ME-SMOCK: A Memory Efficient Public Key Scheme for WSN

Divya Jegatheesan, Margret Johnson

Abstract— Wireless Sensor Networks (WSNs) are constructed by a set of small devices, called the sensor nodes, with sensing and wireless communication capabilities. They are widely used in many applications such as military, ecological, and health-related areas. WSNs have certain constraints like low computation capability, limited energy resources, small memory and poor resilience to physical capture. Also, they are mostly deployed in adverse or even in hostile environment. These constraints and issues make security in WSNs a challenge. So, efficient key distribution and management schemes are must. In this paper, a novel key management scheme based on deployment knowledge has been proposed which adds memory efficiency to the existing public key based integrity scheme. Using this scheme memory efficiency can be improved by minimizing the total number of keys to be stored in each sensor node without altering the security.

Index Terms — Cryptography, Key Management, Key Predistribution, Resilience, WSN

1 Introduction

Advances in wireless communication and electronics have resulted in the development of tiny sensor nodes which includes sensing, data processing, and communication components and hence it is possible to deploy Wireless Sensor Networks (WSNs) which represent a significant improvement over traditional wired sensor networks [1]. WSNs are widely used in military applications, environmental applications, health applications, home applications, commercial applications and so on. These applications require communication in WSN which must be highly secure.

To inhibit the adversary from intercepting the transmitted information in the wireless communication channel and to prevent the false information distribution in the network, authentication and confidentiality must be used to achieve network security, which require key management [2].

The key establishment technique for a secure application must incorporate certain requirements like authenticity, confidentiality, integrity, scalability, control resilience, flexibility, fault-tolerance, memory efficiency, energy efficiency and self-healing.

Cryptographic schemes can be either symmetric or asymmetric. Most of the key management schemes in WSNs are symmetric based. Inspite of perfect security, asymmetric schemes result in more memory consumption and communication overhead. In this paper, a novel key management scheme is proposed which focuses on reducing the memory usage and hence paving a way to use asymmetric cryptography in WSNs.

From the traditional asymmetric methods [3], [4], it is obvious that certificate based authentication leads to high overhead reducing the service availability. In contrast, self-contained public key management scheme will reduce communication overhead.

- *Divya Jegatheesan is currently pursuing masters degree program in computer science and engineering in Karunya University, India. E-mail: divs.it@gmail.com*
- *Margert Johnson is currently working as Assistant Professor in computer science and engineering department of Karunya University, India. E-mail: margret_cse@karunya.edu*

2 SMOCK - an Overview

SMOCK is a self-contained public key management scheme which achieves zero communication overhead and high service availability [5]. According to this scheme certain number of schemes will be stored in the nodes by off-line trusted server.

SMOCK requires a pair of public and private keys to encrypt and decrypt respectively. If node A needs to communicate node B, A will send an ID request to node B. Depending upon the ID that node A receives from node B, it will encrypt the message using a pair of public key and B will decrypt using corresponding pair of private key. The private key pair used for encryption is related to ID and hence the public key pair to be used for encryption can be identified from the ID. This is illustrated in the Table 1.

The scenario of SMOCK will be as follows. This scheme was suggested for mobile ad-hoc networks. So it has been assumed that, each node has a unique pair of private keys and all public keys before deployment. For key allocation, following steps are followed: 1) Determine x and y 2) Allocation of distinct pair of private keys to the nodes. Here “x” represents the number of key pairs used by the network, in other words, public key pairs and “y” is the number of private keys in each node.

The values of x and y can be calculated using the algorithm which has been mentioned in [5]. It can be explained as follows:

```
1) Initialize I=2
2) While (C(I, ⌊I/2⌋ ) <n))
3) I=I+1
4) x=I, y= ⌊I/2⌋
```

Here n represents the number of nodes. To illustrate, consider n=100, I=2, C(2,1)=2<100 and now I=3. The same steps are repeated. And at the eighth iteration the condition fails when I=9. So the calculated x and y values will be 9 and 4 respectively. Each node should hold 'x+y' keys i.e. for a network with 100 nodes only 13 keys must be stored in each node. This is much memory efficient when compared to traditional scheme which requies 'n+1' i.e 101 keys must be stored.

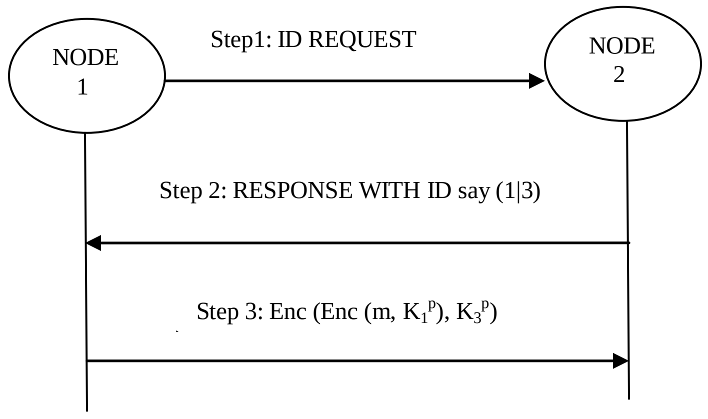
NODE 1
Step1: ID REQUEST
NODE 2
Step 2: RESPONSE WITH ID say (1|3)
Step 3: Enc (Enc (m, K_1^P), K_3^P)

Fig. 1 SMOCK Scenario

Consider an example of a small group with 10 nodes. We need 5 distinct public-private key pairs to build pairwise secure communication channels among 10 nodes, which can be (K_1^r; K_1^p), (K_2^r; K_2^p), (K_3^r; K_3^p), (K_4^r; K_4^p), (K_5^r; K_5^p). Each node keeps 5 public keys and 2 private keys. The unique private key set allocation for each node for this scenario is shown in Table 1. The ID is used while communication as shown in Fig. 1 where secure communication is done between 2 nodes.

Table 1
Private Key Allocation for 10 Nodes: An Example

USER	PRIVATE KEYS STORED AT EACH USER	NODE ID
1	$K_1^R = \{K_1^r, K_2^r\}$	1|2
2	$K_2^R = \{K_1^r, K_3^r\}$	1|3
3	$K_3^R = \{K_1^r, K_4^r\}$	1|4
4	$K_4^R = \{K_1^r, K_5^r\}$	1|5
5	$K_5^R = \{K_2^r, K_3^r\}$	2|3
6	$K_6^R = \{K_2^r, K_4^r\}$	2|4
7	$K_7^R = \{K_2^r, K_5^r\}$	2|5
8	$K_8^R = \{K_3^r, K_4^r\}$	3|4
9	$K_9^R = \{K_3^r, K_5^r\}$	3|5
10	$K_{10}^R = \{K_4^r, K_5^r\}$	4|5

Fig. 2 shows the graph which demonstrates the memory efficiency of SMOCK. This graph is constructed by calculating the number of keys to be stored by each node for different network size from 10 to 200 using the steps explained to calculate 'x' and 'y' and the same is compared with the traditional public key scheme.

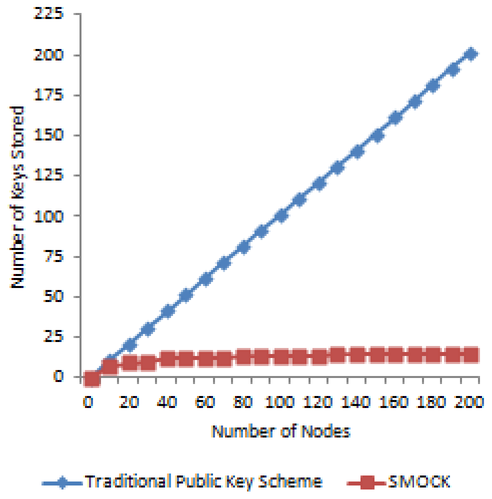

Fig. 2 Comparison of Traditional Public Schemes with SMOCK in accordance with memory usage

The above graph clearly shows that SMOCK is more memory efficient when compared to the traditional public scheme that requires 'n+1' keys to be stored.

3 ME - SMOCK

In this paper we propose an enhanced SMOCK which is meant for increasing memory efficiency and hence it can be called as ME-SMOCK, Memory Efficient-SMOCK. In this section, we explain how ME-SMOCK can be effectively used in WSNs.

In WSNs the nodes are assumed to be static after deployment. So, deployment knowledge will be useful to know about probable adjacent nodes which can help in key predistribution. This results in high memory efficiency and low cost. If the adjacent sensors are known, only the keys required to communicate with those sensors are enough to be stored in each sensor node. This will obviously decrease the memory usage.

3.1 Applying Deployment Knowledge

The point where the sensor is supposed to be deployed is called as deployment point. The point at which the sensor actually resides after deployment is called as resident point.

Let us assume that the network is divided into non-overlapping hexagonal cells. The centre of the cell (i, j) is said to be the deployment point. The deployment knowledge can be obtained by using non-uniform distribution function like Gaussian pdf. With non-uniform distribution, there is higher probability that two adjacent sensors can belong to a same cell. Using two dimensional Cartesian coordinates [6], the pdf of resident points can be calculated as follows:

$$f_k^{ij}(x,y|k \in C(i,j)) = f(x-x_i, y-y_j) = \frac{1}{2\pi\sigma^2}e^{-[(x-x_i)^2+(y-y_j)^2]/2\sigma^2}$$

where

$$f(x,y) = \frac{1}{2\pi\sigma^2}e^{-[x^2+y^2]/2\sigma^2}$$

Here, (x_i, y_i) is the deployment point of the cell C(i,j) and is the standard deviation. The typical distribution of sensor nodes belonging to the cell C(1,2) is as shown in Fig 3.

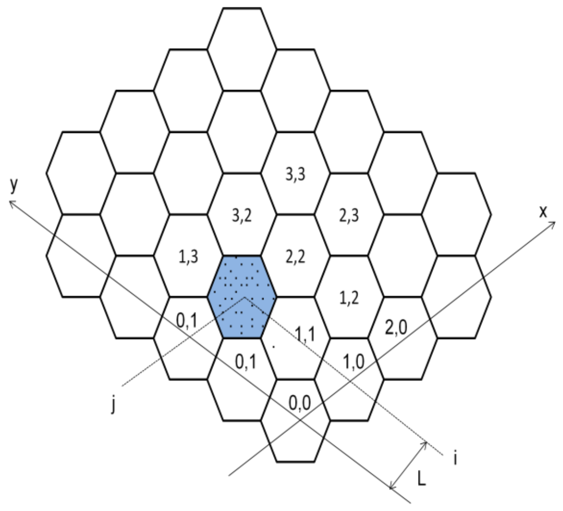

Fig. 3 Deployment Model using pdf

3.2 ME-SMOCK: An Example Scenario

The memory efficiency of ME-SMOCK can be explained through an example scenario with a sample network model that can be formed using pdf as shown in Fig. 4

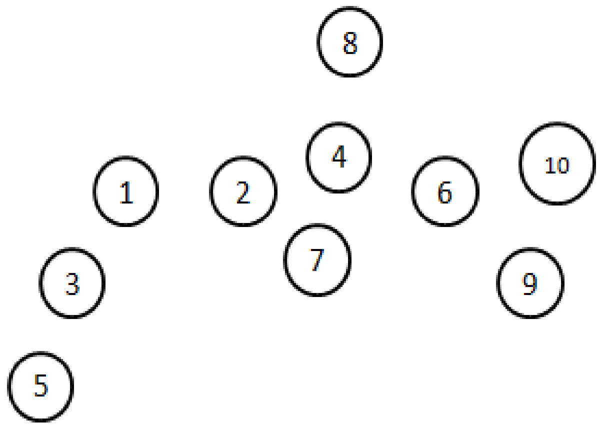

Fig. 4 Sample network model with 10 sensor nodes

Using deployment knowledge the possible adjacent neighbors can be figured out. Fig shows an example scenario of sensor nodes that are deployed using pdf in a hexagonal cell region. Using the scenario the memory usage is analysed only with 10 sensor nodes. But the same scheme suits for more number of sensor nodes also. It can be illustrated as follows.

The example scenario shown is one among the probabilities that the network model can have when using pdf to imply deployment knowledge. Based on the number of adjacent nodes, the number of keys to be stored by a particular node varies. As per SMOCK each node will encrypt using a pair of public keys. These public keys will be known from the ID they exchange. For example, node 8 has only one neighbor node i.e. node 4. So node 8 needs to store two public keys 1 and 5 (since node ID of node 4 is 1|5, refer table) and a pair of its own private key. So totally, only 4 keys must be stored in node 8. But in traditional SMOCK without deployment knowledge, 7 keys must be stored in node 8 (5 public keys and a pair of private keys). If node 2 is considered, it has three neighbors namely, 1, 4 and 7 with IDs 1|2, 1|5, 2|5 respectively. So three public keys 1, 2, 5 and two private keys are enough to be stored. The same calculation has been made and listed in Table 2.

Table 2
Number of Keys Calculated based on ME-SMOCK

Node Number	Neighbor nodes	Number of keys stored
1	2, 3	5
2	1, 4, 7	5
3	1, 5	5
4	2, 6, 7	7
5	3	4
6	4, 9, 10	6
7	1, 2, 4, 6	7
8	4	4
9	6, 10	4
10	4, 9, 10	6

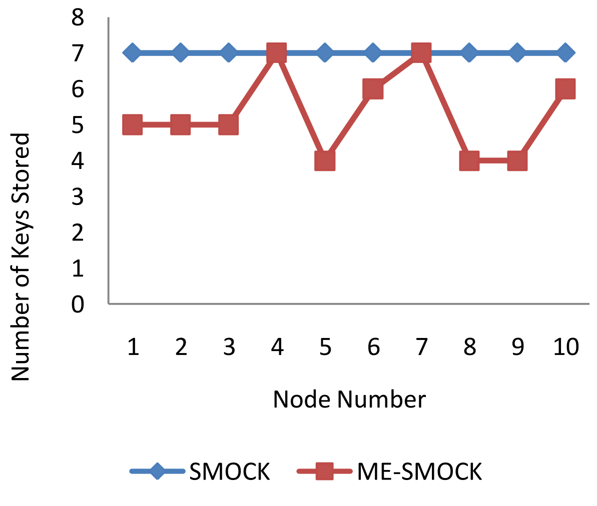

Fig. 5 Comparison of SMOCK with ME-SMOCK with respect to memory

usage

Fig. 5 shows the comparison graph of memory usage between SMOCK and ME-SMOCK. Even if there are many sensor nodes say 100 per cell, the results stay good. Because, there will be no necessity to store 100 public keys. Instead public keys of only the neighbor nodes are needed to be stored. ME-SMOCK will surely decrease the memory usage. The average number of keys to be stored by a network with n number of sensor nodes can be calculated using the formula,

$$AverageNumberOfKeys = \frac{\sum_{i=1}^{n}(AdjPub_i + PrivKeyPair_i)}{n}$$

where *n* is the network size, *AdjPubi* is the number of public keys to be stored in a node *i* to communicate with the neighbor nodes and *PrivKeyPair*$_i$ is the private key pair to be owned by node *i*.

Also it is stated that attacks can never take place by simply knowing the node ID. This is because the node ID is related to the private key pair, say, node 1 with 1|2 as ID will have the private key pair (K_1, K_2). So an attacker who tries out to receive the message from a legitimate node by giving node 1's ID as the response ID, attacker can receive the encrypted message by it cannot decrypt as the attacker will not hold the private key pair of node 1.

The evaluation made in [7] proves that, SMOCK provides scalability, control resilience and resilience to breakins with low communication overhead. So as an enhancement, ME-SMOCK adds memory efficiency.

4 Conclusion

Memory efficiency is must in WSNs, as they comprise nodes with poor resources. Hence, public schemes are not popularly used in WSNs, though it offers perfect security. Taking this challenge, ME-SMOCK has been proposed in this paper based on public key scheme SMOCK, which was proposed for ad hoc networks. Implementing the same in WSNs must concentrate more in memory consumption. To face this challenge, ME-SMOCK uses deployment knowledge to minimize the memory usage at the same time inheriting the features of SMOCK. The analysis made proves that ME-SMOCK can be efficiently used in WSNs by breaking the reluctance to the usage of public key schemes.

Acknowledgment

Our sincere thanks to the experts who have contributed towards the successful development of this paper.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, " A survey on sensor networks", *IEEE Commun,* vol.40, pp. 102-114, 2002.

[2] Yong Wang, Garhan Attebury, and Byrav Ramamurthy, "A survey of security issues in wireless sensor networks", *IEEE Commun,* vol.8, 2nd quarter 2006.

[3] S.Kent and T.Polk, "Public-key infrastructure (x.509)(pkix)charter", OnlineAvailable:http://www.ietf.org//html.charters/pkixcharter.html.

[4] L. Zhou and Z. J. Haas, "Securing Ad Hoc Network", *IEEE Network Magazine*, vol. 13, no. 6, pp. 24-30, Nov. 1999.

[5] S. Capkun, L. Buttyan, and J. P. Hubaux, "Self organized public-key management for mobile ad hoc networks," *IEEETrans. Mobile Computing*, vol. 2, no. 1, pp. 52-64, January -March 2003.

[6] W. Du *et al.*, "A Key Management Scheme for Wireless Sensor Networks using Deployment Knowledge," *Proc. IEEE INFOCOM*, Hong Kong, pp. 586–97, 2004, (SHA1)," *RFC 3174 (Informational)*, Sept. 2001.

[7] Wenbo He, Ying Huang, Ravishankar Sathyam, Klara Nahrstedt, and Whay C. Lee, "SMOCK: A Scalable method of cryptographic key management scheme for wireless ad hoc networks," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 1, pp. 140- 150, March 2009.