacked or compromised base stations. In this paper, intensive research was undertaken on LEAP protocols, finding out its security drawbacks and limitations. One of the biggest concerns of WSNs is that they are very defenceless to security threats. One of the biggest concerns of WSNs is that they are very defenceless to security threats. Due to the fact that these networks are susceptible to hackers; it is possible for one to enter and render a network. A solution has been proposed in order to overcome the security issues faced in implementing this protocol whilst employing more than one base station. The performance of the proposed solution has been evaluated and simulated to provide a better network performance.

***Index Terms*-** Network Protocols, Wireless Sensor Network (WSN), LEAP protocol, Security, compromised nodes, In-network Processing, Key Erasure, Key Management, Pairwise Key, and Sensor Networks.

I. INTRODUCTION

In literature, the majority of the key management protocols usually focus on the aspect that only a singular base station or sink node is used in a WSN and these protocols assume that it is trustworthy. For some systems, however, several sink nodes are used [8]. In these systems, two important things must be considered: cost and security. In the leap protocol, several efforts are made through the use of the keying mechanisms to ensure that a compromised node is revoked or at least prevent it from slowing the network operations. A base station, on the other hand, will be treated the same as any compromised node and the idea is to apply the same mechanisms used to overcome a compromised node to also prevent a hacked base station node. With a lot of excessive research, the literature usually covers WSN functionalities in terms of one base station participating in one system. It is important to remember that with an increase in a sensor network there's an increase in the distance separating the base station and its related sensor nodes and the increase in the distance may alter the following:

- With a long distance for packets to propagate through, they may get lost on the way resulting in network performance degradation.

- Data transmissions between sensor nodes and a single base station in a large network require high energy consumptions giving the need to reduce the lifetime of nodes.

- For the nodes that are situated nearby a base station, their energy is worn out rapidly, which in turn shortens the network life time very drastically.

To overcome these problems, a network employing several base stations shows potential in bettering the performance. However, there is of course the tradeoff between performance and cost. By deploying more than one sink node in a network may be costly, but the distance between the sink nodes and its associated sensor nodes will be reduced providing more successful paths for data transmission as well as eliminating the disadvantage of the high energy consumptions otherwise faced. For this research, a WSN with several base stations will be considered. Under the circumstances that a base station and a sensor node are compromised, an evaluation of the network performance will be analyzed.

Wireless sensor networks provide the advantage of using a large number of nodes (from a hundred up to thousands of nodes) communicating with each other inexpensively. One or more base stations process all of the network functions. Should there be a need to increase the number of sink nodes, one has to consider enhancement in expenses. The LEAP protocol offers much security to a system with the establishment of the four keys, mentioned previously. The protocol consists of key revocation and refreshing mechanisms in the attempts to successfully avoid or deal with compromised sensor nodes. The methods used in detecting the isolated compromised nodes are done through µTESLA and one-way key chain hash authentication functions [9]. However, this protocol lacks in security against a base station, should it be compromised, and network robustness. These are significant aspects to consider because if a sink node is compromised, it could severely affect the entire network or system as all the network functions are dealt by these nodes. The flexibility feature of a LEAP protocol is advantageous over many other security protocols used, but improvements in robustness are needed. Therefore, to improve the LEAP protocol, a solution is proposed to overcome the limitations faced for possible attacks on the base stations itself and thereby adding more robustness to the network system in terms of recovering from a compromised base station as well as a compromised sensor node. In theory, the majority of research papers, consider the presumption of a reliable base station and only take measures for compromised nodes. In isolated locations, it is relevant to be vigilant in case a base station is compromised. Security against higher levels of attacks against a base station, which usually occur from sources with higher computational power, is a necessity

To evaluate the proposed solution, an algorithm has been developed to simulate a sensor network using the MATLAB program. The whole idea was to implement a system whereby multiple base stations have been employed for the soul purposes of improving the data transmissions amongst nodes and to come up with a solution for a base station, should it be compromised.

The LEAP protocol was implemented and simulated using one base station and fifty sensor nodes situated randomly. Initially, an individual key was generated for each node from a randomly generated master key. Then a cluster key was generated by each node and published to their neighboring nodes using the pair-wise keys. Finally, the global key was generated in order to enable public broadcasts. an ideal case for the LEAP protocol. It simply represents a base station surrounded by fifty sensor nodes. In this scenario, none of the nodes are compromised. However, even though it is an ideal case, we still face the problem of data loss. For arguments sake, let's assume that node z wants to communicate with the base station. Having a singular base station means that no matter how far the distance, the sensor node and the base station will commune with each other. The longer the distance, however, the more nodes they have to transmit through, the more bandwidth will be used and the higher the possibility of loss of data.

II. Network Initialization

Time synchronization is a critical issue in distributed WSN. The TDMA-based MAC component imposes a strict synchronization between clusters, while the CSMA component requires robustness against clock drift between nodes inside a cluster.

Token Passing Procedure

Improved LEAP protocol implements a token passing procedure that:

• ensures synchronization between nodes and clusters;

• allows initializing and self configuring to the optimal working point;

• allows for the addition of new nodes.

A token is a particular message that carries the information on the duration of a TDMA-slot and a TDMA-cycle, the transmitting and receiving schedule of a TDMA-cycle, a synchronization message carrying the current execution state of the TDMA-cycle.

The Controller has all the information to calculate the optimal set of parameters, consequently, it is able to generate a token before the network starts operating. The network initialization algorithm works as follows:

1. When the network starts, all nodes are awake and listening. Nodes remain in this state and cannot transmit before receiving the token.

2. The Controller multi-casts the token to all nodes of one of the connected clusters. In general this is the first in the scheduling table. In the reference example (Fig. 3.1), assume the selected cluster is the cluster 4.

3. Nodes of the selected cluster read the information on scheduling and duration of TDMA-slot and TDMA-cycle. Moreover, each node acquires the information about the global time and launches periodic timers for CSMA and TDMA slots. In the meantime, a random back-off timer starts for each node before sending an acknowledgement.

4. The first node that expires the back-off time sends the acknowledgement to the Controller and becomes the token forwarder. Then, all nodes in the cluster go to sleep.

5. At the beginning of the second TDMA-slot the token forwarder wakes up and immediately multi-casts the token to all nodes in the next cluster (i.e. cluster 2).

6. With the same random acknowledgement-based scheme, a node is elected token forwarder for nodes in the following cluster (i.e. cluster 1).

7. After the first branch of the routing tree is explored, the Controller sends a token to cluster 5, the new branch is explored, and so on.

Information about routing and TDMA-slot duration needs also to be updated during the network operation. Hence, the Controller periodically performs a token refreshing procedure. It is a critical phase because the Controller needs to ensure that all nodes in the network receive the new token. First, all nodes should be in listening state and, moreover, when the token is forwarded along the network, the scheduled packet transmission has to be interrupted, to avoid collisions. If the frequency of token refreshing is high, the response of the protocol to variation in the surrounding conditions is faster and the adaptability increases. On the other side, the procedure is costly in terms of energy consumption. In our implementation, we do not consider strict requirements on the protocol adaptability, while the minimum energy consumption is fundamental. We think that a suitable choice of the refreshing period is 20 TDMA-cycles. Nodes are informed by the Controller about the refreshing period through the first token passing. We can also suppose that the Controller can modify this value and update it during each token refreshing, according to the network behavior.

III. Routing Implementation

ActiveMessageP: This is the highest layer in the stack, responsible for filling in details in the packet header and providing information about the packet to the application level.

• UniqueSend: This layer generates a unique Data Sequence Number (DSN) byte for the packet header. This byte is incremented once per outgoing packet, starting with a pseudo-randomly generated number. A receiver can detect duplicate packets by comparing the source and DSN byte of a received packet with previous packets. DSN is defined in the 802.15.4 specification [31].

• PacketLink : This layer provides automatic retransmission functionality and is responsible for retrying a packet transmission if no acknowledgement was heard from the receiver.

• CC2420AckLplP / CC2420NoAckLplP: These layers provide asynchronous low power listening implementations. AckLplP implementation supports acknowledgement gaps during the preamble. The NoAckLplP implementation does not support it.

• UniqueReceive: This layer maintains a history of the source address and DSN byte of the past few packets it has received, and helps filter out duplicate received packets.

• TinyosNetworkC: This layer allows the TinyOS 2.x radio stack to interoperate with other non-TinyOS networks.

• CsmaC: This layer is responsible for defining 802.15.4 FCF byte information in the outbound packet, providing default back-off times when the radio detects a channel in use, and defining the power-up/power down procedure for the radio.

• TransmitP/ReceiveP: These layers are responsible for interacting directly with the radio through the SPI bus, interrupts, and GPIO lines. The main part of SERAN code belongs to the Application Layer of the stack, but some parts are mixed with the other layers. Awake and sleeping states are governed by the SplitControl interface. It is used for switching between the on and off power states of the component providing it, with the functions start() and stop(). The implementation of the SplitControl interface is defined in the CC2420CsmaP component.

The TDMA and CSMA slotted structure is realized with the instances of the TinyOS 2.x TimerMilliC component, wired to the interface Timer <Tmilli > that provides a set of commands to handle periodic events. It ensures a precision in order of milliseconds.

Node configuration consists of two parts:
($ns node-config)

- The first part deals with node configuration
- The second part creates nodes of the specified type.
- The multicast classifier classifies packets according to both source and destination (group) addresses.

Node Configuration Methods:

- **Control functions**
  $node entry returns the entry point for a node.
  $node reset will reset all agents at the node.
- **Address and Port number management**
  $node agent(port) returns the handle of the agent at the specified port.
  alloc-port returns the next available port number

Connector

A connector will receive a packet perform some function and deliver the packet to its neighbor or drop the packet.

Queues Locations where packets may be held (or dropped).

Queue Class

```
class Queue : public Connector {
public:
virtual void enque(Packet*) = 0;
virtual Packet* deque() = 0;
void recv(Packet*, Handler*);
void resume();
int blocked();
void unblock();
void block();
protected:
Queue();
int command(int argc, const char*const* argv);
int qlim_; /* maximum allowed pkts in queue */
int blocked_;
int unblock_on_resume_; /* unblock q on idle? */
QueueHandler qh_;
}
```

Queue Blocking

```
class QueueHandler : public Handler {
public:
inline QueueHandler(Queue& q) : queue_(q) {}
void handle(Event*); /* calls queue_.resume() */
private:
Queue& queue_;
};
void QueueHandler::handle(Event*)
{
queue_.resume();
}
Queue::Queue() : drop_(0), blocked_(0), qh_(*this)
{
Tcl& tcl = Tcl::instance();
bind("limit_", &qlim_);
}
void Queue::recv(Packet* p, Handler*)
        {
        enque(p); void Queue::resume()
        {
        Packet* p = deque();
        if (p != 0)
        target_->recv(p, &qh_);
        else
       {
        if (unblock_on_resume_)
        blocked_ = 0;
        else
        blocked_ = 1;
        }
        }
        if (!blocked_)
 {
        /* We're not block. Get a packet and send it on. We perform an extra check because the queue might drop the packet even if it was previously empty! */
        p = deque();
        if (p != 0) {
        blocked_ = 1;
        target_->recv(p, &qh_);
        }
        }
        }
```

```
void Queue::resume()
{
Packet* p = deque();
if (p != 0)
target_->recv(p, &qh_);
else {
if (unblock_on_resume_)
blocked_ = 0;
else
blocked_ = 1;
}
}
```

Scheduling algorithm(Round Robin)

- provide absolute and relative(proportional) loss and delay differentiation independently.
- at each node for *classes of traffic* performs scheduling and buffer management in a single pass.
- Unit of time used by scheduler is seconds.
- The calendar queue scheduler uses a data structure analogous to a one-year desk calendar
- events on the same month/day of multiple years can be recorded in one day.

Delays

Represent the time required for a packet to traverse a link.

- class LinkDelay : public Connector {
- public:
- LinkDelay();
- void recv(Packet* p, Handler*);
- void send(Packet* p, Handler*);
- void handle(Event* e);
- double delay(); /* *line latency on this link */*
- double bandwidth(); /* *bandwidth on this link */*
- inline double txtime(Packet* p) { /* *time to send pkt p on this link */*
- hdr_cmn* hdr = (hdr_cmn*) p->access(off_cmn_);
- return (hdr->size() * 8. / bandwidth_);
- }
- protected:
- double bandwidth_; /* *bandwidth of underlying link (bits/sec) */*
- double delay_; /* *line latency */*
- int dynamic_; /* *indicates whether or not link is ~ */*
- Event inTransit_;
- PacketQueue* itq_; /* *internal packet queue for dynamic links */*
- Packet* nextPacket_; /* *to be delivered for a dynamic link. */*
- Event intr_;
- };
- The recv() method overrides base class Connector
- void LinkDelay::recv(Packet* p, Handler* h)
- {
- double txt = txtime(p);
- Scheduler& s = Scheduler::instance();
- if (dynamic_) {
- Event* e = (Event*)p;
- e->time_ = s.clock() + txt + delay_;
- itq_->enque(p);
- schedule_next();
- } else { class Event {
- public:
- Event* next_; /* *event list */*
- Handler* handler_; /* *handler to call when event ready */*
- double time_; /* *time at which event is ready */*
- int uid_; /* *unique ID */*
- Event() : time_(0), uid_(0) {}
- };
- /* *The base class for all event handlers. When an event's scheduled*
- s.schedule(target_, p, txt + delay_);
- }
- s.schedule(h, &intr_, txt);
- }

Routing Module

Routing implementation c*onsists of 3 functionalities:*

- *Routing agent exchanges routing packet with neighbors*
- *Route logic uses the information gathered by routing agents* to perform the actual route computation,
- *Classifiers sit inside a Node. They use the computed routing table to perform packet forwarding.*

Simulation

When a simulation object is created, the initialization procedure performs the following operations:

- initialize the packet format (calls create_packetformat)
- create a scheduler (defaults to a calendar scheduler)
- create a "null agent" (a discard sink used in various places)

The Simulator class provides methods used to set up the simulation.

- Simulator instproc now ;# *return scheduler's notion of current time*
- Simulator instproc at args ;# *schedule execution of code at specified time*
- Simulator instproc cancel args ;# *cancel event*
- Simulator instproc run args ;# *start scheduler*
- Simulator instproc halt ;# *stop (pause) the scheduler*
- Simulator instproc flush-trace ;# *flush all trace object write buffers*
- Simulator instproc create-trace type files src dst ;# *create trace object*
- Simulator instproc create_packetformat ;# *set up the simulator's packet format*

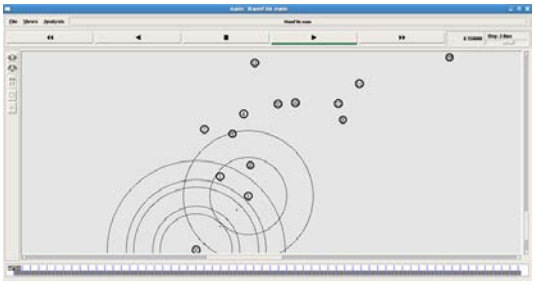
System Requirements

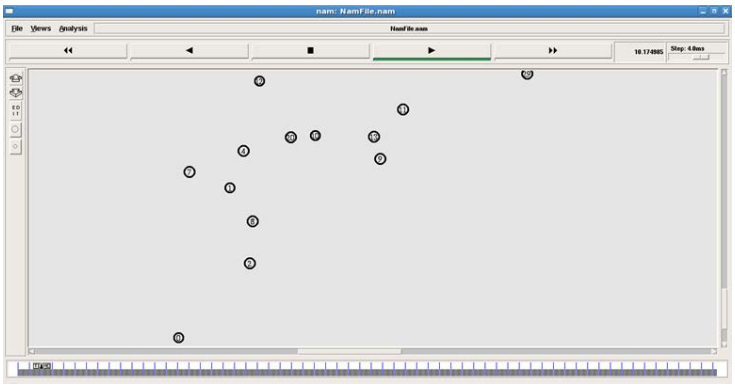
- More base stations.(reduces packet loss)
- base station key(bk) to each node.(authentication/ security)

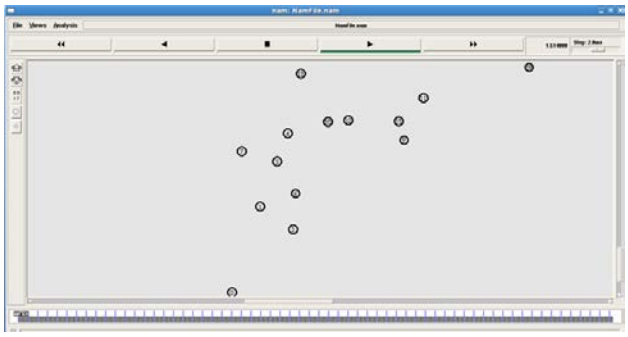
Simulation Parameter:

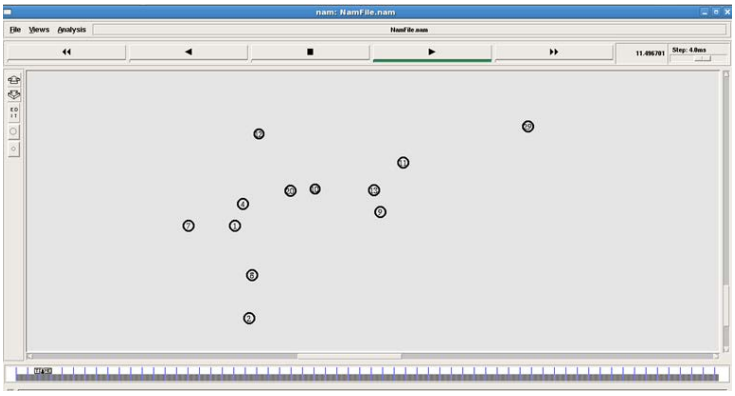
Routing Protocol	AODV
Simulation Area	500*500
Number of Mobile Node	5-50
Threshold Distance	100-250
Transmission Range	250m

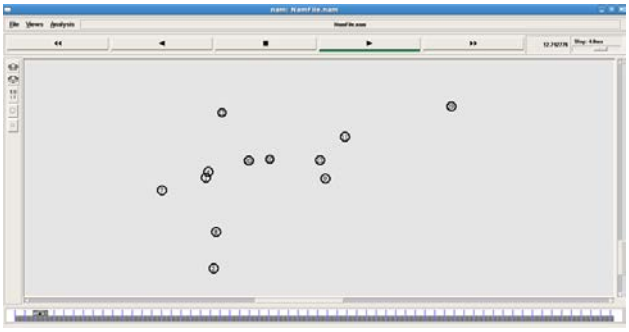
Simulation Scenario

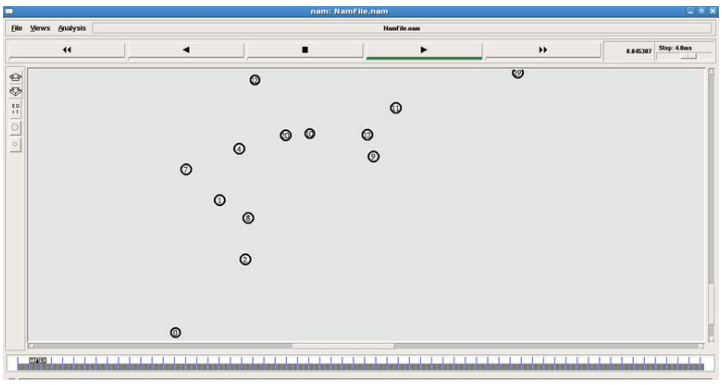


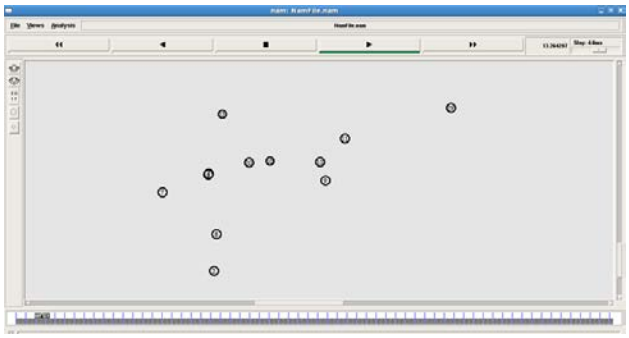












Experimental Results

To evaluate the proposed solution, an algorithm has been developed to simulate a sensor network using the MATLAB program. The whole idea was to implement a system whereby multiple base stations have been employed for the soul purposes of improving the data transmissions amongst nodes and to come up with a solution for a base station, should it be compromised.

The LEAP protocol was implemented and simulated using one base station and fifty sensor nodes situated randomly. Initially, an individual key was generated for each node from a randomly generated master key. Then a cluster key was generated by each node and published to their neighboring nodes using the pair-wise keys. Finally, the global key was generated in order to enable public broadcasts. an ideal case for the LEAP protocol. It simply represents a base station surrounded by fifty sensor nodes. In this scenario, none of the nodes are compromised. However, even though it is an ideal case, we still face the problem of data loss. For arguments sake, let's assume that node z wants to communicate with the base station. Having a singular base station means that no matter how far the distance, the sensor node and the base station will commune with each other. The longer the distance, however, the more nodes they have to transmit through, the more bandwidth will be used and the higher the possibility of loss of data.

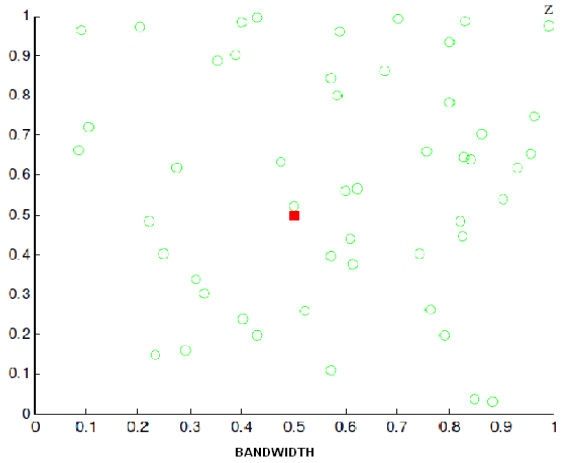


Figure LEAP protocol: Ideal scenario

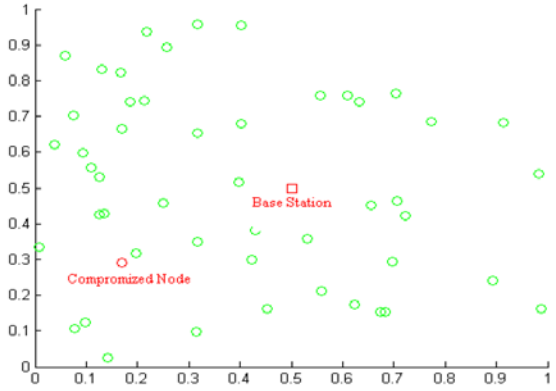


Figure shows the scenario of a compromised node. The node that is labelled symbolizes a node that has been hacked. As mentioned throughout this paper, the LEAP protocol is very efficient when it comes to dealing with compromised nodes. With its key refreshing and revocation schemes, if a node is affected, these mechanisms prove advantageous. With the many keys assigned to all the sensor nodes with its periodic updates, if one of the nodes is unable to decrypt an updated key, the compromised node will not be able to further participate in the data transmission which will then inform the surrounding nodes and eventually the base station that this node is no longer wanted. The compromised node will be removed. Figure. LEAP protocol: A compromised Sensor Node

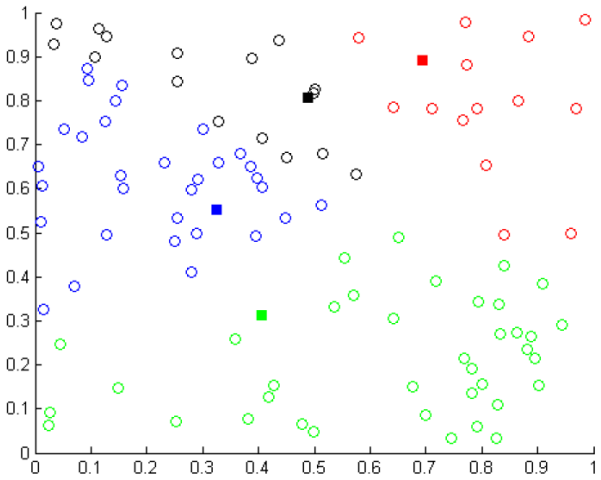


Figure 5. Improved LEAP protocol: Ideal Scenario

In order to simulate and test the performance of the proposed improved LEAP protocol, a WSNof hundred sensor nodes situated randomly, and four base stations also situated randomly has been generated.

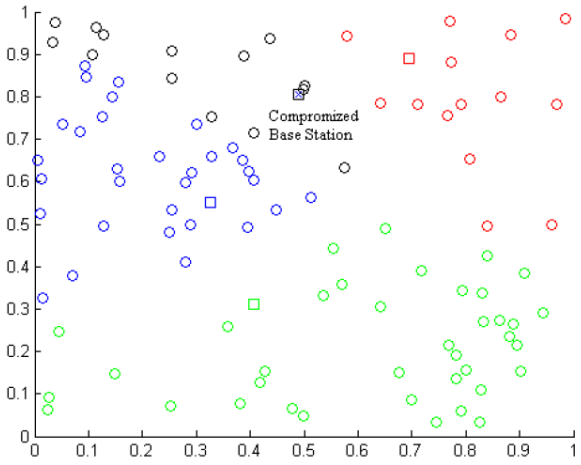


Figure 6. Improved LEAP protocol: A compromised base station

Figure 5 shows the improved LEAP protocol in an ideal scenario, whereby multiple base stations are supported. Depending on the distance between the nodes and the base stations, each sensor node was assigned to its closest base station. The four different colors (red, blue, green and black) are used in order to distinguish between the base stations and its corresponding sensor nodes. This diagram illustrates an ideal scenario whereby none of the nodes or the base stations are compromised. The use of four base stations provides an advantage over the existing LEAP protocol. The idea that more than one base station has been used, the nodes will not need to transmit to a base station that is extremely distant from it, which

means that it minimizes the problem most networks sometimes face regarding lost data during transmission. Figure 5. Improved LEAP protocol: Ideal Scenario Figure 6 demonstrates the scenario whereby a base station is compromised. A hacked base station can be detected and revoked by the other base stations using the new generated key known as base station key (Kb). If a base station is hacked, it will not be aware of the updated session key, which is also updated periodically, and continue to use its old key. In doing so, the base station will not be involved in the data transmission, and the other remaining base stations will identify that this base station is compromised. The authenticated base stations will send the administrator a message indicating that one of the base stations is hacked. It is then up to the administrator to remove it from the system or replace it with another one. However, there is always a case whereby the opponent base station will act like an authenticated node and accuse one of the other validated base stations of being hacked. The administrator will consider any of the base stations to be hacked if at least more than one of the remaining three base stations declares otherwise.Figure 6. Improved LEAP protocol: A compromised base station In this solution, the LEAP protocol was improved in terms of using multiple base stations for the purpose of minimizing loss of data transmissions, and also the proposed solution was able to detect a compromised base station. In using multiple base stations, the performance of the system is improved but the cost of implementation is increased.

Table 1. Comparison between LEAP and Improved LEAP protocols

	LEAP	Improved LEAP
Detects and removes Compromised Sensor Nodes	Yes 50	Yes 100
Detects and removes compromised base stations	No 01	Yes 04
Data Loss	High 10	Minimal 02
Cost	Low	High
Bandwidth use	High	Low
Transmission Delay time	High 0.9	Low 0.1
Energy consumption	High	Low
Node lifetime	Low	High

Performance Analysis

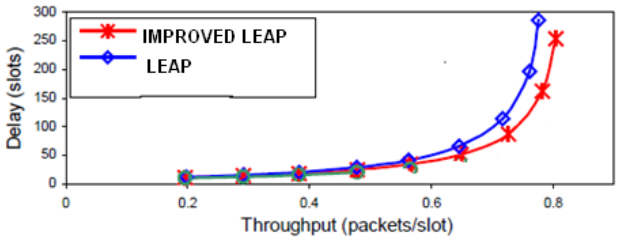


Fig. 1. Average packet delay versus throughput, where we plot for packet loss rate lower than 15%

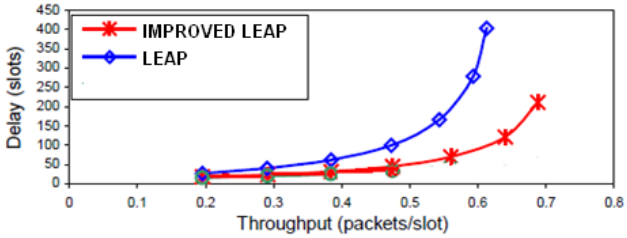


Fig. 2. Average packet delay versus throughput, where we plot for packet loss rate lower than 20%

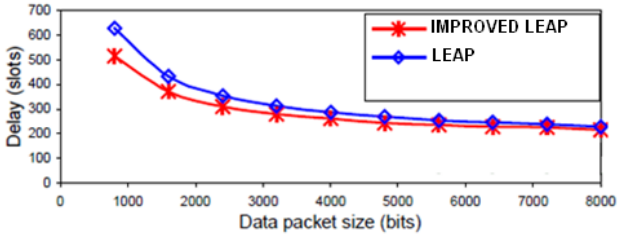
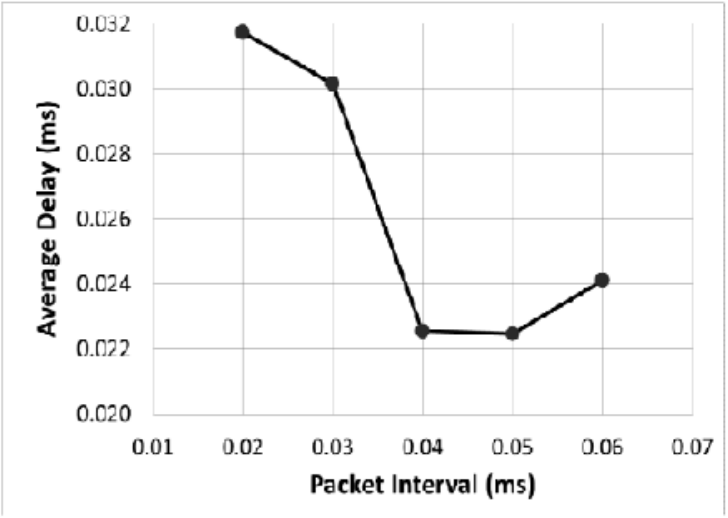
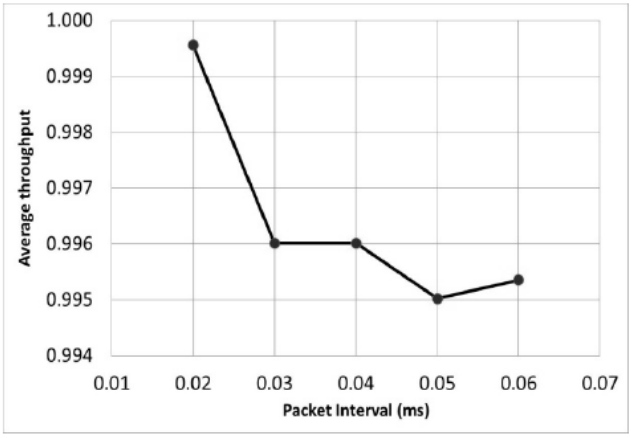


Fig. 3. Average packet delay versus data packet size, where we plot for any packet loss rate

Performance Metrics



Packet Intervals versus Average Delay



Packet Intervals versus Average Throughput

IV. Conclusion

To evaluate the proposed solution, an algorithm has been developed to simulate a sensor network using the MATLAB program. The whole idea was to implement a system whereby multiple base stations have been employed for the soul purposes of improving the data transmissions amongst nodes and to come up with a solution for a base station, should it be compromised.
The LEAP protocol was implemented and simulated using one base station and fifty sensor nodes situated randomly. Initially, an individual key was generated for each node from a randomly generated master key. Then a cluster key was generated by each node and published to their neighboring nodes using the pair-wise keys. Finally, the global key was generated in order to enable public broadcasts. It simply represents a base station surrounded by fifty sensor nodes. In this scenario, none of the nodes are compromised. However, even though it is an ideal case, we still face the problem of data loss. For arguments sake, let's assume that node z wants to communicate with the base station. Having a singular base station means that no matter how far the distance, the sensor node and the base station will commune with each other. The longer the distance, however, the more nodes they have to transmit through, the more bandwidth will be used and the higher the possibility of loss of data.

References

[1] A. Bonivento, C. Fischione, A. Sangiovanni-Vincentelli, F. Graziosi,F. Santucci: SERAN: A Semi Random Protocol Solution for Clustered Wireless Sensor Networks, MASS '05, November 2005.

[2] A. Bonivento, C. Fischione, F. Pianegiani, A. Sangiovanni-Vincentelli:System Level Design for Clustered Wireless Sensor Networks, IEEE Transactions on Industrial Informatics, Vol. 3, No. 3, August 2007

[3] A. Bonivento, C. Fischione, A. Sangiovanni-Vincentelli: Randomized Protocol Stack for Ubiquitous Networks in Indoor Environment. Proceedings of Consumer Communication and Network Conference (CCNC),Las Vegas, January 2006.

[4] A. Bonivento: Platform Based Design for Wireless Sensor Networks. PhD Thesis, UC Berkeley, 2007.

[5] P.G. Park: Protocol Design of Sensor Networks for Wireless Automation. MSc Thesis, KTH Stockholm, 2007.

[6] P.G. Park, C. Fischione, A. Bonivento, K.H. Johansson, A. Sangiovanni-Vincentelli: Breath: a Self-Adapting Protocol for Wireless Sensor Networks. Accepted paper at IEEE SECON 2008, San Francisco US, June2008.

[7] D. Culler, D. Estrin, M. Srivastava: Overview of Sensor Networks, IEEEComputer Society, August 2004.

[8] L. Pomante: Wireless Sensor Networks, Seminar in Wireless Communications - University of L'Aquila, March 2007.

[9] J.N. Al-Karari, A.E. Kamal: Routing Techniques in Wireless Sensor Networks: A Survey, IEEE Wireless Communications, December 2004.

[10] W. Heinzelman et al.: Energy-efficient Communication Protocol for Wireless Microsensor Networks, Proc. 33rd Hawaii Int'l Conf.Sys.Sci.,January 2000.

[11] nesC: A Programming Language for Deeply Networked Systems, UC Berkeley WEBS Project, December 2004.

[12] C.Y. Chong: Sensor Networks: Evolution, Opportunities, and Challenges, Proceedings of IEEE Vol 91, No 8, August 2003.

[13] I.F. Akyildiz , W. Su , Y. Sankarasubramaniam , E. Cayirci: Wireless sensor networks: a survey, Computer Networks: The International Journal of Computer and Telecommunications Networking, v.38 n.4, p.393-422, 15 March 2002

[14] A. Sangiovanni-Vincentelli, M. Sgroi, A.Wolisz and J. M. Rabaey: A service-based universal application interface for ad-hoc wireless sensor networks, Whitepaper, U.C.Berkeley, 2004.

[15] W. Ye and J. Heidemann: Medium access control in wireless sensor networks, Norwell, MA, USA: Kluwer Academic Publishers, pp. 73 – 91, 2004.

[16] R. Rom, M. Sidi: Multiple Access Protocols - Performance and analysis Springer-Verlag, New York, 1990.

[17] J. Heidemann W. Ye and D. Estrin: Medium access control with coordinated adaptive sleeping for wireless sensor networks, IEEE/ACM Transactions on Networking, 12, n. 3 pp. 493-506, 2004.

[18] T.V. Dam and K. Langendoen: An adaptive energy-efficient MAC protocol for Wireless Sensor Networks, in SenSys '03. New York, NY, USA:ACM Press, pp. 171–180, 2003.

[19] J. Hill, J. Polastre and D. Culler: Versatile low power media access for wireless sensor networks, Proceedings of SenSys 2003.

[20] I. Rhee, A. Warrier, M. Aia and J. Min: ZMAC: a Hybrid MAC for Wireless Sensor Networks, New York, NY, USA: ACM Press, 2005

Authors

First Author – Sapna A. Aekre, Department of Computer Technology, Rajiv Gandhi College of Engineering & research, Chandrapur (MH), sapna.aekre@gmail.com
Second Author – Prof. R. K. Krishna, Department of Electronics, Rajiv Gandhi College of Engineering & research, Chandrapur (MH), rkkrishna40@rediffmail.com