IJCSC 0973-7391 Vol 6 • Number 2 April - Sep 2015 pp. 180-188 Impact Factor: 2.5 Available at www.csjournals.com

Memory Conscious Protocol for Replay Attack Detection and Prevention in WSN

Bharati N. Sawant[1], Bhupendra S. Chordia[2]

[1]Master Student, [2]Assistant Professor

Dept. of Computer Science & Engg., SSVPS'S B. S. Deore College of Engineering, Dhule, INDIA

[1]bharatisawant26@gmail.com; [2]chordiabs@yahoo.com

ABSTRACT: Efficient design and implementation of Wireless Sensor Networks remain one of the most challenging researches due to their wide range of applications. As the use of wireless sensor networks (WSN) grow vastly, so it should require effective security mechanisms. These sensor networks may interacts with the sensitive data and operating in hostile unattended environments so its security concerns should be address from the beginning of the system design are getting much preference. In this paper we present MoteSec-Aware security protocol using SHA hash function based scheme (MAUSHA) to detect replay and jamming attack based on the symmetric key cryptography using AES in OCB mode also it provides access control for both inside memory data and outside network message. For data access control key lock matching method as memory data access control policy (MDACP) is used to prevent unauthorized data access. Previous methods that have been proposed till now, are partially successful in detection of these attacks. But the prevention and securing the actual transmitted messages has been the real problem. Thus, to prevent the replay attacks we use a novel filter based approach which improve the system performance, so by comparing our system with previous system achieves the results in less memory requirements and less energy consumption.

Keywords: Communication architecture, Sensor network security, Hash based approach.

1. INTRODUCTION

A Wireless Sensor Networks (WSNs) are heterogeneous systems containing large number of small devices called sensor nodes and actuators with general-purpose computing elements. These networks should consist of thousands of low cost, low power and self-organizing nodes which are highly distributed either inside the system or very close to it. There are Various applications of WSN includes ocean and wildlife monitoring, monitoring of manufactured machinery, building safety, earthquake monitoring environmental observation, military applications, manufacturing and logistics, forecast systems, health, home and office application and a variety of intelligent and smart systems. The WSNs topology can be vary from a simple star network to the advanced multi-hop wireless mesh network [1].

A typical simple wireless sensor network is shown in fig.1. In which, a complete wireless sensor network usually consists of one or more base stations or gateway, a number of sensor nodes, and the end user.

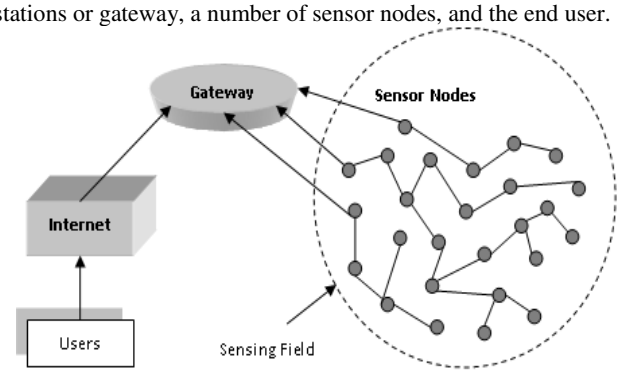

Figure 1: A Simple Wireless Sensor Networks

Sensor nodes are use to measure physical quantities such as temperature, position, humidity, pressure etc. The outputs of those sensor nodes are wirelessly transmit to the base station for data collection, analysis, and logging. End users may also be able to receive and manage the data from the sensor using a website from applications/long-distance in console terminal.

A Wireless Sensor Network (WSN) is usually composed of several resource-limited sensor nodes, which can work collaboratively and deliver useful information to users upon queries and events. Since sensor nodes may collect sensitive information, security and privacy become an important and serious issue that cannot be ignored. Moreover, several real-world scenarios, including community or environment monitoring, smart home, need data transmitted over the network and data stored in nodes' memories. Due to the resource-limited sensor nodes, traditional network security mechanisms are not suitable for WSNs [1] [2].

In wireless sensor network sensor node consist of several parts such as energy source it is usually a battery source, radio transceiver which has an internal antenna or connection to the external antenna, and a microcontroller which is an electronic circuit for the communication with the sensors. These sensor nodes may have different size. The cost of sensor nodes may

vary, it may ranges from a few to hundreds or thousands of dollars, it is solely depends on the complexity of the individual sensor nodes. Variation in size and cost of sensor nodes result in corresponding variation of resources such as energy, memory, communication bandwidth, computational speed.

Thus by taking inspiration from above challenges here we proposed MAUSHA secure network protocol for data access control in WSNs to avoiding data leakage to the adversary or unauthorized party.

2. LITERATURE REVIEW

For the secure network protocol for WSNs, some technologies have been developed, including MoteSec-Aware [3], SPINS [4], TinySec [5], ZigBee [6], MiniSec [7]. Yao-Tung Tsou and Chun-Shien Lu and *et.al,* present a security mechanism called MoteSec-Aware which is build on network layer for wireless sensor networks with the focus to provide secure network protocol and data access control using Virtual Counter Manager (VCM) with synchronized incremental counter approach and applied to detect the jamming and replay attacks using the symmetric key cryptography with AES in OCB [10] mode. But it does not provide prevention against replay and jamming attacks, also it does not suitable for large type of network.

As WSNs are qualitatively different from traditional network, they need a different routing approach for their data to route. Data-centric routing is one of them. In data-centric routing scheme, data are retrieving through querying. Querying the data is base on some of their attributes values. SPIN is a data-centric routing protocol. It fits under event driven data delivery model in which the nodes sense data and disseminate the data throughout the network by means of negotiation.

TinySec and ZigBee are belonging to a kind of secure sensor network link-layer protocol. TinySec achieves low energy consumption by reducing the level of security provided. In contrast, where as ZigBee satisfies high security, but suffers from high-energy consumption. SPINS, on the other hand, achieves low energy consumption by keeping a consistent count between the sender and receiver, such that an initialization vector (IV) is not required to be appended to each packet. MiniSec achieves low energy consumption by appending a few bits of the IV to each packet. Packet loss, however, would cause SPINS and MiniSec to incur more energy consumption for communication and computation, respectively.

Previous works such as ContikiSec [8] and FlexiSec [9] all focus on secure network protocol and do not consider the security of data stored in nodes. Kun *et al.*'s [11], method is composed of three stages first is network admission control, second is network access control, and last one is network access maintenance. These three stages aim to add new eligible nodes into the system, guarantee that all traffic in the system is authenticated, as well as revoke compromised nodes and update group key is unable to defend or detect replay and jamming attacks. Although R-LEAP+ [12], provided an approach to resist replay and jamming attacks, it is too computation intensive to be practical for resource-limited sensor network.

3. OVERVIEW OF PROPOSED METHOD AND CONTRIBUTIONS

Here a secure network protocol for wireless sensor networks is propose which works with low energy consumptions, less memory overhead as well as establishes high security mechanism on sensor nodes. It provides a secure network protocol to permit data transmitted in an encrypted format in air and a filtering capability to permit or deny data access based on some set of rules used for protecting data from illegal access.

The design is based on existing MoteSec-Aware security protocol using SHA-1 [13] algorithm to detect replay and jamming attack which also called as MAUSHA. We modify the MoteSec by using hash based scheme to overcome the storage overhead which results in performance improvement of previous MoteSec method. In which we used Authenticated Encryption Standard (AES) which is most suitable block cipher for WSNs and filtering scheme to prevent replay attack. Memory Data Access Control Policy (MDACP) is presented along with Key Lock Matching (KLM) method to achieve memory data access control. In KLM, each user is coupled with a key like a prime number and each file is associated with a lock value. For each file, there are some corresponding locks, which can be extracted from prime factorization. Through simple computations on the basis of keys and locks, protected memory data can be accessed. Here, data access control is designed exclusively for function nodes.

Our main contributions are summarized as follows:

1. The MAUSHA method is propose here to detect replay and jamming attack.
2. By using the filtering scheme we can provide a secure mechanism for data reporting in wireless sensor Networks and to protect replay attack.
3. MAUSHA is able to achieve the goals of much less memory overhead, less energy consumption and higher security than previous works.

The rest of the paper is organized as follows: Section IV describes the system topology model and attack model. Section V describes the details for proposed method with corresponding algorithms. Section VI includes experimental results and analysis. Finally in section VII we are concluding remarks.

4. SYSTEM MODEL

In this section, we first introduce the wireless sensor network topology that is adopted in our method. Then, the attack model is described.

4.1 Network Topology

The network topology plays a key role in determining the quality of WSNs since it affects both the sensing capability and the wireless connectivity. Here relationship between the sensor nodes is illustrating as shown in fig. 2.

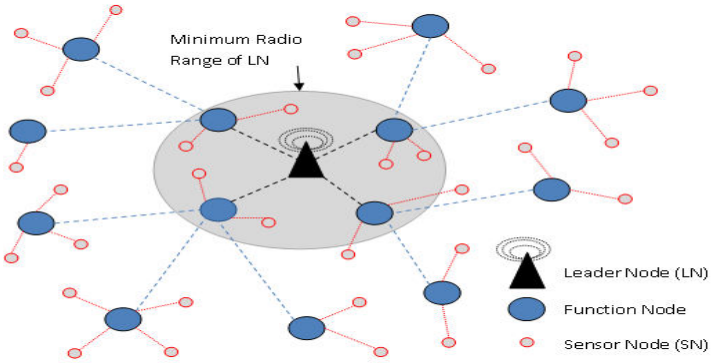

Figure 2: Network Topology

There are three types of nodes, includes Leader Node (LN), Function Node (FN), and Sensor Node (SN), in sensor network topology. They are classified according to their hardware resources with conditions like remaining energy, memory size etc. The network region is partitioned into various physical clusters, each of which contains a FN having charge of SNs in that cluster. Depending on existing applications, clusters may overlaps with each other and hence Sensor Nodes in overlapping may be associated with multiple Function Nodes. In each individual cluster, Sensor Nodes are responsible for collecting sensed data, while Function Nodes will aggregate the data from their associated Sensor Nodes. Function Nodes can also send commands to Sensor Nodes to keep utility data, appliances, *etc*. in inside memory. They forward the received data to their upper level nodes i.e., to Leader Node.

The Leader Node is a network owner with plentiful resources that can query data by an on-demand wireless link connected to all Function Nodes. To prevent storage overflow of Function Nodes, the Leader Nodes can also be periodically dispatched to collect data and empty the storage of Function Nodes.

4.2 Attack Model

The opponent can launch both external and internal attacks. In external attacks, the opponent does not control any valid nodes in the network. Instead, he may attempt to tap sensitive information, inject fake messages, replay previously intercepted messages, and pretend to be valid sensor nodes, with taking the assumption that the opponent can jam the communication between two nodes by transmitting signals that disturb packet reception at the receiver. As for internal attacks, it does not consider that the Function Node will be captured. Instead of it consider that the opponent may attempt to read the data stored in Function Nodes memories, utilizing an unauthorized node to read important data from FNs arbitrarily. Due to all above vulnerabilities, the critical security requirements that need to be satisfied, these are summarizing as follows:

- **Data Confidentiality:** This is the basic property of a secure communication protocol in that data should be kept secret from unauthorized reading.
- **Replay and Jamming Detection:** Communication data should be ensured to be recent and verified that an adversary does not replay or jam data.
- **Data Authentication:** It is necessary to prevent an opponent from spoofing packets. In general, a MAC is use for each packet to verify whether it indeed originates from another legitimate node or is alter during transmission.
- **DoS-Resilience**: For depleting energies the DoS must have attacks, by resisting in particular for resource limited sensor nodes.
- **Data Access:** The opponent should be detected and prevented from accessing data stored on nodes.

4.3 Key Distribution and Update

Since TinySec, ZigBee, and MiniSec are free to use keying mechanism. They are also free to use any key distribution and update mechanism. SPINS and Kun *et al.*'s method makes use of one way key chain and stateless group key update, respectively. In MoteSec- Aware for key distributions and their updates different approaches are used.

To keep the confidentiality of messages transmitted over the network, there are two types of keys used in this system:

- Session keys: Used for Leader/Function Nodes to Broadcast, packet to Function or Sensor Node.
- Pair wise keys: Used for each pair of nodes.

Session key is distributed in advance before deployment of sensors in network. After sensor deployment, pair wise keys are constructing for pairs of sensor nodes by applying CARPY+ scheme [11]. The main advantage of CARPY+ is that it can establish a pair wise key between each pair of sensor nodes without needing any communication. This property is essential in constructing the CFA scheme, because establishing a key via communication incurs an authentication problem, leading to circular dependency. CARPY+ is also flexible to a large number of node compromises so that the complexity for breaking the CARPY+ scheme as $\Omega(2^{\ell+1})$, where ℓ is a security parameter independent of the number of sensor nodes.

When updating the session keys, it customize stateless session keys update schemes, which organize one-way key chain to facilitate the authentication of future keys based on previous ones. In stateless session keys update scheme, network owner α uses the pair wise key $K_{\alpha,\beta}$ shared with each non-revoked node β to encrypt the new session key.

5. PROPOSED METHODOLOGY

5.1 Secure Authentication via CFA

In the CFA scheme, the network planner, before sensor deployment, selects a secret polynomial $f(x,y,z,w)$ from the constrained function set ξ whose coefficients should be kept secret. For simplicity, it assumes that the degree of each variable in $f(x,y,z,w)$ is the same, although they can be distinct. For each node u, the network planner constructs two polynomials, $f_{u,1}(y,z,w) = f(u,y,z,w)$ and $f_{u,2}(x,z,w) = f(x,u,z,w)$.

Since directly storing these two polynomials enables the opponent to obtain the coefficients of $f(x,y,z,w)$ by capturing a few nodes, the authentication polynomial $auth_u(y,z,w)$ and verification polynomial $verf_u(x,z,w)$ should be constructed from the polynomials $f_{u,1}(y,z,w)$and $f_{u,2}(x,z,w)$, respectively, by adding independent perturbation polynomials. Afterwards, the authentication and verification polynomials are stored in node u. For source node u, the MAC attached to the message msg is calculated according to its own authentication polynomial. Let the verification number be the result calculated from the verification polynomial $verf_u(x,z,w)$ by applying the claimed source node ID, the shared pair wise key, and the hashed message into $x, z,$ and $w,$ respectively. The receiver considers the received message authentic and intact if and only if the verification difference, which is the difference between the MAC and its calculated verification number, is within a certain predetermined range.

5.2 Replay Attack Detection and Prevention

Unlike the previous algorithm, our algorithm is design to provide replay protection. Here, as the message type tag, we use a hash value generated by a hash function, instead of using a counter value. We have used SHA-1 [14] as the hash function. SHA-1 is a one-way hash function that produces 160-bit output, when message of any length less than 264 bits is input. The basic algorithm followed for replay detection is illustrated in algorithm 1. The recipient computes the hash value from the received packet using the hash function agreed upon into the variable *Hash Value*.

In this methodology we are using two data structures Filter array and buffer which is used for probabilistic membership test and space overhead problem. It uses number of hash function and their output to check the Set membership on large size datasets. Data structure buffer consist of N size vector which consist of, Addressable cells = {a1; a2; a3; a4} and Hash functions = {h1; h2; h3; h4}. At start data structure is empty where all addressable bits are set to 0. All hash function (hj) is applied on all data (di). For replay detection, we are calculating hash of each packet. If the buffer is empty then put calculated hash on buffer if not then check the calculated hash value with value present in buffer if match found set the flag. If flag set put that hash value in the filter array data structure. All the values in filter array are discarded and the packets present in buffer are only forwarded to next node. Following algorithm is used for replay detection and for filtering of duplicate packets.

Algorithm 1: Replay Attack Detection and Prevention

Input: Incoming packets Pi and Buffer[ ];
Output: Filtered packet Fi
Step 1: Calculate Key $K_{u,v}$ based on Key For pair u and v Key generated randomly.
Step 2: Calculate *EKu,v,* (*msg*) = Encrypt< $K_{u,v}$, msg> and *h*(*msg*) is the hash generated for the message
EKu,v is the Encrypted message.
Step 3: Calculate *MACu*(*v,msg*) = *Generate Hash* (*v, h*(*msg*))
MAC code used for Authentication and Data integrity.
Step 4: Send the packet *M along with header* = <*Header, u, v,*$EK_{u,v}$(*msg*),*MACu*(*v,msg*)>
Step 1: Incoming Packets(All_Packets[], Buffer[])
Step 2: For Each packet P
Step 3: Apply_SHA1_ HashValue(Packet P)
Step 4: Generated HashValue (G_HashValue) = HashValue of ith packet Pi
Step 5: if(Buffer.lengh ==0)
Step 6: Add G_HashValue in Buffer[];
Step 7: if(Buffer[ ]!contains(G_HashValue))
Step 8: Add G_HashValue in Buffer[];
Step 9: else Current Packet is replay packet
Step 11: Create Filter[ ] (Fi)
Step 12: Add Present Packet in Filter Fi
Step 13: Filter all the Replay packets.

5.3 MDACP

In order to efficiently secure information in storage and defend against unauthorized users accessing data, KLM is constructed to realize MDACP In the network, personal information, key materials, and other information that have security concerns will be encrypted by AES-OCFA and stored in the inside memory. In MDACP, each user is associated with a key (e.g. a prime number) and each file is associated with a lock value. For each file, there are some corresponding locks, which can be extracted from prime factorization. Through simple computations on the basis of keys and locks, protected memory data can be accessed. MDACP not only stores encrypted files in nodes but also binds the user keys and specific encrypted files together. This approach has greatly reduced the risk of cracking the keys by attacking the encrypted files. In addition,

by employing the KLM method, when any new user or also file is joined with corresponding lock value and key values can be defined immediately without any change in previously defined locks and keys.

Algorithm 2: MDACP

Input: Key value *Ki* of user *Ui* and lock value *Lj* of file
Fj; $1 \le i \le m$ and $1 \le j \le n$
Output: Access rights *rij*'s
Step 1: Set *rij* = 0 and *Temp*= *Lj*.
Step 2: Calculate *Q* = *Temp/Ki*. If *Q* is an integer, set *rij* = *rij* + 1, *Temp* = *Q*,
repeat this step until *Q* is not an integer or *rij* = *rmax*,
where *rmax* is the maximum of access right.
Step 3: Output access right *rij*.
Step 4: If *rij*=*Yij*, then execute designate tasks and retrieve
corresponding files from the memory;
Step 5: Else reject the request.

5.4 Jamming Attack Detection

In jamming attack detection at receiver end we are calculating data rate for packets received at receiver .and comparing this data rate with threshold data rate if it is greater the jamming attack is detected and receiver will automatically get stopping it will not further receive the packets. Following algorithm is used for jamming detection.

Algorithm 3: Jamming Attack Detection

Input: All Incoming packets Pi;
Output: Jamming Detection
Step 1: For each incoming packet, packet header is generated as
send the packet M along with header =<Header, u, v, $EK_{u,v}$,(msg),MACu(v,msg)>
Step 2: Incoming Packets(All_pacekts[])
Step 3: For First packet P1
Step 4: Calculate the incoming Time T1
Step 5: For Last Packet
Step 6: Calculate the incoming Time T2
Step 7: Calculate Total Time Required
Step 8: Total Time T = T2-T1
Step 9: Set Threshold T
Step 10: if(Threshold >T)
Step 11: All the packets received normally
Step 12: if(Threshold <T)
Step 13: Jamming attack detected.
Step 14: Calculate Packet delay
δ = Total Time – Threshold

6. EXPERIMENTAL RESULTS AND ANALYSIS

6.1 Simulation Setup

For our experiments, we simulate an environment with graphical representation of 100 different sensor nodes randomly placed at different locations in the field of size (700m × 800m). The performance analysis of system is made on Windows based platform under Java Universal Grange Framework (JUNG) [14]. Java software development kit with minimum 1.5 versions or higher and eclipse/net beans IDE is used for simulating the system. Nodes are simulated using the Graphical representation in Java through AWT and swing based classes and using event handling.

Table 1: Simulation Parameter

Parameters	Value
Simulation Model	Default Random Waypoint
Simulation Area	700 metersX800meters
MAC	IEEE 802.11
No of Nodes	1 to 100
Communication range	1 to 250 meters
Energy	Max Up to 200 J
Packet Length	1024 bytes
Distance between Sensor Nodes	Pixel Range
Simulation Time	200 seconds

6.2 Simulation Result

Many experiments have been done to evaluate the performance of the enhanced Security protocol for WSN using the different security scheme. Our analysis focus on replaying and jamming attack detection, memory overhead, semantic security and data access control and energy consumption. Here we compare the existing system MoteSec-Aware with our proposed system MAUSHA results.

Table 2: Simulation Result

Methods	Energy Consumption in mA	Memory Overhead In %	Time Delay In ms
MoteSec-Aware	528	97.65	3000
MAUSHA	426	6.25	1045

The results are shown in Table 2. We can see that longer packets are highly costly because of the extra energy consumed by the radio. The results also reveal that Proposed System have no security overhead increase due to packet communication. In other words, the security overhead of Proposed System is zero. Table 2 reveals that MoteSec-Aware and our proposed system cost approximately 528mAs and 426mAs, respectively. In addition, MoteSec-Aware consumes a constant amount of energy and always outperforms other methods.

6.2.1 *Communication Overhead*

We compare the communication time take place between the systems such as communication without any attack, communication with replay attack, and communication with jamming attack. Lets RT shows replay time. CT shows communication time and, JT jamming time. Thus we can define the actual time for to take place an attack such as for replay and jamming attack as,

$$T_{RT} = RT\text{-}CT \qquad (1)$$

$$T_{JT} = JT\text{-}CT \qquad (2)$$

The resulted communication time graph is shown in fig. 3, which shows time comparison for normal, Replay and jamming communication. Here from eq. (1) and eq. (2) it shows that the communication with jamming attack requires 1591ms more time where as the 374ms time required for communication with replay attack.

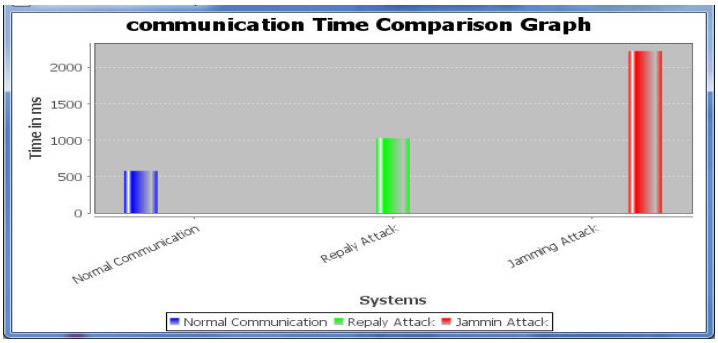

Figure 3: Communication Time Comparison graph

Table 3: Communication Overhead Comparison

WSN Communication Systems	Communication Time in ms
Communication without any Attack	671
Communication with Replay Attack	1045
Communication with Jamming Attack	2262

6.2.2 *Memory Overhead*

The memory overhead graph is shown in fig. 4, which shows that our proposed system comparison with the existing system on the buffer size required for the detection of replay attack. Here we are using the SHA hash algorithm to minimize the buffer size to decrease the detection time and increase the performance of the system.

- For Existing System: packets are compared as it is i.e. if the packet size is 100 kb then actual memory occupied on buffer is 10kb.
- Proposed System: In the proposed system memory overhead is overcome by using the hash function like SHA which convert the whole packet into just 64 bytes.
- The Memory overhead is derived as:

$$\text{Memory overhead\%} = \text{Packet Size}/1024\text{byte}*100 \qquad (3)$$

Thus from eq. (3), it shows that proposed system produce 91.4% less memory overhead than existing system.

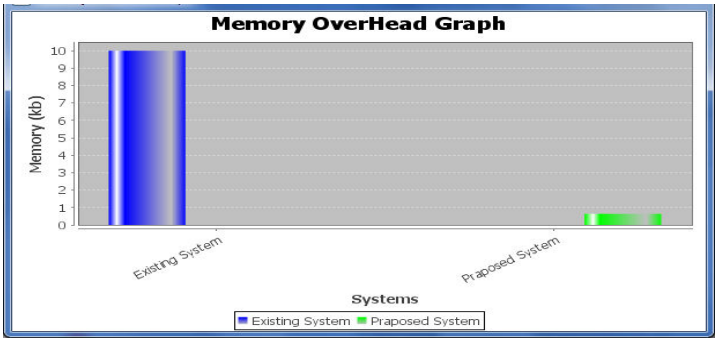

Figure 4: Memory Overhead Graph

Table 4: Memory Overhead Comparisons

Methods	Memory Overhead
MoteSec-Aware	97.65%
MAUSHA	6.25%

6.2.3 *Replay Packets Detection*

All Replay packets can be detected and time required to detect replay packet can be calculated as:

$$RP_{DT} = \sum_{i=0}^{n} RP_{RT} - \sum_{i=0}^{n} NP_{RT} \qquad (4)$$

Where, RP_{DT} is replay packets detection time, RP_{RT} replay packet receiving time at receiver side and NP_{RT} is normal packet receiving time. Thus from eq.(4) calculated replay packet detection ratio is shown in fig. 5, which shows the time required for replay detection in existing and proposed system. This graph clearly shows that proposed system require minimum time than the existing system for replay detection, as we are using hashing technique so we need to compare only 0.64 kb data always as hashing always gives the output of that much hashed value, which is very low than the existing system.

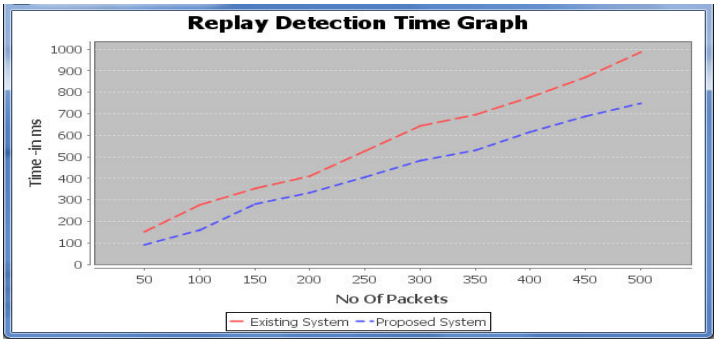

Figure 5: Replay Packets Detection Ratio

6.2.4 *Energy Consumption*

In MAUSHA, the energy consumed for the network communication is calculated as energy spent for MDACP when events happen is E_{DA}, the energy required for to decrypt Encrypted message with time delay δ by using AES Algorithm is δE_{DEC}, and the energy consumed for verifying a replay attack is E_{VER}.

In MDACP at the reception of an access request, two cases are possible for the receiver:

1. The receiver verifies the access request and permits authorized user to access its memory
2. An unauthorized user is blocked from accessing the receiver's memory while the request verified by the receiver is not successful.

Thus, the energy consumption of data access in MDACP that includes two code sections, the process of KLM and the course of memory access, is defined as:

$$E_{DA} = (1 - P_{UA})(E_{KLM} + E_{MA}) + P_{UA} E_{KLM} \qquad (5)$$

Where, P_{UA} stands for the probability that an unauthorized user wants to access data from a node, E_{KLM} denotes the energy consumption of using KLM to verify a request, and E_{MA} is the energy consumption when reading or writing in a node's memory.

Thus, total energy consumed for communication is defined as:

$$E_{TOTAL} = E_{DA} + \delta E_{DEC} + E_{VER} \qquad (6)$$

The total simulation time was set to 100 seconds, and 100 nodes were distributed in a sensing field. Each node receives about 500 packets, and one of the packets is a request to access the data stored in nodes' memories. We obtained the CPU and radio energy consumption associated with each node. The result of energy consumption obtained from our simulation is shown in fig. 6, which shows that, proposed system cost 426mA minimum energy than existing system which cost 528mA energy.

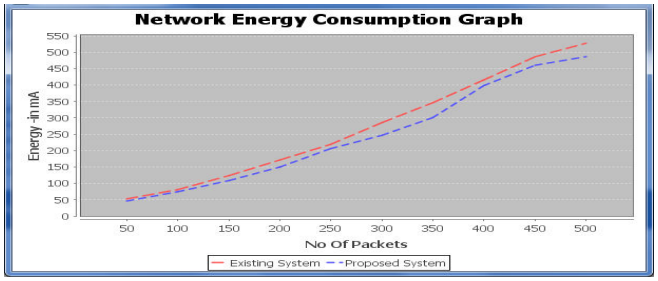

Figure 6: System Energy Consumption Graph

Table 5: Result Analysis

Methods	Replay Detection	Jamming Detection	DoS Resilience	Memory Overhead	Energy Consumption
MoteSec-Aware	Low	High	High	High	High
MAUSHA	High	High	High	Low	Low

7. CONCLUSION

The method proposed here, is implemented on Windows Java Universal Graph Framework. It is an efficient network layer security system and is the fully implemented security mechanism that provides protection for both inside memory data and outside network message. It achieves lower energy consumption during communication and satisfies a high level of security without appending any additional information. Achieves replay attack detection and packet filtering with minimum storage overhead also achieves jamming attack detection. Along with this it provides flexibility of deploying system with higher security platforms.

REFERENCES

[1] Wei Hong David E. Culler, "Wireless Sensor Networks: Introduction," *Communications Of The ACM*, vol. 47, no. 6, pp. 30-33, June 2004.

[2] L. An, N. Peng, and M. Douglas S. Kun, "Securing network access in wireless sensor networks," in *International Conference on Wireless Network Security*, 2009, pp. 261–268.

[3] Yao-Tung Tsou, Sy-Yen Kuo, and Chun-Shien Lu, "MoteSec-Aware: A Practical Secure Mechanism for Wireless Sensor Networks," *IEEE Trans. Wireless Communication*, vol. 12, no. 6, pp. 2817-2829, June 2013.

[4] R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar A. Perrig, "SPINS: security protocols for sensor networks," in *International Conference on Mobile Computing and Networking*, 2001, pp. 189–199.

[5] N. Sastry, and D. Wagner, C. Karlof, "TinySec: a link layer security architecture for wireless sensor networks, 2004 in Proc," in *International Conference on Embedded Networked Sensor Systems*, 2004, pp. 162–175.

[6] Technical Report Document 053474r06, "ZigBee Alliance, Zigbee specifications," 2005.

[7] G. Mezzour, A. Perrig, and V. Gligor M. Luk, "MiniSec: a secure sensor network communication architecture," in *International Conference on Information Processing in Sensor Networks*, 2007, pp. 479–488.

[8] L. Casado and P. Tsigas, "Contikisec: a secure network layer for wireless sensor networks under the Contiki operating system," in *Nordic Conference on Secure IT Systems*, 2009, pp. 133–147.

[9] D. Patel, and K. Dasgupta, D. Jinwala, "FlexiSec: a configurable link layer security architecture for wireless sensor networks," *Inf. Assurance and Securit*, vol. 4, no. 2, pp. 582–603, 2009.

[10] National Institute of Standards and Technology, Computer Security Division, AES standard NIST, , 2010. [Online]. http://csrc.nist.gov/archive/aes/index.html,

[11] Chun-Shien Lu, and Sy-Yen Kuo Chia-Mu Yu, "Non-interactive pairwise key establishment for sensor networks," *IEEE Trans. Inf. Forensic and Security*, vol. 5, no. 3, pp. 56–569, 2009.

[12] S. Blackshear and R. Verma, "R-LEAP+: Randomizing LEAP+ Key Distribution to Resist Replay and Jamming Attacks," in *ACM Press*, Switzerland., 2010, pp. 1985–1992.

[13] SHA-1. NIST. Secure hash standard. In Federal Information Processing Standard. National Institute of Standards and Technology. [Online]. Available: http://www.itl.nist.gov/fipspubs/fip180-1.html.

[14] Joshua Madadhain, Danyel Fisher, "Analysis and Visualization of Network Data using JUNG". [Online]. http://www.jstatsoft.org/