Paper on Implementation of Improved LEAP Protocol

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

*Department of Computer Technology, Rajiv Gandhi College of Engineering & research, Chandrapur (MH)
sapna.aekre@gmail.com
** Department of Electronics, Rajiv Gandhi College of Engineering & research, Chandrapur (MH)
rkkrishna40@rediffmail.com

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

Many sensor systems are deployed in unattended and often adversarial environments such as a battlefield. Hence, security mechanisms that provide Confidentiality and authentication are critical for the operation of many sensor applications. Providing security is particularly challenging in sensor networks due to the resource limitations of sensor
nodes. A fundamental issue that must be addressed when using key management protocols based on symmetric shared keys is the mechanisms used for establishing the shared keys in the first place. The constrained energy budgets and the limited computational and communication capacities of sensor nodes make protocols such as TLS [13] and

Kerberos [23] proposed for wired networks impractical for use in largescale sensor networks. At present, the most practical approach for bootstrapping secret keys in sensor networks is to use pre-deployed keying in which keys are loaded into sensor nodes before they are deployed. Several solutions based on pre-deployed keying have been proposed in the literature including approaches based on the use of a global key shared by all nodes [5, 8], approaches in which every node shares a unique key with the base station [36], and approaches based on random key sharing [9, 15, 16, 28]. An important design consideration for security protocols based on symmetric keys is the degree of key sharing between the nodes in the system. At one extreme, we can have network-wide keys that are used for encrypting data and for authentication. This key sharing approach has the lowest storage costs and is very energy-efficient since no communication is required between nodes for establishing additional keys. However, it has the obvious security disadvantage that the compromise of a single node will reveal the global key. At the other extreme, we can have a key sharing approach in which all secure communication is based on keys that are shared pairwise between two nodes. From the security point of view, this approach is ideal since the compromise of a node does not reveal any keys that are used by the other nodes in the network. However, under this approach, each node will need a unique key for every other node that it communicates with. Moreover, in many sensor networks, the immediate neighbors of a sensor node cannot be predicted in advance; consequently, these pairwise shared keys will need to be established after the network is deployed. saving energy is one of the most important research challenges in WSNs. Sensor nodes are powered by external atteries and often it is hard to replace them after consuming, while most of the applications require long lifetime in the order of years. Hence, the choice of an objective function in terms of energy consumption is clearly justified. On the other hand, a mere optimization for energy can lead the network to work without fulfilling its tasks. Energy efficiency has to be well-balanced with the assigned requirements and network purposes. Basically, the application requires two constraints:

1. Error rate guarantee: in terms of Packet Reception Rate (PRR) defined as the probability that a packet is received at destination.
2. End-to-End delay guarantee: in terms of maximum delay between the furthest node and the destination node.The problem is rewritten as:

Minimize Etot subject to PRR ≥ PRRmin

D ≤ Dmax,

where the objective function Etot is the total energy consumption of the network,
PRRmin is the minimum threshold for the PRR and Dmax is the maximum admitted end-to-end delay.

The basic wireless model in ns

The wireless model essentially consists of the MobileNode at the core,with additional supporting features that allows simulations of multi-hop ad-hoc networks, wireless LANs etc. The MobileNode object is a split object. The C++ class MobileNode is derived from parent class Node. Refer to Chapter 5 for details on Node. A MobileNode thus is the basic Node object with added functionalities of a wireless and mobile node like ability to move within a given topology,ability to receive and transmit signals to and from a wireless channel etc. A major difference between them, though, is that a MobileNode is not connected by means of Links to other nodes or mobilenodes. In this section we shall describe the internals of MobileNode, its routing mechanisms, the routing protocols dsdv, aodv, tora, puma and dsr, creation of network stack allowing channel access in MobileNode, brief description of each stack component, trace support and movement/traffic scenario generation for wireless simulations.

Mobilenode: creating wireless topology

MobileNode is the basic nsNode object with added functionalities like movement, ability to transmit and receive on a channel that allows it to be used to create mobile, wireless simulation environments. The class MobileNode is derived from the base class Node. MobileNode is a split object. The mobility features including node movement, periodic position updates, maintaining topology boundary etc are implemented in C++ while plumbing of network components within MobileNode itself (like classifiers, dmux , LL, Mac, Channel etc) have been implemented in Otcl.

Creating Node movements

The mobilenode is designed to move in a three dimensional topology. However the third dimension (Z) is not used. That is the mobilenode is assumed to move always on a flat terrain with Z always equal to 0. Thus the mobilenode has X, Y, Z(=0) co-ordinates that is continually adjusted as the node moves. There are two mechanisms to induce movement in mobilenodes. In the first method, starting position of the node and its future destinations may be set explicitly. These directives are normally included in a separate movement scenario file.

Network Components in a mobilenode

The network stack for a mobilenode consists of a link layer(LL), an ARP module connected to LL, an interface priority queue(IFq), a mac layer(MAC), a network interface(netIF), all connected to the channel. These network components are created and plumbed together in OTcl.

Link Layer The LL used by mobilenode . The only difference being the link layer for mobilenode, has an ARP module connected to it which resolves all IP to hardware (Mac) address conversions. Normally for all outgoing (into the channel) packets, the packets are handed down to the LL by the Routing Agent. The LL hands down packets to the interface queue. For all incoming packets (out of the channel), the mac layer hands up packets to the LL which is then handed off at the node_entry_ point. The class LL is implemented in ~ns/ll.{cc,h} and ~ns/tcl/lan/ns-ll.tcl.

ARP The Address Resolution Protocol (implemented in BSD style) module receives queries from Link layer. If ARP has the hardware address for destination, it writes it into the mac header of the packet. Otherwise it broadcasts an ARP query, and caches the packet temporarily. For each unknown destination hardware address, there is a buffer for a single packet. Incase additional packets to the same destination is sent to ARP, the earlier buffered packet is dropped. Once the hardware address of a packet's next hop is known, the packet is inserted into the interface queue.

Interface Queue The class PriQueue is implemented as a priority queuewhich gives priority to routing rotocol packets, inserting them at the head of the queue. It supports running a filter over all packets in the queue and removes those with a specified destination address.

Mac Layer Historically, ns-2 (prior to release ns-2.33) has used the implementation of IEEE 802.11 distributed coordination function (DCF) from CMU. Starting with ns-2.33, several 802.11 implementations are available.

Tap Agents Agents that subclass themselves as class Tap defined in mac.h can register themselves with the mac object using method installTap(). If the particular Mac protocol permits it, the tap will promiscuously be given all packets

received by the mac layer, before address filtering is done. See ~ns/mac.{cc,h} for class Tapmplementation.

Network Interfaces The Network Interphase layer serves as a hardware interface which is used by mobilenode to access the channel. The wireless shared media interface is implemented as class Phy/WirelessPhy. This interface subject
to collisions and the radio propagation model receives packets transmitted by other node interfaces to the channel. The interface stamps each transmitted packet with the meta-data related to the transmitting interface like the transmission power, wavelength etc. This met -data in pkt header is used by the propagation model in receiving network interface
to determine if the packet has minimum power to be received and/or captured and/or detected (carrier sense) by the receiving node. Themodel approximates the DSSS radio interface for network interface implementations.

Radio PropagationModel It uses Friss-space attenuation (1/r2) at near distances and an approximation to Two ray Ground (1/r4) at far distances. The approximation assumes specular reflection off a flat ground plane. for implementation.

Antenna An omni-directional antenna having unity gain is used by mobilenodes. See ~ns/antenna.{cc,h} for implementation details.

II. DIFFERENT MAC LAYER PROTOCOLS FOR MOBILE NETWORKING

802.11MAC protocol

Historically, ns-2 (prior to release ns-2.33) has used the implementation of IEEE 802.11 distributed coordination function (DCF) from CMU. Starting with ns-2.33, several 802.11 implementations are available.

Preamble based TDMA protocol

this works is still at a preliminary stage, some practical issues, such as: contention in the preamble phase and time slot reuse in a multi-hop environment are not considered. Unlike contention basedMAC protocol (802.11, for example), a TDMAMAC protocol allocates different time slots for nodes
to send and receive packets. The superset of these time slots is called a TDMA frame.

Different types of Routing Agents in mobile networking

The five different ad-hoc routing protocols currently implemented for mobile networking in nsare dsdv, dsr, aodv, tora and puma.

DSDV

In this routing protocol routing messages are exchanged between neighbouring mobilenodes (i.e mobilenodes that are within range of one another). Routing updates may be triggered or routine. Updates are triggered in case a routing information from
one of t he neighbours forces a change in the routing table. A packet for which the route to its destination is not known is cached while routing queries are sent out. The pkts are cached until route-replies are received from the destination. There is a maximum buffer size for caching the pkts waiting for routing information beyond which pkts are dropped.

All packets destined for the mobilenode are routed directly by the address dmux to its port dmux. The port dmux hands the packets to the respective destination agents. A port number of 255 is used to attach routing agent in mobilenodes. The
mobilenodes al so use a default-target in their classifier (or address demux). In the event a target is not found for the destination in the classifier (which happens when the destination of the packet is not the mobilenode itself), the pkts are handed to the
default-ta rget which is the routing agent. The routing agent assigns the next hop for the packet and sends it down to the link layer. The routing protocol is mainly implemented in C++.

DSR

SRNode is different from the MobileNode. The SRNode's entry_ points to the DSR routing agent, thus forcing all packets received by the node to be handed down to the routing agent. This model is required for future implementation of piggy-backed routing information on data packets which otherwise would not flow through the routing agent.

The DSR agent checks every data packet for source-route information. It forwards the packet as per the routing information. Incase it doesnot find routing information in the packet, it provides the source route, if route is known, or caches the packet
and sends out route queries if route to destination is not known. Routing queries, always triggered by a data packet with no route to its destination, are initially broadcast to all neighbours. Route-replies are send back either by intermediate nodes or
the destination node, to the source, if it can find routing info for the destination in the route-query. It hands over all packets destined to itself to the port dmux. In SRNode the port number 255 points to a null agent since the packet has already been
processed by the routing agent.

TORA

Tora is a distributed routing protocol based on "link reversal" algorithm. At every node a separate copy of TORA is run for every destination. When a node needs a route to a given destination it broadcasts a QUERY message containing the address of the

destination for which it requires a route. This packet travels through the network until it reaches the destination or an intermediate node that has a route to the destination node. This recepient node node then broadcasts an UPDATE packet listing its height wrt the destination. As this node propagates through the network each node updates its height to a value
greater than the height of the neighbour from which it receives the UPDATE. This results in a series of directed links from the node that originated the QUERY to the destination node. If a node discovers a particular destination to be unreachable it sets
a local maximum value of height for that destination. Incase the node cannot find any neighbour having finite height wrt this destination it attempts to find a new route. In case of network partition, the node broadcasts a CLEAR message that resets all
routing states and removes invalid routes from the network. TORA operates on top of IMEP (Internet MANET Encapsulation Protocol) that provides reliable delivery of route-messages and informs the routing protocol of any changes of the links to its neighbours. IMEP tries to aggregate IMEP and TORA messages into a single packet (called block) in order to reduce overhead. For link-status sensing and maintaining a list of neighbour nodes, IMEP sends out periodic BEACON messages which is answered by each node that hears it by a HELLO reply message.

AODV

AODV is a combination of both DSR and DSDV protocols. It has the basic route-discovery and route-maintenance of DSR and uses the hop-by-hop routing, sequence numbers and beacons of DSDV. The node that wants to know a route to a given
destination generates a ROUTE REQUEST. The route request is forwarded by intermediate nodes that also creates a reverse route for itself from the destination. When the request reaches a node with route to destination it generates a ROUTE REPLY
containing the number of hops requires to reach destination. All nodes that participates in forwarding this reply to the source node creates a forward route to destination. This state created from each node from source to destination is a hop-by-hop state
and not the entire route as is done in source routing. **PUMA**

The Protocol for Unified Multicasting Through Announcements (PUMA) is a distributed, receiver initiated, mesh based multicast routing protocol. By default, the first receiver in a multicast group acts as the core (i.e., rendezvous point) for that particular group. PUMA uses a simple and very efficient control message, a multicast announcement, to maintain the mesh. Besides that, multiple meshes can be compiled into a single announcement bucket. PUMA does not require any unicast protocol, and all transmissions are broadcasts. Even though broadcast transmissions are unreliable, the mesh itself introduces some redundancy, and because the mesh includes only groupmembers and the nodes interconnecting them, broadcasts remain
scoped within the mesh. As a multicast announcement propagates throughout the mesh, nodes learn the shortest path to the core. This way, data packets can be quickly routed to the core. On its way toward the core, two things can happen to a data packet: (a) the packet goes all the way until it reaches the core, or (b) a mesh member is hit before reaching the core. Anyway, once a data packet reaches the mesh, the packet propagates only inside the mesh. The core is not a single point of failure, because when the core fails a group member quickly takes the core role.

M-DART

The Multi-Path Dynamic Addressing Routing (M-DART) is a routing protocol for ad hoc networks with the following features:

- proactive, every node keeps information about the available routes;
- multi-path, every node tracks redundant routes to face with topology changes;
- hierarchic, the routing overhead is reduced with a logarithmic factor;
- DHT-based, since a DHT is used at the network layer.

The M-DART extends the DART protocol, first proposed by J. Eriksson, M. Faloutsos and S. Krishnamurthy. The ns-2 implementation has been extensively tested for ad hoc networks up to 4096 nodes. The ./mdart/example/ folder contains an
example of static ad hoc network scenario used for the results published in M. Caleffi, L. Paura, "On Reliability of Dynamic Addressing Routing Protocols in Mobile Ad Hoc Networks", accepted for publication in the special issue on "Architectures
and Protocols forWirelessMesh, Ad Hoc, and Sensor Networks" ofWireless Communications andMobile Computing, 2010.

III. HARDWARE TECHNOLOGIES

A sensor network is an embedded system, or rather a digital system committed to specific duties. Each node consists of a sensor board and a programming board [8]. The sensor board could be differentiated by the specific kind of sensor: light, temperature, humidity, but also distance tracking or GPS receiver. The programming board supplies wireless between a node and a base station (PC). The benchmark is the IEEE 802.15.4 radio standard, with low data rate (around 250 kbps). A node is equipped with a icrocontroller (8-16 bit) and low storage memories.

NS2 platform

NS2 is a node platform for low power and high data-rate sensor network applications designed with the dual goal of fault tolerance and development ease [29]. Designed at the University of California, Berkeley, it is the successor of the popular TelosA and TelosB research platforms. The Tmote Sky platform offers vertical integration between the hardware and the TinyOS operating system. The Tmote Sky module has integrated sensors, radio, antenna, microcontroller and programming capabilities. The low power operation of the module is due to the low power TI MSP430 microcontroller. This 16-bit RISC processor features low active and sleep current consumption. In order to minimize power consumption, the processor in sleep mode during majority of the time, wakes up as fast as possible to process, then returns to sleep mode again. Tmote Sky provides an easy-to-use USB protocol from FTDI to communicate with the host computer for programming, debugging and data collection. It features the Chipcon CC2420 radio for reliable wireless communications [32], which is high configurable for many applications with the default radio setting providing IEEE 802.15.4 [22] compliance. The radio provides fast data rate and robust signal. It is controlled by the microcontroller through the SPI port and can be shut off for low power duty cycled operation. Tmote Sky's internal antenna is an Inverted-F microstrip design, with a pseudo mnidirectional pattern that may attain 50 meter range indoors and up to 125 meter range outdoors. The picture in Fig. 4.1 shows a Tmote Sky platform, compared in size with a Swedish coin.

IV. SOFTWARE TECHNOLOGIES

The link between hardware platform and software equipment is stricter than the other technologies, because of the particular resource constraints (e.g. low power onsumption, reduced memory). It follows the demand of specific ad hoc software technologies. Hence, operating systems for WSN nodes are typically less complex than general-purpose operating systems. In general operating systems in WSNs should fulfill these requirements:

• Robustness: once deployed, a sensor network must work unattended for months or years;
• Low resource usage: sensor network nodes include very small RAM, and run off batteries;
• Multiple service implementation: applications should be able to choose between various implementations;
• Adaptability to evolutions: mote hardware is in constant evolution; applications and most system services must be portable across hardware generations;
• Adaptability to application requirements: applications have very different requirements in terms of lifetime, communication, sensing, etc.

On the other side, they do not require interactivity in the same way as applications for PCs and the operating system does not need to include support for user interfaces.

• **Storage**

Nonvolatile storage is used in WSNs for logging, configuration parameters, files, and program images. Several different kinds of Flash memory are used with different interfaces and different protocols. Lower-layer TinyOS components provide a block storage abstraction for various Flash chips and platform configurations, with one or more application-level data services provided on this substrate.

Time Synchronization and Network Initialization

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

Token Passing Procedure

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.

V. Conclusion

The achieved objectives of this research can be resumed, referring to different levels. Regarding theoretical aspects, we provided a survey on the applications and the design of cross-layer protocols on WSNs. We focused the attention on Improved LEAP protocol, a semi-random protocol for clustered wireless sensor networks,proposed in its first formulation in paper [1]. Improved LEAP satisfies system level requirements on packet delay and successful transmission probability while minimizing energy consumption. Improved LEAP is characterized by a mathematical model that allows for cross-layer optimization without the need for extensive simulations. We enhanced the optimization problem, including a mathematical analysis of the PRR. The main advantage is the possibility of an on-line optimization, which considers the actual dynamics in the network. The major part of the work concerned the implementation of Improved LEAP , using ns2. The protocol allows the network to get excellent performance for low data rate transmissions, ensuring low average node duty cycle,which yields a long network lifetime.

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 Net-
works. 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