MCPSC: Memory Conservation Protocol for Secure Communication in Wireless Sensor Networks

Suleman Awan *, M. Ali Rafique †
Department of Electronics & Telecommunications Engineering, Polytechnic University of Turin, Italy * †
Email: sulemanawan@gmail.com *, alirafique@hotmail.com †

***Abstract*—Wireless Sensor Networks (WSNs) are being used everywhere around us whether it be a smart office or an application related to environmental monitoring, healthcare, surveillance, education and training. Most of the time such application domains involve handling of sensitive data that would require secure dissemination of information from sensor node to the data collection point (target node). However, scarcity of memory, battery life and computational power has always posed a serious challenge to ensure secure communication in such versatile application domains. In this paper we have proposed an algorithm that establishes secure communication in WSNs with memory conservation. Our proposed algorithm divides the sensor nodes into two groups, strong nodes and weak nodes based on their memory capacity. The algorithm itself utilizes a hybrid cryptosystem; combination of public and private key cryptography, where the exchange of an encrypted symmetric key is done using public key cryptography.**

***Index Terms*—Cryptosystem, Cryptography, Key management, Memory conservation, Sensor, Security.**

I. Introduction

A Wireless Sensor Network (WSN) is a network with the set of inexpensive small form factor wireless devices that organize themselves into a cooperative network. These tiny sensor nodes are used to monitor the environmental conditions like air pressure, temperature, humidity etc. Each sensor node has the capacity of doing only limited processing but due to their coordination in a big network with all other sensor nodes, information can be propagated to more powerful nodes (also known as sink). Unlike traditional wireless devices, wireless sensor nodes do not need to communicate directly with the nearest high-power control tower or base station, but only with their local peers. Instead of relying on a pre-deployed infrastructure, each individual sensor or actuator becomes part of the overall infrastructure. A sensor node might vary in size from that of a hand held device down to the size of a grain of dust. The size and cost of sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth. Applications of wireless sensor networks are in critical systems like hospitals, military, airports, home security system and surveillance. Because of much limitations and challenges that we face in WSN, classic security mechanism should be avoided. It is therefore challenging to come up with a new technique that addresses the issue while keeping in mind the limitations of WSN. One important aspect of security is to know which specific thing should be protected and how to protect it. Sensor networks are vulnerable against external and internal attacks due to their unique characteristics like open area deployment, public communication channels, and limited resources of WSN nodes, difficult monitoring and control of actual state of the deployed nodes. Other than protection of information from illegal use, disclosure, modification, destruction or unauthorized access, the three most important security aspects are data confidentiality, integrity and availability [1]. In our proposed memory conservation algorithm for secure handshake between sensor nodes, we opted for symmetric pair wise keys scheme in wireless sensor network because of these underlying reasons:

- Wireless sensor nodes are generally low powered and symmetric key encryption uses less power.
- These nodes have limited memory and symmetric key encryption uses rather less memory.

Moreover, symmetric key encryption is much faster than asymmetric key encryption. It also prevents widespread message security compromise. Many researchers have claimed that symmetric key encryption could be the best choice for the encryption of data keeping in view the physical constraints of WSN nodes [2]. After selecting the symmetric key encryption, the next step is to find out how we could establish the key among the nodes [3]. A simple technique would be to provide all nodes with the secret keys at the time of network deployment. Once the network is deployed, we cannot add any further nodes in the network. Moreover, we cannot apply such technique to large networks due to scalability issues. However, there is an alternative way to exchange keys among nodes for such type of networks that is called 'hybrid cryptosystem'. Hybrid cryptosystem utilizes combination of public and private key cryptography, where the exchange of an encrypted session key is completed using public key cryptography. The following encrypted session is then pursued with private key cryptography. We have used this technique to make our algorithm more secure and robust since security of asymmetric key encryption is considered higher.

II. Related Work

Several protocols for encryption and authentication have been proposed by researchers in the past decade for secure dissemination of information on wireless sensor networks. One of the earliest and special methods for exchanging keys was Diffie-Hellman key exchange. This method enables two different users who do not know each other to establish a shared secret in between them over an insecure communication

ISBN: 978-1-4799-5754-5/14/$26.00 ©2014 IEEE

channel. This protocol provides the basis for a variety of authenticated protocol and is used for perfect secrecy. For exchanging keys, another cryptosystem is RSA (Rivest Shamir Adleman) in which the encryption key is public and differs from the decryption key which is kept secret. In this particular algorithm asymmetry is based in the factoring the product of two large prime numbers along with the auxiliary value, as their public key. But due to the limitations of the wireless sensor nodes described in the previous chapters, we cannot manage to use them for key exchange [4]. Using a fast multiplication algorithm, Qing [5] presented an authentication scheme that can be employed in WSNs based on ECC (Elliptic Curve Cryptography). This resulted in reduction of computational cost. Another similar experiment was performed by Arazi in [6], [7] where a group key generation technique based on the ECC for the clustered based wireless sensor networks is proposed. They distributed the computational load of key generation process among neighboring nodes to decrease the execution time and balancing the power consumption [8]. After reviewing the experiments and results of several researchers, we reached a conclusion that ECC supports a higher level of security than the standard encryption techniques for instance RSA, AES etc., while using shorter key length and introducing less computational overhead.

III. Proposed Algorithm

While cryptographic mechanisms are optimized to suit the resource constrained sensor and actuator networks, where the networks are integrated into the internet, such cryptographic mechanisms can still experience poor performance due to the size of the packets exchanged and the length of the keys. Furthermore, it is difficult to distribute the security keys in the federated combination of networks [9]. Thus these mechanisms need certain assumptions while designing our algorithm:

- Static Network: Our network is static that means the nodes are not mobile.
- Everlasting Battery: Power supplied to the nodes is supplied by long lasting batteries.
- Computational and communication capabilities of all nodes are equal.
- Few nodes are assumed to be strong nodes, helping in doing handshaking between the weak nodes.
- Nodes will establish a symmetric key among themselves at time of network deployment
- Memory: Nodes have enough space to store hundreds of bytes of keying material.
- Attacker will know information of a node, if the node is being compromised.

A. Cryptographic Mechanism

Elliptic curved based algorithms use significantly smaller key sizes than their non elliptic curve equivalents. The approximate equivalency in the security strength for symmetric algorithms compared to the standard asymmetric algorithms and ECC algorithms are mentioned below in the table I.

TABLE I
NIST Recommended Key sizes

AES key length(bits)	RSA key length(bits)	ECC key length(bits)
Symmetric	Asymmetric	Asymmetric
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Its obvious from above comparison that to in order to take strength of 80 bits symmetric key, a standard asymmetric algorithm should have an enormous key of 1024 bits. Such huge number of bits are not practical for wireless sensor networks due to the amount of processing that would be required, and hence the speed of the operation. For the same operation 160bits of Elliptic Curve Cryptography key would have been required.

B. Overview of Proposed Algorithm

In the proposed algorithm, different pair wise keys are computed for each pair of nodes. Each pair wise key is generated randomly by one node of the pair which provides it to the other node through a cryptography algorithm. Asymmetric key is distributed in this way:

- Every node will have a pair of private and public key stored in it together with the Node_ID.
- Each node stores an authentication table, which is composed of Node_ID, Bit_Mask and the precomputed Hash on the authentic public keys.

C. Authentication table

To make this algorithm memory conservative, strategy chosen is to divide the network in two parts on the basis of amount of data stored in their authentication table.

- A_type nodes(Weak nodes)
- B_type nodes(Strong nodes)

Weak nodes are large in number; however the strong nodes are few in number. Each node in its memory, stores an authentication table which is composed of Node_ID, Bit_Mask and the precomputed Hash on the authentic public keys. A_Type nodes will store this data for only B_Type of nodes. However B_Type nodes, being stronger will save data for A_type of nodes and also B_type of nodes. And each strong node will help many weak nodes to communicate with each other by helping them in a handshake.

D. Test Case Example

To illustrate the proper working of our proposed algorithm, we consider a test case scenario that assumes a wireless sensor network comprising of six (6) nodes. We take five weak nodes, such as (A1, A2, A3, A4 and A5) and one strong node (B_type) for helping A_type nodes for communication as explained before. The authentication table of node A1 will store all available information including the Node_ID, M_bits and the Hash of B_type of node. But for all other A_type of nodes it will just store Node_ID, hence saving the memory as shown in the table II. However, the strong node (B_type) will store complete information about all the neighboring nodes as shown in the following table. This data stored in the authentication table is used to authenticate each other before starting proper communication.

TABLE II
AUTHENTICATION TABLE FOR A_TYPE NODE (A1)

Node_ID	M_bits	Hash
B	0110	********
A2	N/A	N/A
A3	N/A	N/A
A4	N/A	N/A
A5	N/A	N/A

TABLE III
AUTHENTICATION TABLE FOR B_TYPE NODE

Node_ID	M_bits	Hash
A1	1110	********
A2	0011	********
A3	1100	********
A4	1010	********
A5	0110	********

E. Communication between weak and strong nodes

Our proposed algorithm divides the WSN nodes in two types. If A_type node wants to communicate with B_type node then to start this communication they need to authenticate each other using their stored information. If A1 need to authenticate B, then A1 have the M_Bits and Hash of B in its authentication table as shown in table 2. Similarly B has M_Bits and Hash of A1 in its authentication table. In order to authenticate each other, they will share their public key with each other. When B will send its public key to A1, A1 will remove M_bits from its public key according to bit masking, and will calculate its hash value. Then it will compare the computed hash value with the hash value stored in its authentication table for B. If both values match then B will be authenticated. Similarly B will authenticate A1. This working can be graphically shown as follows:

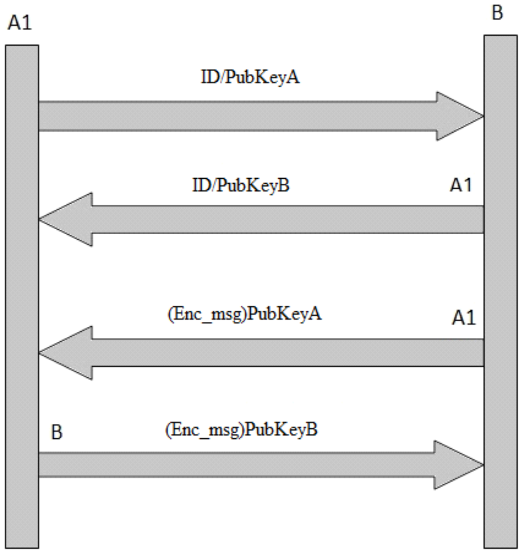

Fig. 1. Communication between weak and strong node

F. Communication between weak nodes

Now if A1 needs to communicate with same type of Node, lets suppose with A2, then A1 does not have M_Bits and Hash for A2 in its authentication table. Now in order to communicate, A1 will send hello message to A2. After receiving hello message, A2 will broadcast handshake request $(A2- > A1)$ to B_Type node asking for symmetric key. B_type node already have pair wise keys for A1 and A2, as it already authenticated them, and it stores data(M_Bits and Hash) for them in its authentication table. B_type node will respond back to A2. A2 will send ACK back to B node ensuring the start of handshaking. Now B generates a random number R, and sends it to both the nodes, encrypting the message with the respective pair wise key. Hence A1 will be able to communicate with A2, keeping the algorithm more secure and memory efficient than before. Graphical representation for communication and key exchange between two same kinds of nodes is given below:

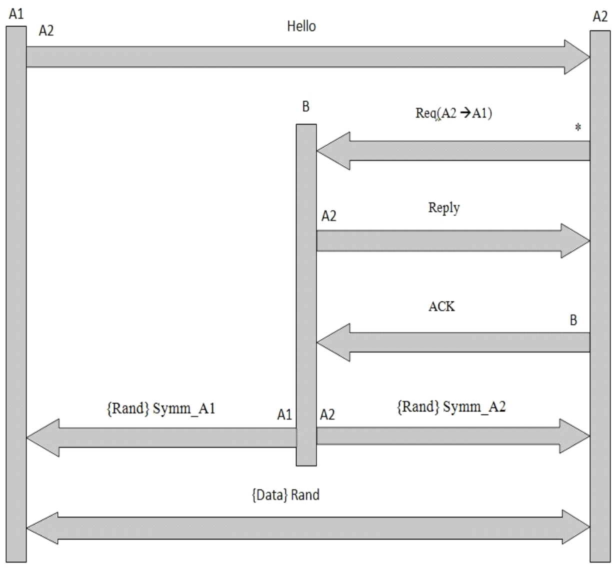

Fig. 2. Communication between two weak nodes

G. Experimental environment

In this section we describe tools that have been used in performing the experimental work. Working environment used for publishing results for this paper is:

- Ubuntu 12.04
- Tiny OS
- Eclipse (with YETI plugin)
- Tiny ECC library

IV. Analysis of Algorithm

As discussed earlier that main purpose for designing this protocol was to reduce the memory requirements of majority of the nodes. Memory storage for A_type of nodes, which are large in number, is reduced significantly. As A_type of nodes will not store data (M_Bits and Hash) for other Anodes as B_type nodes will help them in doing the handshaking and furthermore in their communication followed by authentication. The table below shows that the A_Type of nodes will store data (M_Bits and Hash) for just B_type of nodes, hence our protocol saves the memory for A_type of nodes.

TABLE IV
Memory Analysis of the nodes

Memory Pre_Storage for A_type Nodes	$(Pri\_Key + Pub\_Key) + y(i + j + m)$
Memory Pre_Storage for B_type Nodes	$(Pri\_Key + Pub\_Key) + (x * (y - 1))(i + j + m)$
Memory working storage for A_type Nodes	$(Pri\_Key + Pub\_Key) + y(SymKey) + x(SymKey)$
Memory working storage for B_type Nodes	$(Pri\_Key + Pub\_Key) + (x * (y - 1))(i + j + m)$

V. Results and Discussions

In our test case implementation, we observed that A_type of node will store 1298 bytes in RAM. For making pair wise key with 4 more A_Type of nodes, 64 more bytes will be required so in total 1362 bytes in RAM will be consumed. Because at the end of the handshake A_type node will have symmetric key established with 4 more A_type nodes. B_Type of node will store 1736 bytes in RAM. These nodes have to store much more keys then A_Type of nodes. During its establishment of keys with A_Type of nodes, there is no special memory requirement. These nodes only save the keys that are established asymmetrically.

A. Results for Key Generation time

In order to evaluate our algorithms working, first we looked at the results in terms of key generation time. Several readings were taken in order to analyze when two same type, say A_Type of nodes communicate with each other by the help of B_Type node. As its shown in graphical representation Fig 2, that there will be some extra exchange of message which will help both A_Type of nodes to communicate with each other, so time to generate pair wise keys between them will be slightly more than the communication between different types of nodes.

B. Results for total number of packets exchanged

After obtaining results for time consumed in key generation process, we then focused our attention to observe the total number of packets that were exchanged between the nodes. The topology and working environment remained the same during this experiment like before. These results are regarding the total number of packets sent, when same type of nodes (say A_Type) took help from B_type node in communication. It is observed that a linear increase in total number of packets exchanged with the addition of more weak nodes trying to communicate via strong node (B_type).

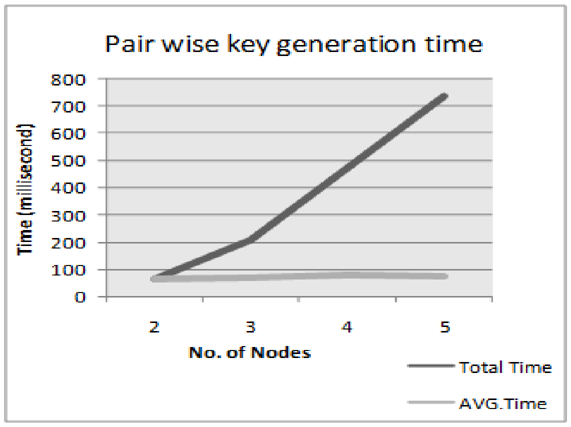

Fig. 3. Pair wise key generation time of the nodes

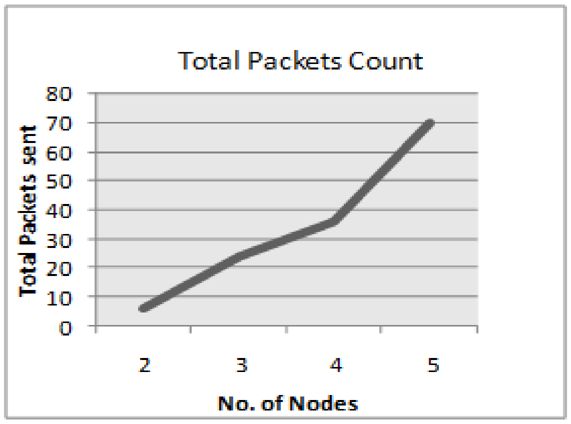

Fig. 4. Total packets exchanged for communication

VI. Conclusion

From the results of implemented algorithm, it can be seen that memory required to establish pairwise keys has been saved for weak nodes, which are higher in number. Moreover, it is also observed that this algorithm also ensures authentication, confidentiality and nonrepudiation at the same time. Time taken to establish the key is almost same as other key management techniques using asymmetric technique (for communication between different types of nodes) and symmetric technique (for communication between same types of weak nodes). In order to conserve memory, computation and communication metrics are affected but by a very small margin. We believe that this algorithm can be adopted for such real networks where most of the WSN nodes have memory constraints yet secure communication between the nodes is of paramount importance.

VII. Acknowledgment

We acknowledge the help taken from Prof. Maurizio Rebaudengo who works in DAUIN, Politecnico di Torino, Italy.

References

[1] Vladimir Cervenka, Dan Komosny, ”Energy efficient public key cryptography in wireless sensor networks”, published in book titled ”Innovations and advances in computers, information, system sciences & engineering”, Volume 152, 2013, pp497-509.

[2] Lingling Si, Zhigang Ji, Zhihui Wang, ”The application of symmetric key cryptographic algorithms in wireless sensor networks”, Physics Procedia 2012, pp552-559.

[3] Al-Sakib Khan, Hyung-Woo Lee, Choong Seon Hong, ”Security in wireless sensor networks: Issues and challenges”, Feb 2006 ICACT2006, ISBN 89-5519-129-4.

[4] Sarmad Ullah Khan, ”Key Management in Wireless Sensor Networks, IP-Based Sensor Networks”, Content Centric Networks in Politecnico di Torino, March 2013. [Online available: http://porto.polito.it/2506342/]

[5] Qing Chang, Yong-ping Zhang, Lin-lin Qin , ”A node authentication protocol based on ECC in WSN”, Computer Design and Applications (ICCDA), 2010

[6] Arazi, O., Qi, H., ”Self-certified group key generation for ad hoc clusters in wireless sensor networks”, Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on , vol., no., pp359-364, 17-19 Oct 2005.

[7] Arazi, O., Elhanany, I., Rose, D., Qi, H., Arazi, B., ”Self-certified public key generation on the Intel mote 2 sensor network platform”, Wireless Mesh Networks, 2006. 2nd IEEE Workshop on, vol. no., pp118-120, 25-28 Sept 2006

[8] Khan Sarmad Ullah, Lavagno L. Pastrone C., Spirito M.A., ”An energy and memory efficient key management scheme for mobile heterogeneous sensor networks”, Risk and security of Internet and systems (CRiSIS), 6th International conference. pp1-8. Sept 2011

[9] Moshaddique Ameen, Jingwei Liu and Kyungsup Kwak, ”Security and Privacy Issues in Wireless Sensor Networks for Healthcare Applications”, Journal of Medical Systems, 2012, Volume 36, Issue 1, pg93-101.

[10] Elaine Barker and Allen Roginsky, ”Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths”, NIST Special Publication, NIST special publication 800-131A.