Implementing Data Security in Wireless Sensor Network's through Location Aware Multifunctional Key Management Framework

V. Prema Tulasi
Sri Indu College of Engg & Tech
Hyderabad, Andhra Pradesh, India

Prof. D. Durga Bhavani
Scient Institute of Technology
Hyderabad, Andhra Pradesh, India

Prof. Dr. CH.G.V.N.Prasad
Sri Indu College of Engg & Tech
Hyderabad, Andhra Pradesh, India

ABSTRACT

As wireless sensor network's (WSN's) consists of large number of nodes which are deployed in unattended/hostile environments and are exposed to different types of attacks due to compromisation of node or group of nodes. Usually, security designs provide a hop-by-hop security paradigm and are vulnerable to many types of Denial of Service (DoS) attacks, such as report disruption and selective forwarding, puts data availability at stake. In this paper, we overcome these vulnerabilities for large-scale static WSN's by using Location aware multifunctional key management framework. Sensors will be deployed in some area in a grid type model and knows its position based on X and Y coordinates in the grid. Further, embed the location information of each sensor into few of its secret keys which are owned by that sensor node and location-aware property effectively limits the impact of compromised nodes only to their vicinity without affecting end-to-end data security. This framework assures both node-to-sink and node-to-node authentication along the report forwarding routes and demonstrates that this design is highly resilient against an increasing number of compromised nodes and effective in energy savings by avoiding DoS attacks.

Keyword

Sensors, Hop-by-Hop, Sink, Report Forwarding Route, Selective Forwarding Attack, Report Disruption Attack.

1. INTRODUCTION

Wireless Sensor Network's have attracted a lot of attention recently due to their broad applications in both military and civilian operations. WSN's are functioned with limited computing power, limited energy resources, limited communications bandwidth, limited communication range and limited storage resources. A WSN's is a collection of information technology where each sensor collects some information related to a particular event and passes the information from one sensor to other in a sensor network. In addition to one or more sensors each sensor node is equipped with radio transceiver or other wireless communication device, a small microcontroller and on energy source usually battery [1],[2],[4],[7]. The size may vary from shoe-box sized nodes to size of grain of dust. As the size varies cost of sensor nodes also varies ranging from hundreds of dollars to cents depending on the size of sensor network and complexity required of individual sensor nodes. The WSN's are deployed in unattended/hostile environment. It is assumed that the size and shape of terrain or area of our interest is known prior or before.

Sensor nodes can be deployed via aerial scattering or by physical installation. We assume that in an area of our interest, sensor nodes are organized into clusters. Each cluster includes at least x + 1 nodes, where x is a design parameter. In a cluster, one node is elected as a cluster head, and each cluster has a Unique cluster id. A cluster head collects sensor readings or votes from x+1 cluster nodes (including itself), and then reports the result to the base station. Note that the role of cluster head may rotate among the cluster nodes, according to appropriate criteria such as remaining energy. A WSN's consists of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions such as Temperature, Earth quakes and weather Forecasting, Environmental monitoring, Machine health monitoring, Industrial monitoring. As applications gains more importance security issues have also become a hot research topic. The applications in WSN's require high secure mechanism.

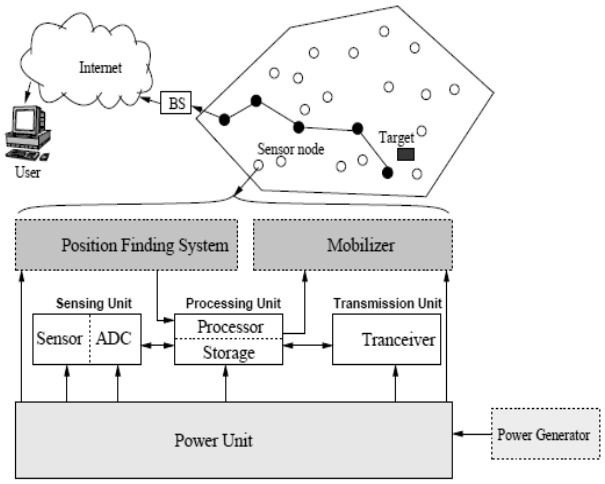


Figure 1: components in a Sensor Node

1.1 Attacks Classifications

Attacks can be classified into two major categories, namely Passive attacks and active attacks based on the interruption of communication act. When it is referred to a ***passive attack*** it is said that the attacker will access data exchanged in the network without interrupting the communication. Examples of passive attacks include ***eavesdropping, traffic analysis, and traffic monitoring.*** When it is referred to an ***active attack*** it is said that the attacker will access the data and disrupt the functionality of the network, meaning information interruption, modification, or fabrication Examples of active attacks include ***jamming, impersonating, modification, denial of service (DoS), and message replay.***

Based on the Active and Passive attacks, we have different types of attacks when the data is exchanged in the network and thus put data security at a high risk.

BLACK HOLE: All the Traffic in a network is redirected to a specific node, which may not forward any traffic at all.
NETWORK PARTITION: A connected network is partitioned into K (K>=2) sub networks where nodes in different sub networks cannot communicate even though a route between them actually does exist.
SLEEP DEPRIVATION: A node is forced to exhaust its battery power.
DoS: A node is prevented from receiving and sending data packets to its destination.
CACHE POISONING: Information stored in routing tables is either modified, deleted or injected with false information.
PACKET DROPPING**:** A node drops packets that it is supposed to forward.
MALICIOUS FLOODING: Deliver unusually large amount of data or control packets to the whole network or some target nodes.
SYBIL ATTACK: A single node pretends to be present in different parts of network.

Since we have different types of attacks which are leading to data security compromise in WSNs. In this paper we are focusing on Data Authenticity, Confidentiality and Availability problems.

1.3 Paper Organization

The paper is structured as follows: Section 2 gives an explanation of data security goals in WSN's and evaluates related work with respect to these goals. Section 3 gives the details of the proposed algorithm design. Section 4 presents the details of proposed algorithms. Section 5 givesExperimental Results and Section 6 finally the conclusion.

2. SECURITY ISSUES IN WSN's AND RELATAD WORK

In this paper, we propose an integrated security design providing comprehensive protection over data confidentiality, authenticity and availability. Our design establishes a location-aware end-to-end data security (LEDS) framework in WSNs. The contributions of LEDS are outlined below.

First, we propose a novel location-aware multifunctional key management framework. In LEDS, the targeted terrain is virtually divided into multiple cells using the concept of a virtual geographic grid. Each sensor node obtains its geographic location via a suitable localization scheme. LEDS then efficiently binds the location (cell) information of each sensor into all types of symmetric secret keys owned by that node. By this means, the impact of compromised nodes can be efficiently confined to their vicinity, which is a nice property absent in most existing security designs. What the attacker can do is to misbehave only at the locations of compromised nodes, by which they will run a high risk of being detected by legitimate nodes if effective misbehavior detection mechanisms are implemented. Second, LEDS provides end-to-end security guarantee. Every legitimate event report in LEDS is endorsed by multiple sensing nodes and is encrypted with a unique secret key shared between the event sensing nodes and the sink. Furthermore, the authenticity of the corresponding event sensing nodes can be individually verified by the sink. This novel setting successfully eliminates the possibility that the compromise of nodes other than the sensing nodes of an event report may result in a security compromise of that event report, which is usually the case in existing security designs. Third, LEDS possesses an efficient en-route false data filtering capability to deal with the infamous bogus data injection attack. As long as there are no more than t compromised nodes in each single area of interest, LEDS guarantees that a bogus data report from that cell can be filtered by legitimate intermediate nodes or the sink deterministically. Effective en-route filtering of bogus data packets also results in significant energy savings as unnecessary forwarding is eliminated. Last, LEDS provides high-level assurance on data availability by dealing with both report disruption attack and selective forwarding attack [3] simultaneously. By taking advantage of the broadcast nature of wireless links, LEDS adopts a one-to-many data forwarding approach, which is fully compatible with the proposed security framework. That is, all reports in LEDS can be authenticated by multiple next-hop nodes independently so that no reports could be dropped by single node(s). Thus, LEDS is highly robust against selective forwarding attack as compared to the traditional one-to-one forwarding approach used by existing security designs [13].

Detailed analysis shows that the proposed LEDS is highly resilient to both types of attacks and partly contributes to the reduction of energy waste due to the incorrect dropping of legitimate data reports. Since there are different Types of Attacks which are leading to security compromise of a Data we are focusing on how to provide security to Confidentiality, Authenticity and Availability.

2.1 Data Confidentiality

Due to the lack of tamper resistance in a node, if a node is compromised the attacker obtains all security information. As the number of compromised nodes grows, communication between uncompromised nodes may also be compromised through malicious cryptanalysis. Hence, this type of attack could lead to data confidentiality [1],[2],[3],[4],[5] compromise in WSN's.

2.2 Data Authenticity

The attacker may use compromised nodes and inject bogus data traffic in WSN's. At that particular point of time, nodes pretend to have detected an event with in its surrounding area or fabricate a bogus event report claiming a nonexisting event at that area and forward the report to sink. The sink is a center equipped which collects data and has sufficient computation and storage capabilities. Hence, this type of attack could lead to data authenticity [1],[2],[3],[4],[5] compromise in WSN's.

2.3 Data Availability

The attacker could use compromised nodes to launch a selective forwarding attack[3] in which case compromised nodes selectively drop the going-through data traffic and data availability can be damaged. Hence, this type of attack could lead to data availability [1],[2],[3],[4],[5] compromise in WSN's. There are many research works which are being carried out. Due to resource constraints most of the proposals are using symmetric cryptography and provide data authenticity and/or confidentiality in a hop-by-hop manner.

3. PROPOSED DESIGN

In this technique the area is divided into grid and consists of many cells. Each cell consists of number of nodes and each node obtains its geographic location by using a suitable localization scheme such as [6], [11], [12],[13]. We assume that each event which occurs is detected by multiple sensor nodes [8],[9],[10]. Once an event happens, the sensing nodes agree on a report, which is forwarded with the help of intermediate nodes towards the sink.

3.1 Location Aware Multifunctional Key Management Framework achieve the following goals

3.1.1 Implementing confidentiality and authenticity

The confidentiality and authenticity of a report should be guaranteed as long as the sending nodes are not compromised. And also if the attacker is able to compromise some nodes he cannot utilize the cryptographic materials to launch attacks at places other than the locations of compromised nodes.

3.1.2 Implementing availability

Be resilient against report disruption attacks and selective forwarding attacks and be able to early detect and drop bogus reports in an effective and deterministic manner, which is having en-route filtering capability.

As we said the virtual grid divides the terrain into multiple cells. The parameters of a geographic virtual grid are as follows: It consist of a reference point referred to as (x0,y0) is said to be the location of the sink. The size of a cell is defined by Ln. A cell is uniquely indexed by its center's location. In a cell if a node say, w is located after network deployment it is called home cell of w, denoted as Iw and Iw=(x1,y1). When an event occurs in a cell,we call it as a event cell. An event report is relayed from the event cell to the sink in a cell-by-cell basis along its report forward route.
The report forward route of node w therefore consists of all the cells that are intersected by the line segment that connects the center of Iw and sink [13].

3.2 Notation and Terms:

For the convenience of description, we use the following notation and terms:

- **Ns**: is the network size.
- **n:** is the number of nodes within one cell.
- **w, x, y, and z**: are the unique ids of sensor nodes.
- **Iw**: is the index of node w's home cell.
- **Ln**: is the side length of a cell.
- **Mk1and Mk2**: are the two master secret keys.
- **Kw**: is the unique secret key shared between w and sink.
- **KIw**: is the cell key shared among the nodes in the same cell Iw.
- **KIw,Ix**: is the authentication key shared Between nodes in cell Iw and nodes in cell Ix
- **RF**: is for the pseudorandom functions.
- **ER**: is the event report to be protected.
- **ENR**: is the encrypted report.
- **ENRw**: is a share of ENR computed through a LSSS,contributed by node w.
- **E. (ER):** is an encryption of ER using key "."
- **Mac. (ER)**: is the MAC computed over ER using key "."
- **A**:is the number of endorsements included when generating a valid report.
- **a**: is the minimum number of endorsements to validate a report .
- **Rd**: is the communication radius of sensor nodes.
- **Pn**: is a large prime number.

3.3 Location Aware Key Mgmt Framework

In this each node stores three different types of location-aware keys:

1) A unique secret key shared between the node and the Sink that is used to provide node-to-sink authentication.The robots first determine a node w's home cell Iw=(x1,y1) and then compute **a** unique secret key Kw, which w shares with the sink as

Kw = RF (mk1|w|Iw)

Where | denotes concatenation operation

2) A cell key shared with other nodes in the same cell that is used to provide data confidentiality. A cell key KIw is further calculated, which is shared among w and other nodes in Iw(x1, y1), and

KIw = RF (mk1|Iw)

3) A set of authentication keys shared with the nodes in its report-auth cells that are used to provide both cell-to-cell authentication and en-route bogus data filtering. A set of authentication keys for all the sensors in the same cell. An authentication key is shared among all the sensors in a given cell and its corresponding report-auth cells. Supposing a report-auth cell of Iw has its location as (x0,y0), then the authentication key between the two cells is

RF (mk2|(x1,y1)|(x0,y0)).

4. PROPOSED ALGORITHM

In this paper we have used two algorithms namely MD5 algorithm which stands for message digest and it is used as a hash function to generate hash for the input message and HMAC algorithm which is used as a message authentication code to implement the data security.

4.1 The implementation of MD5 Algorithm

MD5 digests have been widely used in the software world to provide some assurance that a transferred file has arrived intact. For example, file servers often provide a pre-computed MD5 (known as Md5sum) checksum for the files, so that a user can compare the checksum of the downloaded file to it. Unix-based operating systems include MD5 sum utilities in their distribution packages, whereas Windows users use third-party applications.
The MD5 algorithm will accept messages that are any bytes, but as a sequence of bits (but only for a short while). The algorithm can process the data, it begins by padding the message to a length it can handle. This length just happens to be any number such that length mod 512 is equal to 448, or 64 bits short of being a multiple of 512. The message is padded by first appending a 1 bit to the message, and then enough 0 bits to make the message the proper length. The 1 bit is always added, so even if the message is already the proper length, it will be padded (a message can be padded with anywhere from 1 to 512 bits).

The next step is to calculate the length of the message (before padding). This number is then appended as the last 64 bits of the message, making the message length a multiple of 512. If the message happens to be greater than 2^64 bits long, then the least significant 64 bits of the message are used (length mod 2^64). The MD5 algorithm uses 4 state variables, each of which is a 32 bit integer (an unsigned long on most systems). These variables are sliced and diced and are (eventually) the message digest. The variables are initialized as follows:

A = 0x67452301
B = 0xEFCDAB89

```
C = 0x98BADCFE
D = 0x10325476
```

Now on to the actual meat of the algorithm: the main part of the algorithm uses four functions to thoroughly goober the above state variables. Those functions are as follows:

```
F(X,Y,Z) = (X & Y) | (~(X) & Z)
G(X,Y,Z) = (X & Z) | (Y & ~(Z))
H(X,Y,Z) = X ^ Y ^ Z
I(X,Y,Z) = Y ^ (X | ~(Z))
```

Where &, |, ^, and ~ are the bit-wise AND, OR, XOR, and NOT operators (respectively) that all C programmers should be familiar with. These functions, using the state variables and the message as input, are used to transform the state variables from their initial state into what will become the message digest. For each 512 bits of the message, the following is performed:

```
/* Group the 512 bit message into 16 different 32 bit chunks */
For j = 0 to 15 do
  Set X[j] to MessageBits[j*32] through MessageBits[j*32 + 31]
end

/* Store the digest variables out of harms way for the time being */
AA = A
BB = B
CC = C
DD = D

/* Round 1. */
/* Let [abcd k s i] denote the operation:
   a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
 [ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]
 [ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]
 [ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22
12]
 [ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15
22 16]

/* Round 2. */
/* Let [abcd k s i] denote the operation:
   a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
 [ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]
 [ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20
24]
 [ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]
 [ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20
32]

/* Round 3. */
/* Let [abcd k s t] denote the operation
   a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
 [ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23
36]
 [ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23
40]
 [ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23
44]
 [ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23
48]

/* Round 4. */
/* Let [abcd k s t] denote the operation
   a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
 [ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21
52]
 [ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21
56]
 [ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21
60]
 [ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21
64]


/* And finally update the state variables */
A+=AA
B+=BB
C+=CC
D+=DD
```

(If you are confused on what '<<<' does, let it be known that it represents a bit-wise rotation to the left, e.g. 110011 <<< 2 = 001111) .After this step, the message digest is stored in the state variables (A, B, C, and D). To get it into the hexadecimal form you are used to seeing, output the hex values of each the state variables, least significant byte first. For example, if after the digest:

```
A = 0x01234567;
B = 0x89ABCDEF;
C = 0x1337D00D ;
D = 0xA5510101;
```

Then the message digest would be:

```
67452301EFCDAB890DD03713010151A5
```

And that's how we calculate MD5 digest[17]. We just presented a pseudocode. Taking this algorithm in to consideration we have implemented MD5 Algorithm. By using MD5 algorithm we are able to generate a HASH value to the given message. The advantage of using digest is that message may be of any length but we will get fixed length of digest. To the digest which we had obtained we will use HAMC algorithm to generate message authentication code which is used for authenticating a message so that we can say that data has been received from the sender to the receiver without any interruption by the attacker. Cryptographic hash functions generally execute faster in software than conventional encryption algorithms.

4.2 Introduction To HMAC:

Before proceeding further we first look at what is hash?
A "hash" (also called a "digest", and informally a "checksum") is a kind of "signature" for a stream of data that represents the contents. The closest real-life analog we can think is "a tamper-evident seal on a software package": if you open the box (change the file), it's detected[14].jHashes are "digests", not "encryption". **Encryption** transforms data from a clear text to cipher text **and back** (given the right keys), and the two texts should roughly correspond to each other in size: big clear text yields big cipher text, and so on. "Encryption" is a **two-way** operation.

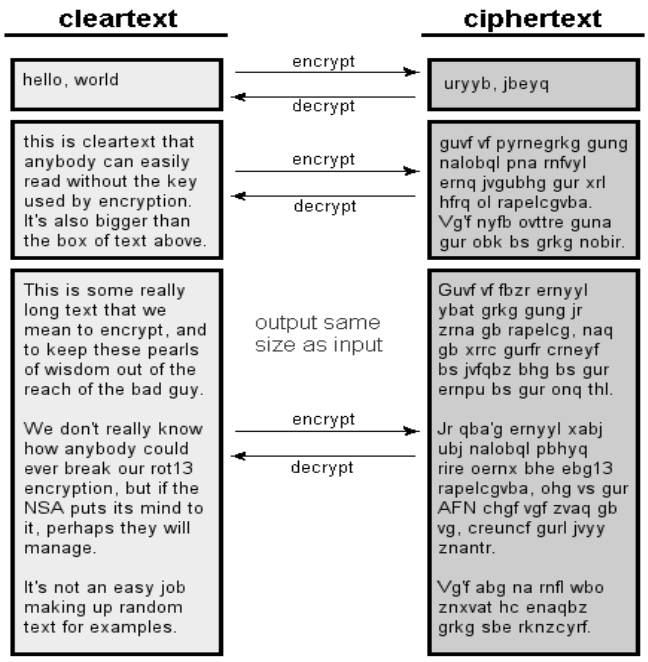


Figure 2: Encryption - two-way operation [14]

Hashes, on the other hand, compile a stream of data into a small **digest** (a summarized form: think "Reader's **Digest**"), and it's strictly a **one way operation**. All hashes of the same type - this example shows the "MD5" variety - have the same size no matter how big the inputs are .

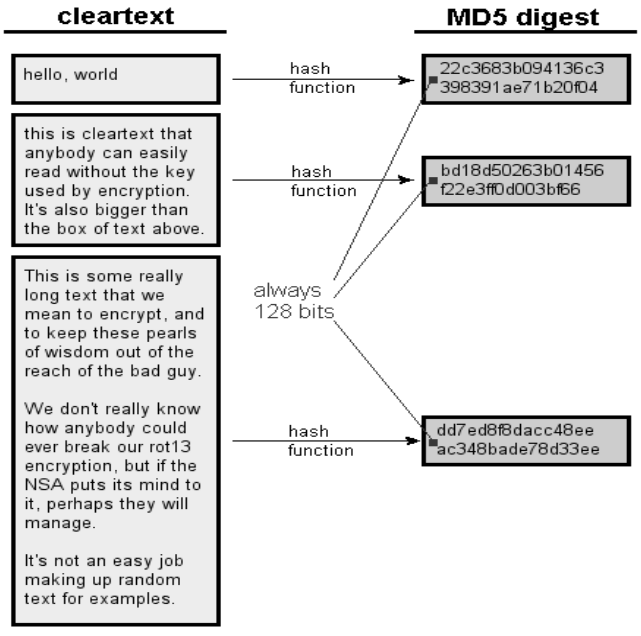


Figure 3 : Hashing - a one-way operation [14]

Message authentication is achieved via the construction of a message authentication code (MAC). MACs based on cryptographic hash functions are known as HMACs. The purpose of a MAC is to authenticate both the source of a message and its integrity without the use of any additional mechanisms. HMACs have two functionally distinct parameters, a message input and a secret key known only to the message originator and intended receiver(s). Additional applications of keyed-hash functions include their use in challenge-response identification protocols for computing responses, which are a function of both a secret key and a challenge message.

An HMAC function is used by the message sender to produce a value (the MAC) that is formed by condensing the secret key and the message input. The MAC is typically sent to the message receiver along with the message. The receiver computes the MAC on the received message using the same key and HMAC function as was used by the sender, and compares the result computed with the received MAC. If the two values match, the message has been correctly received, and the receiver is assured that the sender is a member of the community of users that share the key. Providing a way to check the integrity of information transmitted over or stored in an unreliable medium is a prime necessity in the world of open computing and Communications. Mechanisms that provide such integrity checks based on a secret key are usually called message authentication codes (MACs). Typically, message authentication codes are used between two parties that share a secret key in order to authenticate information transmitted between these parties. This standard defines a MAC that uses a cryptographic hash function in conjunction with a secret key. This mechanism is called HMAC and is a generalization of HMAC.HMAC shall be used in combination with an Approved cryptographic hash function. **HMACMD5** is a type of keyed hash algorithm that is constructed from the MD5 hash function and used as a Hash-based Message Authentication Code (HMAC). The HMAC process mixes a secret key with the message data, hashes the result with the hash function, mixes that hash value with the secret key again, and then applies the hash function a second time. The output hash will be 128 bits in length. An HMAC can be used to determine whether a message sent over an insecure channel has been tampered with, provided that the sender and receiver share a secret key. The sender computes the hash value for the original data and sends both the original data and hash value as a single message. The receiver recalculates the hash value on the received message and checks that the computed HMAC matches the transmitted HMAC. Any change to the data or the hash value will result in a mismatch, because knowledge of the secret key is required to change the message and reproduce the correct hash value. Therefore, if the original and computed hash values match, the message is authenticated.

MD5 is a cryptographic hash algorithm developed at RSA Laboratories. HMACMD5 accepts keys of any size, and produces a hash sequence 128 bits in length. HMAC uses a secret key for the calculation and verification of the MACs. The main goals behind the HMAC construction are:

1. To use available hash functions without modifications; in particular, hash functions that perform well in software, and for which code is freely and widely available.
2. To preserve the original performance of the hash function without incurring a significant degradation.
3. To use and handle keys in a simple way.
4. To have a well-understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the underlying hash function.
5. To allow for easy replaceability of the underlying hash function in the event that faster or more secure hash functions are later available.

4.2.1 Terms And Acronyms

Cryptographic key (key*)*
A parameter used in conjunction with a cryptographic algorithm that determines the specific operation of that algorithm. The cryptographic key is used by the HMAC algorithm to produce a MAC on the data.[15]

Hash function
An Approved mathematical function that maps a string of arbitrary length (up to a pre-determined maximum size) to a fixed length string. It may be used to produce a checksum, called a hash value or message digest, for a potentially long string or message.[15]

Keyed-hash based message authentication code (HMAC)
A message authentication code that uses a cryptographic key in conjunction with a hash function.[15]

Message Authentication Code (MAC)
A cryptographic checksum that results from passing data through a message authentication algorithm. In this standard, the message authentication algorithm is called HMAC, while the result of applying HMAC is called the MAC [15].

Secret key
A cryptographic key that is uniquely associated with one or more entities. The use of the term "secret" in this context does not imply a classification level; rather the term implies the need to protect the key from disclosure or substitution [15].

4.2.2 Parameters And Symbols used in HMAC

HMAC uses the following parameters:

B: Block size (in bytes) of the input to the Approved hash function.
H: An Approved hash function.
Ipad: Inner pad; the byte x'36' repeated *B* times.
K: Secret key shared between the originator and the intended receiver(s).
K0: The key *K* after any necessary pre-processing to form a *B* byte key.
L Block size: (in bytes) of the output of the Approved hash function.
Opad: Outer pad; the byte x'5c' repeated *B* times.
T: The number of bytes of MAC.
Text: The data on which the HMAC is calculated; *text* does **not** include the padded key.
The length of *text* is *n* bits, where $0 \oslash n \oslash 2 - 8B$.
x'N': Hexadecimal notation, where each symbol in the string '*N*' represents 4 binary bits.
|| Concatenation
$\oslash$:Exclusive-Or operation.

CRYPTOGRAPHIC KEYS
The size of the key, *K*, shall be equal to or greater than *L*/2, where *L* is the size of the hash function output. Note that keys greater than *L* bytes do not significantly increase the function strength. Applications that use keys longer than *B*-bytes shall first hash the key using *H* and then use the resultant *L*-byte string as the HMAC key, *K*. Keys shall be chosen at random using an Approved key generation method and shall be changed periodically. Note that the keys should be protected in a manner that is consistent with the value of the data that is to be protected (i.e., the *text* that is authenticated using the HMAC function)[15].

TRUNCATED OUTPUT
A well-known practice with MACs is to truncate their output (i.e., the length of the MAC used is less than the length of the output of the MAC function *L*). Applications of this standard may truncate the output of HMAC. When a truncated HMAC is used, the *t* leftmost bytes of the HMAC computation shall be used as the MAC. The output length, *t*, shall be no less than four bytes ((i.e., $4 \oslash t \oslash L$)). However, *t* shall be at least L/2 bytes (i.e., $L/2 \oslash t \oslash L$) unless an application or protocol makes numerous trials impractical. For example, a low bandwidth channel might prevent numerous trials on a 4 byte MAC, or a protocol might allow only a small number of invalid MAC attempts [15].

4.2.3 HMAC Specification

To compute a MAC over the data '*text*' using the HMAC function, the following operation is performed:
$$MAC(text)t = HMAC(K, text)t = H((K0 \oslash opad) || H((K0 \oslash ipad) || text))t$$

Table 1 illustrates the step by step process in the HMAC algorithm, which is depicted as follows:[15].

Table 1: The HMAC Algorithm

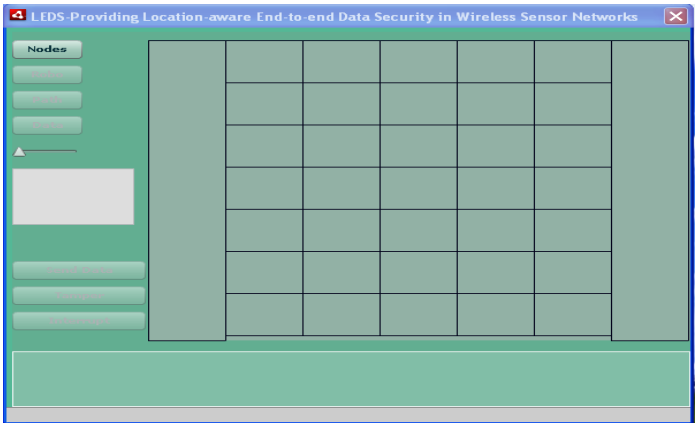
STEPS	STEP-BY-STEP DESCRIPTION
Step 1	If the length of $K = B$: set $K_0 = K$. Go to step 4.
Step 2	If the length of $K > B$: hash K to obtain an L byte string, then append $(B-L)$ zeros to create a B-byte string K_0 (i.e., $K_0 = H(K) \| 00...00$). Go to step 4.
Step 3	If the length of $K < B$: append zeros to the end of K to create a B-byte string K_0 (e.g., if K is 20 bytes in length and $B = 64$, then K will be appended with 44 zero bytes 0x00).
Step 4	Exclusive-Or K_0 with *ipad* to produce a B-byte string: $K_0 \oplus ipad$.
Step 5	Append the stream of data '*text*' to the string resulting from step 4: $(K_0 \oplus ipad) \| text$.
Step 6	Apply H to the stream generated in step 5: $H((K_0 \oplus ipad) \| text)$.
Step 7	Exclusive-Or K_0 with *opad*: $K_0 \oplus opad$.
Step 8	Append the result from step 6 to step 7: $(K_0 \oplus opad) \| H((K_0 \oplus ipad) \| text)$.
Step 9	Apply H to the result from step 8: $H((K_0 \oplus opad) \| H((K_0 \oplus ipad) \| text))$.
Step 10	Select the leftmost t bytes of the result of step 9 as the MAC.

The HMAC algorithm is specified for an arbitrary Approved cryptographic hash function. With minor modifications, an HMAC implementation can easily replace one hashfunction, *H*, with another hash function, *H'*.Conceptually, the intermediate results of the compression function on the *B*-byte blocks(*K0 ⊕ipad*) and (*K0 ⊕opad*) can be precomputed once, at the time of generation of thekey *K*, or before its first use. These intermediate results can be stored and then used to initialize *H* each time that a message needs to be authenticated using the same key.

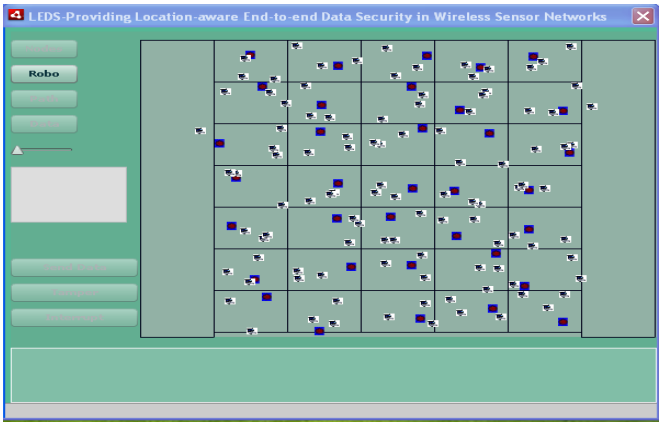
For each authenticated message using the key *K*, this method saves the application of the hash function of *H* on two *B*-byte blocks (i.e., on (*K ⊕ipad*) and (*K ⊕opad*)). This saving May be significant when authenticating short streams of data**.** These stored intermediate values shall be treated and protected in the same manner as secret keys[15].

5. EXPERIMENTAL RESULTS

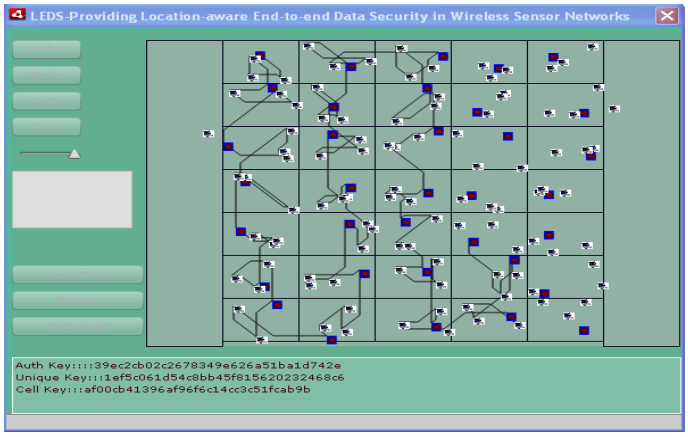
In order to implement Location aware multifunctional key management framework ,we have used MD5 Algorithm and HMAC Algorithm .These are the experimental results which are obtained by using those two algorithms. In these results each screen shot will explain the basic function which it has performed in order to implement this paper..



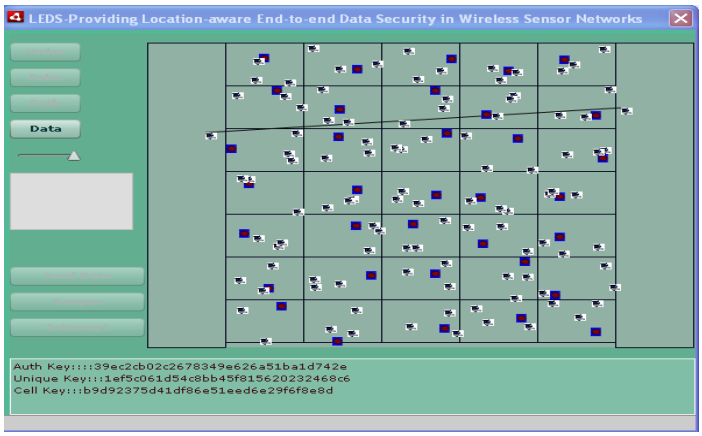
a) SCREEN NO 1



b) SCREEN NO 2



c) SCREEN NO 3

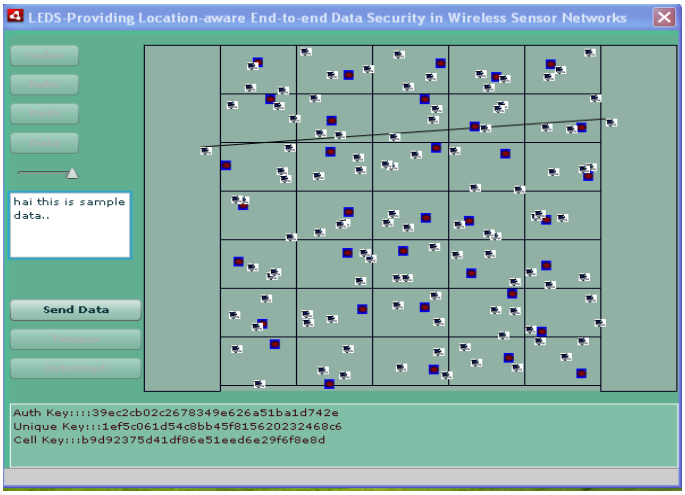


d) SCREEN NO 4

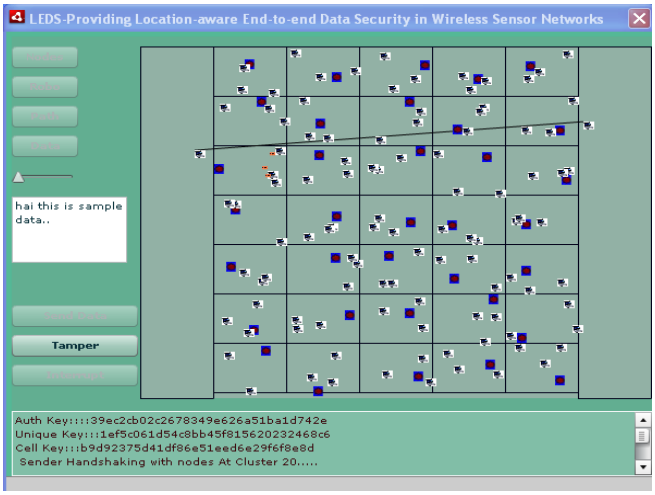
Figure 4: The Experimental Results of this paper which has been done by implementing the HMAC and MD5 Algorithms:
(a)Screen shot of a Nodes Button (b) Screen shot of a robo button
(c) Screen shot of a robo distributing keys (d) Screen shot of establishment of a path

In the screen shot of figure a**)** The nodes button is in enabled state. By pressing this button the nodes will be deployed in the grid. The grid consists of cells and in each cell we have a node and cluster head. In the screen shot of figure b) The robo button is in enabled state. By pressing this button, the robo will start distributing keys to all the nodes in the cell.

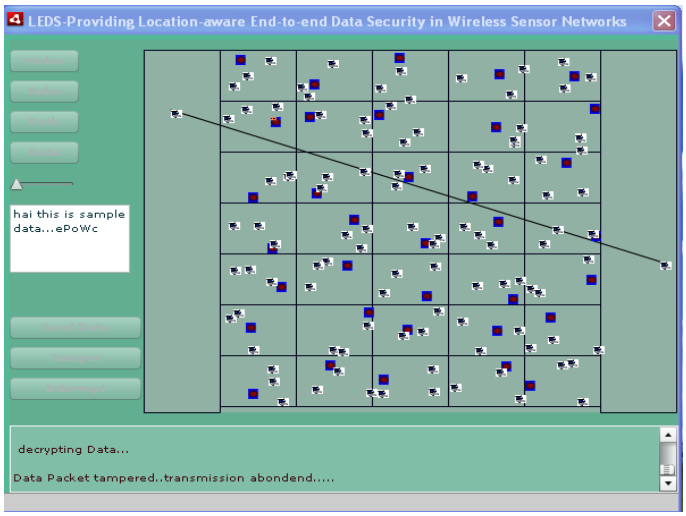
In the screen shot of figure c) The robo is distributing keys from the first sensor in the grid to the last sensor in the grid. When the robo is distributing keys, in the Dialog box we Can observe all the keys like Unique key, Cell key and Authentication key. In the screen shot of figure d**)** The Data button is in enabled state. By pressing this button a text box appears and data should be entered in the text box.



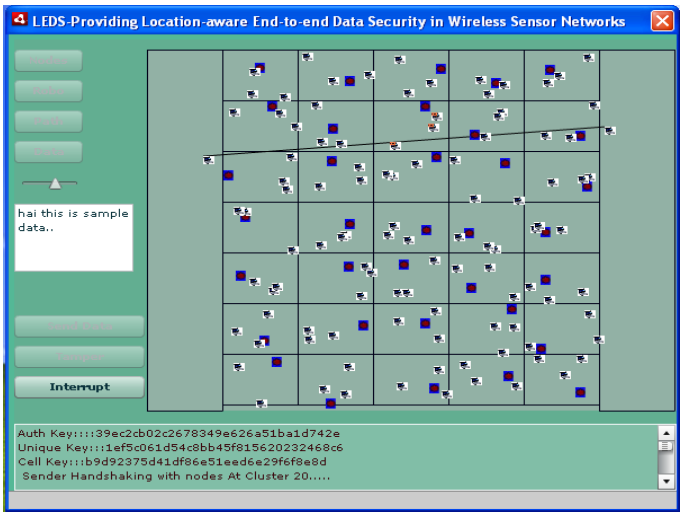
a) SCREEN NO 5



b) SCREEN NO 6



c) SCREEN NO 7



d) SCREEN NO 8

Figure 5: The Experimental Results of this paper which has been done by implementing the HMAC and MD5 Algorithms:
(a)Screen shot of a Send Data Button (b) Screen shot of a Tamper button
(c) Screen shot representing action taken by Tamper button (d) Screen shot of a Interrupt button

In the screen shot of figure a**)** The send data button is in enabled state. When we press this button the data will be moved from source node to destination node. If any interruption of the data is made during the data sending process the data may not reach the destination. In the screen shot of figure b**)** The Tamper button is in enabled state. When we press this button the result is being showed in the next screen. In the screen shot of figure c**)** The data which is being moved from source to destination node gets stopped and transmission of data is abandoned. This action is taking place because we had assumed that sensors are not tamper resistance.
In the screen shot of figure d**)** The interrupt button is in enabled state. When we press this button the data which is being sent gets stopped and no action will take place. The data has reached successfully to the destination sink.

a) SCREEN NO 9 **b) SCREEN NO 10**

Figure 6: The Experimental Results of This paper which has been done by implementing the HMAC and MD5 Algorithms:
(a)Screen shot representing action taken by Interrupt button when it is pressed
(b) Screen shot representing that data reached successfully when no other button is pressed.

In the screen shot of figure a) When we press Interrupt button the data transmission gets stopped and the process again start.Once again data starts from the beginning till it reaches successfully to the destination. In the screen shot of figure b**)** It is displaying a message that the data has reached successfully when no other button is pressed.

6. CONCLUSION

In this paper, we have implemented the key management framework by providing data security in which secret keys are bound to node geographic location. This location-aware property successfully limits the impact of compromised nodes only to their vicinity without affecting end-to-end data security. Furthermore, the proposed multifunctional key management framework assures both node-to-sink and node-to-node authentication along report forwarding routes. Moreover, our data delivery approach guarantees efficient en-route bogus data filtering and is highly robust against DoS attacks. We evaluate our design through extensive analysis, which demonstrates its high resilience against an increasing number of compromised nodes and effectiveness in energy savings that is, achieving 85 percent or more energy savings in contrast to the case without using our design when appropriate parameters are chosen. Furthermore, by using MD5 and HMAC algorithm we are implementing data security in a more efficient way. By using MD5 algorithm we are generating a HASH which is also called as "digest" And HMAC provides message authentication. In the paper[13], they have used MAC algorithm in which we will encrypt and decrypt the message with the help of keys. But in HMAC we will generate the HASH i.e. also called as "Message Digest" and message digest is independent of the message length which we are sending. Due to this reason it takes less time to send the data to the authorized receiver. There are many advantages by using MD5 and HMAC algorithm.

1. We are able to execute the code much faster because we are using hashing technique not the encryption technique.
2. By using HMAC algorithm we are providing message authentication so that the sender and receiver can be confident enough to say that it is the original message and no one has interrupted.
3. By implementing this technique we are able to save the energy resources of a sensor.

7. ACKNOWLEDGEMENTS

The authors would like to express their gratitude to the reviewers for their valuable suggestions and comments.

REFERENCES

[1] D. Carman, P. Kruus, and B. Matt, "Constraints and Approaches for Distributed Sensor Network Security," Technical Report 00-010, NAI Labs, 2000.

[2] A. Wood and J. Stankovic, "Denial of Service in Sensor Networks,"Computer, Oct. 2002.

[3] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," Ad Hoc Networks, vol. 1, no.2, 2003.

[4] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "SPINS:Security Protocols for Sensor Networks," Proc. MobiCom, July 2001.

[5] E. Shi and A. Perrig, "Designing Secure Sensor Networks," Wireless Comm. Magazine, vol. 11, no. 6, Dec. 2004.

[6] S. Capkun and J.P. Hubaux, "Secure Positioning in Wireless Networks," IEEE J. Selected Areas in Comm., Feb. 2006

[7] H. Chan and A. Perrig, "Security and Privacy in Sensor Networks," Computer, pp. 103-105, Oct. 2003.

[8] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected

False Data inSensor Networks," Proc. IEEE Symp. Security and Privacy, May2004.

[9] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical En-Route Filtering of Injected False Data in Sensor Networks," Proc. IEEE INFOCOM, Mar. 2004.

[10] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, "Toward Resilient Security in Wireless Sensor Networks," Proc. ACM MobiHoc, 2005.

[11] L.Lazos and Poovendran,"Serloc:SecureRange-Independent Localization for Wireless Sensor Networks," Proc. ACM Int'l Conf.Mobile Computing and Networking (WiSe '04), Oct. 2004.

[12] W. Du, J. Deng, Y. Han, and P. Varshney, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," ACM Trans. Information and System Security, vol. 8, no. 2, pp. 228-258, May 2005.

[13] Kun Ren, Wenjing Lou, Yanchao Zhang. "LEDS: Providing location Aware end to end data security in wireless sensor networks" IEEE Transactions vol.7 no.5.May.2008.

[14] Steve Friedl's Unixwiz.net Tech Tips An Illustrated Guide to Cryptographic Hashes

[15] Federal Information Processing Standards publications The Keyed-Hash Message Authentication Code(HMAC)

[16] Network Security Essentials by William Stallings.

[17] Copy right owner source institute written by paranoia.

AUTHORS PROFILE

Ms. V.Prema Tulasi, completed her B.Tech from JNTU, Hyderabad currently pursuing M.Tech (CSE) from Sri Indu college of Engg & tech affiliated to JNTU Hyderabad. She worked as a Assistant Professor in Netaji institute of engineering and Technology, Hyderabad for two years. Her Areas of interest includes UNIX, Databases, Computer Networks, and Information Security.

Prof. D. Durga Bhavani, holds a B. E (CSE) from Gulbarga University, M.Tech (IT - Part Time) and is pursuing Ph.D in Data Mining from Acharya Nagarjuna University, Guntur A.P under the guidance of Dr. V. Vijaya Kumar. She has seventeen years of teaching experience in various engineering colleges and at present is working as a Professor in Scient Institute of Technology, Hyderabad, Andhra Pradesh, India. Her area of interest includes Data Mining, Software Testing, Management and Information Technology, Software Process Management, and Computer Organization. She was chosen as a potential candidate to represent professional community in the Cambridge Who's Who 2010 Edition. She is the life member of CSI and ISTE. She has presented many papers in various International Conferences and International Journals.

Dr. Ch.G.V.N.Prasad, has done his M.Tech in Computer Science from JNTU, Hyderabad. PhD (CS) in Data Mining from Allahabad University. He has 20 years of teaching and Industry experience, Also has worked as Scientist in National Informatics Centre, Hyderabad . Currently working as Professor and Head of the Department CSE at Sri Indu College of Engineering &Technology, Hyderabad, He has guided many PG level and engineering students. Life member of CS.