

A Distributed Approach to Security in Sensornets

Vijay Bhuse, Ajay Gupta, Rishi Pidva
Department of Computer Science
Western Michigan University
Kalamazoo, Michigan, USA
{vsbhuse, gupta, rpidva}@cs.wmich.edu

Abstract— Secure communication is an important aspect of any network and it has largely remained unexplored in wireless sensor networks (WSN). Security becomes a major challenge because of ad-hoc and resource constrained nature of sensor networks. In this paper we present a scalable and distributed security protocol, DSPS, for WSN that fits in between the network and the transport layers. DSPS satisfies the essential requirements of secure communication such as Data Confidentiality, Data Authentication, Data Integrity and Data Freshness. Basic building blocks of our security protocol are Key Generation-Distribution and Signatures. The key is a 56-bit random number generated initially by a key server, which is then securely distributed to all the nodes in the cluster. This key is used for encryption. DSPS also supports security critical transactions by dynamically generating a key, which can be shared between two nodes. The simplicity of DSPS allows compatibility with most of the routing protocols.

Keywords—component; wireless sensor networks (WSN); security protocols;

I. INTRODUCTION

Providing secure communication becomes a major concern when hundreds to thousands of smart sensors are envisioned to be deployed in hostile environments. Currently, these self-organizing wireless sensor networks (WSN or sensornets) are severely resource constrained (limited power, memory and processor speeds). Achieving security for these sensornets is thus a challenging task. Researchers have so far mainly focused on making WSN feasible, useful, robust and reliable. Only a few researchers have considered the secure communication aspects in WSN. Existing proposals for WSN security protocols like SPINS [1] take a centralized approach. SPINS relies heavily on a powerful base station and assumes that this base station cannot be compromised. In this paper, we propose a new security protocol, namely DSPS – a Distributed Security Protocol for Sensornets, which is distributed and does not depend on the availability of a powerful base station. Our distributed security protocol fits in between the network and the transport layers.

Basic building blocks of DSPS are Key Generation-Distribution and Signatures. A WSN is typically composed of many clusters. Each cluster of sensor nodes uses DSPS independently of each other. The proposed DSPS uses symmetric cryptographic algorithms with key distribution

within a cluster. All the nodes within a cluster share a common key and different clusters use different keys. The key is a 56-bit random number generated initially by a key server, which is then securely distributed to all the nodes of the cluster. DSPS provides the essential requirements of secure communication such as Data Confidentiality, Data Authentication, Data Integrity and Data Freshness. The simplicity of DSPS allows compatibility with most of the routing protocols.

Asymmetric key cryptography is expensive for sensornets in terms of computation and energy consumption. Whereas using efficient symmetric key cryptography is more feasible. For symmetric encryption algorithms, we mainly have three choices, DES [3], TEA [4] and RC5 [5]. The S-box and entry tables of DES at every sensor node of WSN do not currently seem feasible. While a very attractive choice, TEA has not gone through a thorough cryptanalysis yet. Hence the alternative we choose is RC5. But it is possible to attack RC5 with brute-force. As can be seen later, we can overcome this limitation by changing the key periodically. The 56-bit key used for RC5 is common to the whole cluster and is generated initially by the key server (i.e., one of the sensor nodes). This key is then distributed to all the nodes of the cluster in a secure way. We define an epoch to be the duration for which a particular key is used. We use a Hughes's variant of Diffie-Hellman algorithm [6] with Encrypted Key Exchange [7] (HDH-EKE) for propagating the key at the start of an epoch. At the end of an epoch a successive key will be generated on all the nodes using a one-way function [8]. So DSPS runs HDH-EKE only at the start when sensor nodes are deployed and network is formed.

Rest of the paper is organized as follows. In section II, we discuss our assumptions that guide the design of DSPS to provide various security features. Sections III and IV give a brief outline of the DSPS. Building blocks and more details of DSPS are given in section V. In section VI we list advantages of DSPS and then compare it with SPINS in section VII. Due to space limitations, discussion of related work other than SPINS is omitted from this paper. Reader is referred to [10, 11, 12, 13, 14, 15] for details.

II. ASSUMPTIONS

Wireless communication is fundamentally broadcast in nature so the DSPS design guards against an adversary who can eavesdrop on the traffic; inject new messages; alter messages in transit or replay old messages. DSPS does not trust

communication infrastructure. But we assume that network layer is reliable.

Security should not be overkill. In order to make communication faster and cheaper or to facilitate specific application development, one may assume a specific communication pattern. However, to design a generic security protocol, it is not desirable to assume any specific communication pattern, e.g. a general communication pattern of node to base-station and an occasional node-to-node communication as in SPINS. (We do note that mainly a node to base-station communication assumption allows certain advantages in SPINS over DSPS. We will discuss that later in sections VI and VII. Furthermore, exploiting specific communication patterns may provide additional security features, however, that is beyond the scope of this paper). Directed Diffusion [9] enables energy-efficient and robust data dissemination in dynamic sensor networks. Its event driven dynamic path discovery requires efficient and quick local communication. For local node-to-node communication contacting base station most of the time is not always desirable or does not always seem feasible especially when sensor nodes are to collaborate together and engage in in-network processing. It may deplete the energy of the intermediate nodes on the path to a base station if routing is not strongly energy aware or not application-specific, which will ultimately decrease the lifetime of the network. There is also a risk of network partition. Therefore one would not like, in general, to assume any specific communication pattern. DSPS design thus attempts to assume no specific communication pattern.

Since WSN is a distributed system and due to envisioned deployments of WSNs, synchronizing sensor node clocks will be a costly affair. An algorithm that depends on time synchronization is prone to failure in case there is a significant clock skew. Hence we do not assume sensor nodes to be time-synchronized.

With large-scale deployment of the tiny resource-constrained nodes, classification and clustering based on geographical boundaries or their functionality is inevitable. So we assume that a network is formed of one or more clusters.

There are various kinds of attacks one can imagine. We say a node is physically compromised when someone has physical access to it and they can extract data from it. This is one of the most dangerous attacks and nobody can prevent this unless sensor nodes are, for example, self-destroying (via hardware or software when tampered with). Furthermore, in order for new sensor nodes to join the network or to support mobility, some sort of trust has to be placed in the valid sensor nodes; e.g. assuming availability of a confidential information which is known only to designers of the system and if a sensor node gets physically compromised then this information is destroyed during the breach of trust. This question needs much fundamental research and is beyond the scope of this paper. On the other hand, to make any progress, we need to place some trust somewhere. We thus assume that sensor nodes are not physically compromised and if they are then they self-destroy (hardware or software-wise).

Since a key server generates a new key at the end of every epoch, we assume that a key server is not compromised during this time period.

All the nodes have Diffie-Hellman constants g and n hardwired in them. These constants can be public. The nodes also have a hardwired password P that is a secret (the confidential information as discussed earlier). A physically compromised node poses a threat to a cluster if it has a shared secret. As mentioned earlier, if new nodes are to join a network dynamically then there need to be some shared secret for authentication. P is such secret. We take care of protecting P by never using it directly in any communication operation.

Although RC5 is tested and found to be quite robust we assume that it takes some finite amount of time T to crack a cryptographic algorithm like RC5. The time to replace old key will depend on T . An epoch is a time interval for which a key is in use in DSPS and is directly proportional to T . Random keys used for RC5 along with the key server are changed for the next epoch.

III. NOTATIONS

We use the following notations to describe DSPS in this paper.

$E_K(R)$ denotes an encryption of message R with key K .

$D_K(Q)$ denotes message obtained by decrypting Q with key K . We have $D_K(E_K(R))=R$ because of symmetric key cryptography.

$S | T$ denotes concatenation of messages S and T .

$\%$ denotes modulus operation.

K_n denotes Key for n^{th} epoch.

$F(P, Q)$ is a one-way function where P and Q are integers.

IV. OUTLINE OF DSPS

Each cluster has a designated key server. The key server generates a random key and distributes it securely to all the nodes in the cluster. All the nodes in a cluster use the same key to encrypt or decrypt messages. At the start of a new epoch the key server initiates generation and distribution of a new key. DSPS changes keys after an epoch expires. Epoch depends on the strength of a cryptographic algorithm. The key server also keeps track of time. The key server can be rotated so that the network lifetime is increased.

For our discussion, DSPS uses RC5 as a cryptographic algorithm (one can replace RC5 with any other reasonable symmetric key cryptography algorithm). All the messages are encrypted using RC5, which provides data confidentiality.

There can be some nodes, which may be on the boundary of two or more clusters. Those nodes will have keys of all the clusters to which they belong. While sending packet from cluster 1 to cluster 2 they first decrypt a message with the key of cluster 1, encrypt with the key of cluster 2 and forward it. (See Figure 1)

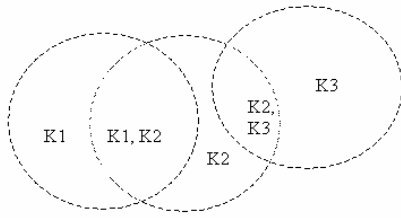


Figure 1. Different clusters have different keys. The nodes common to two or more clusters have keys of all the clusters to which they belong.

Each packet has a unique encrypted Signature attached with data. Signature provides data authentication, data integrity and data freshness.

DSPS supports intercluster and intracluster movement of nodes. DSPS also allows new nodes to join network.

A compromised key will compromise the entire cluster till a new key is set up. We next describe the details of DSPS.

V. DSPS BUILDING BLOCKS

A. Key Generation

The current key server will generate a key. All the nodes have Diffie-Hellman constants g and n hardwired in them, which could be public. They also have a shared secret password P . Initially when network is deployed the key server generates a random number x and computes the key $K = g^x \% n$.

The key server will keep track of time to find when an epoch expires. After expiration of an epoch a new key is to be generated. It will send a broadcast message asking nodes to compute new key using one-way function. In a broadcast message it will send a random number *Counter*. Counter along with P will be the arguments to a one-way function F which will produce password $P' = F(\text{Counter}, P)$. P' will be used only once. New key K' is generated from P' , the previous key K and one-way function F as $K' = F(K, P')$.

New keys may be generated on different nodes at slightly different global time. There may be a case that two neighboring nodes are using different keys at the same time. DSPS addresses this issue by sending the current epoch number with every packet. So if node A is using key K_{10} , B will send a message to A that it is using an outdated key. Then A has to join the cluster again as a new node and resume communications with B. New nodes can securely join a network as explained later. If a node using K_{10} sends a packet to a node using K_5 , the receiving node will come to know that it is using an outdated key. So it will join the network again.

B. Key Distribution

The key server generates the key to be used by all the nodes in the cluster. Care must be taken while the key is distributed among all the nodes in the cluster. Hughes's variant of Diffie-Hellman algorithm plays an important role here. Hughes's variant of Diffie-Hellman algorithm is susceptible to Man-in-the-middle attack. So we use Hughes's variant of Diffie-

Hellman algorithm with Encrypted Key Exchange (HDH-EKE). The key server will generate and broadcast a random number *Counter*. All the nodes will generate a one time password P' using one-way function F , i.e., $P' = F(\text{Counter}, P)$. This way we protect P and never directly use P .

The key server will initiate the HDH-EKE individually with all its neighbors. The neighboring nodes will carry on the process and will establish communication with other nodes. This process will continue until all the nodes in the cluster have established one to one communication with some neighboring node. This way every node will have the key, which is common to the cluster. After key distribution the nodes can use this key for encryption or decryption (symmetric key) using RC5 algorithm.

Figure 2 shows how HDH-EKE protocol works. Suppose A is a key server and it wants to pass the key to node B.

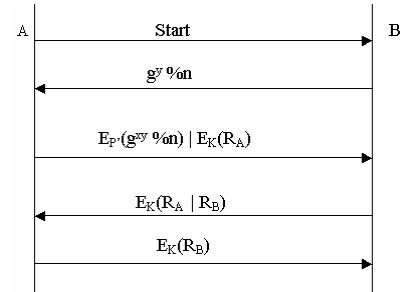


Figure 2. Hughes's variant of Diffie-Hellman algorithm with Encrypted Key Exchange. Key server A generates the key K and passes it to sensor node B securely

1. A will generate a random number x . It will compute the key $K = g^x \% n$ and send a start message to B.
2. B will generate a random number y . It will compute the number $g^y \% n$. It will send this number unencrypted to A.
3. After receiving $g^y \% n$ from B, A will compute $g^{xy} \% n$. It will encrypt that number with P' . It will generate a random number R_A and will encrypt it with K . It will concatenate these two encrypted numbers $(E_{P'}(g^{xy} \% n) | E_K(R_A))$ and will send to B.
4. Since B has P' it can decrypt $E_{P'}(g^{xy} \% n)$. It computes the key $K = g^x \% n$. K is the key that A and B will use for encryption and decryption. After computing K it can decrypt $E_K(R_A)$. Then it generates a random number R_B and sends $E_K(R_A | R_B)$ to A.
5. A will decrypt the message $E_K(R_A | R_B)$ and will send $E_K(R_B)$ to B.
6. B will receive $E_K(R_B)$ and decrypt it to check against its local R_B . HDH-EKE protocol is now complete.

HDH-EKE will survive Man-in-the-middle attack. The purpose of random numbers R_A and R_B is to provide 2-way authentication. It takes 5 messages to transmit a key to one node. For n nodes it takes $\Theta(n)$ messages. Key distribution takes place only initially when nodes are deployed. After that all the subsequent keys are generated locally.

C. Signatures

Encryption using the common key K ensures Data Confidentiality. With HDH-EKE we distribute K securely which is used for encryption and decryption. For a secure communication only encryption is not sufficient. Another building block of DSPS, Signatures provides us a secure channel of communication.

Every packet has a signature consisting of five fields: *source-id*, *receiver-id*, *session*, *counter* and *CRC*.

Signature = (source-id, receiver-id, session, counter, CRC)

Signature is also encrypted along with the data using common key K . A signature can be used for both unicast and broadcast packets. The purpose of signature is to distinguish packets from one another for an entire lifetime of the network. If two nodes are communicating i.e. they are sending messages back and forth, we call that as a session. In one session there can be one or more packets sent. A counter is a packet index in a particular session. Counter distinguishes packets in the same session. The sender will start a session with a counter value 1 and will end it with a counter value -1. If there is only one packet in a session then the counter value will be -1. Each node will have one table. Each record in the table has five fields, which are node-id, unicast session, unicast counter, broadcast session and broadcast counter. See table 1 for an example of the table at node B and what these fields imply.

Node-id	Unicast Session	Unicast Counter	Broadcast Session	Broadcast Counter
C	1	0	2	0
D	2	0	1	2
E	6	12	0	0
F	9	23	0	0

TABLE I. B'S TABLE SHOWS THAT IT HAD UNICAST SESSIONS WITH C, D, E, F AND BROADCAST SESSIONS WITH C, D. AT PRESENT B IS HAVING UNICAST SESSION WITH E, F AND BROADCAST SESSION WITH D.

A receiver keeps track of source-ids and corresponding counters for a particular session. When a new packet arrives the receiver will decrypt and check the signature of the packet. If the receiver-id in the signature matches with its own id (if ids don't match then it will drop the packet) then it will check the source-id in the signature against its stored source-ids. If it does not find an entry for that source then it will add the source-id to the table. If it finds an entry then it will check the counter, which should be smaller than the counter in the signature. If so it will replace the counter with the counter in the signature. If the counter is -1 then the node will reset the counter for the corresponding entry.

We next describe various scenarios of WSN that are addressed by DSPS.

D. Critical Transactions

If two nodes want to perform some critical transaction then one of them will generate the key. It will then pass the key to the other node using HDH-EKE. The key will be used only for

that transaction and will be discarded after the critical transaction is performed.

E. New Node Can Join

For a new node to join it must have the trusted secret password P . The new node will send a request to any node in a cluster that it wants to join. The node in the cluster will send a random number Counter to the new node. The new node can compute P' using one-way function as $P' = F(\text{Counter}, P)$. Now they have a common number P' . So they can perform HDH-EKE as explained above. The new node will get the current key and will become a part of the cluster and thus the WSN.

F. Intercluster Movement of Nodes

DSPS supports intercluster movement of nodes. The nodes can freely move within a cluster as they have the common key K . Hence, intercluster movement requires no extra work for security.

G. Intracluster Movement of Nodes

DSPS supports intracluster movement of nodes. When a node leaves cluster $C1$ it will delete the key of the cluster $C1$. To join cluster $C2$ it needs the key of cluster $C2$. So it is like a new node joining a cluster. DSPS supports joining of new node provided it has the trusted secret password P .

H. Nodes Common to two or more Clusters

DSPS supports nodes that are common to two or more clusters. Those nodes will have keys of all the clusters they belong. Key change may occur at slightly different time in neighboring clusters. Before forwarding a packet from cluster $C1$ to cluster $C2$, sensor nodes will first decrypt packet with key of $C1$ and then encrypt with key of $C2$.

I. Packets floating during Key Distribution

The nodes will buffer these packets. They will decrypt with the previous key, encrypt with the new key and will forward it.

The above building blocks of the proposed DSPS protocol achieve the required security properties of WSN as follows:

- *Data Confidentiality*: The use of HDH-EKE during the setup phase and use of a robust symmetric key cryptographic algorithm (such as RC5) makes sure that there is a secure communication among the nodes in the cluster.
- *Data Authentication*: The encrypted signature with the data has a source-id, which guarantees authentication. DSPS does not provide broadcast authentication. Since DSPS relies on changing keys of a cryptographic algorithm periodically, adversary cannot become a part of a network.
- *Data Integrity*: The CRC in the signature makes sure that the message is not modified. Since the signature is encrypted it assures data integrity.
- *Data Freshness*: The session and counter in the signature together contribute weak data freshness. The

receiver anticipates the session and counter in the signature of the incoming packet. If the session is higher than the current one then weak freshness is obtained. If the session is same as the current one then, it will check the counter. A greater counter value in the signature ensures weak data freshness. DSPS also assures that an appropriate receiver receives the packet. This prevents some kind of Denial of Service [8] attacks.

VI. ADVANTAGES OF DSPS

The following are some of the advantages of DSPS:

- Clock or time synchronization is not required.
- It does not assume any specific communication pattern (for e.g. node to base station). So it will work efficiently with any routing protocol.
- It will perform better when there is more node-to-node interaction. So it supports collaboration among nodes more easily.
- Easy to implement and change according to the application.
- Supports intercluster and intracluster movement of nodes.
- New nodes can securely join the sensornet.
- With periodic resetting of keys, an adversary cannot become part of a network permanently, i.e. if at all possible, security breach is localized within a cluster and after the current epoch expires, an adversary can be detected during the key setup phase.

VII. COMPARISON WITH SPINS

DSPS differs from SPINS in the following ways:

1. SPINS assumes some specific communication patterns, whereas DSPS does not.
2. SPINS needs nodes to be loosely time synchronized. DSPS does not require time synchronization.
3. DSPS uses same encryption key for an entire cluster.
4. A compromised key will compromise the whole cluster till the next key is set up in case of DSPS. In SPINS a compromised key only compromises one node.
5. DSPS allows periodic resetting of keys. So an adversary cannot become part of a network permanently. SPINS does not allow that.
6. SPINS provides broadcast authentication whereas DSPS does not.

VIII. CONCLUSIONS

We designed a security protocol for wireless sensor networks, which is distributed in nature. It does not assume any communication pattern. It does not require nodes to be time-

synchronized. It supports node mobility and scalability. DSPS is easy to implement and modifiable depending on the application and it will work efficiently with any routing protocol. Our future work involves efficient and optimized implementations of DSPS on sensornets of Berkeley motes [2] using TinyOS and NesC and other sensornet platforms.

ACKNOWLEDGMENTS

We thank Junaith Ahmed for discussions on the initial designs of DSPS. Funding for this research was received from a Congressional Award, administered by the US Department of Education, Fund for the Improvement of Education (R215K020362), 2003.

REFERENCES

- [1] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. Culler, "SPINS: Security Protocols for Sensor Networks", *Wireless Networks*, 8:521-534, 2002.
- [2] Networked Embedded Systems Technology research at University of California, Berkeley [Online], Available: <http://webs.cs.berkeley.edu/nest-index.html> and http://www.xbow.com/Products/Wireless_Sensor_Networks.html.
- [3] National Institute of Standards and Technology, "Data Encryption Standard (DES)," U.S. Department Of Commerce, U.S.A, 1999.
- [4] David Wheeler and Roger Needham, TEA, a tiny encryption algorithm.[Online], Available: <http://www.ftpl.cam.ac.uk/ftp/papers/djw-rmn/djw-rmn-tea.html> , November 1994.
- [5] R. L. Rivest, "The RC5 encryption algorithm", *Proc. 1st Workshop on Fast Software Encryption*, pages 86-96, 1995.
- [6] W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, IT-22:644-654, 1976.
- [7] B. Schneier, *Applied Cryptography*, 2nd Edition, John Wiley & Sons, 1996, pp:518-520
- [8] B. Schneier, *Applied Cryptography*, 2nd Edition, John Wiley & Sons, 1996, Chapter 18
- [9] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, August 2000, Boston, Massachusetts.
- [10] C. Karlof, N. Sastry and D. Wagner , TinySec: Link Layer Security for Tiny Devices [Online], Available: www.cs.berkeley.edu/~nks/tinysec/
- [11] L. Eschenauer, V. D. Gligor, "A key-management scheme for distributed sensor networks", In *Proceedings of the 9th ACM conference on Computer and communications security 2002*, Washington, DC, USA, pp: 41-47.
- [12] Y.W. Law, S. Dulman, S. Etalle and P. Havinga, "Assessing Security-Critical Energy-Efficient Sensor Networks" , Department of Computer Science, University of Twente, Technical Report TR-CTIT-02-18, Jul 2002.
- [13] Jeffery Undercoffer, Sasikanth Avancha, Anupam Joshi, and John Pinkston, "Security for Sensor Networks", 2002 CADIP Research Symposium.
- [14] S. Slijepcevic, M. Potkonjak, V. Tsatsis, S. Zimbeck, M. B. Srivastava, "On communication Security in Wireless Ad-Hoc Sensor Network", In *Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02)* June 10 - 12, 2002 Pittsburgh, Pennsylvania, USA.
- [15] L. Yuan, G. Qu, "Design Space Exploration for Energy-Efficient Secure Sensor Network", *The IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP'02)* July 17 - 19, 2002 San Jose, California.