SAKE
Secure Authenticated Key Establishment in Sensor Networks

Muhammad Yasir, Mureed Hussain
SZAB Institute of Science and Technology, Islamabad, Pakistan

Kahina Kabri, Dominique Seret
Department of Mathematics et Informatics, University of Paris5, France

Keywords: SAKE, Authentication, MANETs security, Sensor networks.

Abstract: Master key schemes are a viable solution to establish pairwise shared secret keys in wireless sensor networks. In these schemes, a master key is preconfigured into each sensor node which is then used by each node to generate pairwise shared secret keys. In the literature so far, it is essential for each sensor node to keep master key in its memory during the entire phase of key setup. As soon as key setup completes, each node erases the master key from its memory. Although key setup phase of a node lasts for a small interval of time, it is not impossible for an adversary to compromise a node during this time. In this situation, the presence of master key can be disastrous. So the challenge is to protect a sensor network from compromise of master key during its key setup phase. We propose Secure Authenticated Key Establishment (*SAKE*) protocol that meets the above challenge by introducing an idea that master key need not to be kept by a sensor node for the entire key setup phase thereby shortening the master key compromise window. With the help of our proposed scheme, other attacks during key setup phase can also be avoided.

1 INTRODUCTION

Wireless Sensor Networks are self-organizing networks of locally communicating sensor nodes having limited computation, memory, and energy resources. Due to the deployment of sensor nodes in large numbers, the nodes are made inexpensive; thereby not tamper-resistant and an adversary is quite capable of compromising some of them. Above all, the sensor nodes with limited resources are not capable of using public key cryptography because it is computationally intensive. All of these inherent unique characteristics make implementation of security in wireless sensor networks far more challenging than in traditional computer networks. Key establishment lays the foundation for implementing security in a sensor network, in which a sensor node establishes secure links with its neighbours when it is deployed in a sensor network. In fact it establishes the keys necessary to provide confidentiality, integrity and authentication services. Due to the resource limitations of sensor nodes, key establishment protocols for sensor networks are based on symmetric key algorithms. Key establishment for wireless sensor networks must be lightweight, secure and efficient.

Camtepe et *al.* (S. A. Camtepe and B. Yener, 2005) described following specific security requirements of wireless sensor networks along with availability, authentication, integrity, confidentiality, and non-reputation.

- *Survivability:* capability of providing minimum level of service in the presence of power loss, failures or attacks.
- *Degradation of security services:* capability of changing security level with the change in resource availability.

Pre-deployed keying is the most suitable solution for bootstrapping secret keys in sensor networks. In pre-deployed keying, sensor nodes are loaded with keys before their deployment. Several solutions based on pre-deployed keying are proposed. Master key schemes ((J. Deng et al.), (S. Zhu et al., 2003), (S.Seys), (B. Lai et al., 2002), (A. Perrig et al., 2001), (B. Dutertre et al., 2004)), are also based on pre-deployed keying in which each

Yasir M., Hussain M., Kabri K. and Seret D. (2008).
SAKE - Secure Authenticated Key Establishment in Sensor Networks.
In *Proceedings of the International Conference on Security and Cryptography*, pages 154-160
DOI: 10.5220/0001930301540160

sensor node is preconfigured with the same master key and that master key is then used by each node to generate pairwise keys for sharing with each of its neighbors. After the key setup phase, each node removes the master key from its memory. Key establishment techniques are evaluated on the basis of following metrics (S. A. Camtepe and B. Yener, 2005):

- Scalability: Key establishment technique should be flexible against significant increase in the size of the network even after deployment.
- Efficiency: Key establishment technique must be efficient in terms of storage (required memory to store security credentials), processing (amount of processor cycles required to establish a key) and communication (number of messages exchanged during key generation process).
- Resilience: Resilience against node capture and security credentials compromise stored in a node or exchanged over wireless radio links.

2 RELATED WORK

Perrig et *al.* (A. Perrig et al., 2001) proposed Security Protocols for Sensor Networks namely ***SPINS***. It consists of two protocols, SNEP and µTESLA. SNEP provides confidentiality, two-party data authentication, integrity, and freshness. µTESLA supports authenticated broadcast. Figure 1 shows the key establishment protocol for ***SPINS*** (B. Lai et al., 2002), It shows that when a node **A** wants to establish a pairwise shared key $\mathbf{SK_{AB}}$ with node **B** with the help of a trusted third party server **S,** which acts as authentication and key distribution server.

Key establishment protocol works as:

1. Node **A** sends a request message to node **B**.
2. Upon receiving this message, Node **B** sends a message to the **S**
3. Key server **S** authenticates and generates the pairwise key and sends the key back to node **A**
4. Key server then sends pairwise key to node **B**.

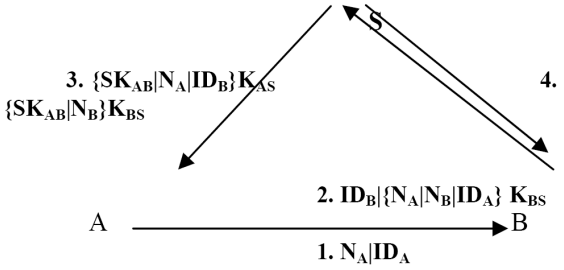

Figure 1: Key establishment in ***SPINS*** (A. Perrig et al., 2001).

The drawback of ***SPINS*** is the use of a central key server **S**. As a sensor network comprises a large number of nodes, the use of central key server restricts the scalability of the sensor networks.

Seys et *al*. (S.Seys) proposed ***SNAKE***, a protocol for key establishment based on an ad-hoc scheme. In contrast to ***SPINS***, in this scheme, no key server is involved in key establishment. Figure 2 describes the protocol.

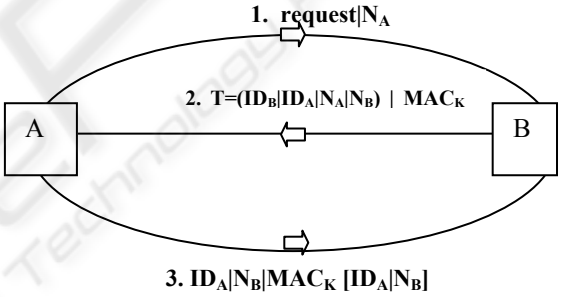

Figure 2: ***SNAKE*** (S.Seys).

Key establishment in ***SNAKE*** is done in the following way.

1. Node **A** sends a request and a nonce to node **B**
2. Node **B** returns **T** (the identity and nonce of node **A**, **B**'s own identity and nonce), concatenated with a Message Authentication Code (MAC) of **T** to **A**.
3. Upon receiving this message, node **A** proves its authenticity and sends the message back to node **B**.

After this process, node **A** and node **B** generate pairwise shared key as: $\mathbf{K_{AB} = MAC_K\ [N_A|N_B]}$

Lai et *al.* (B. Lai et al., 2002), proposed ***BROSK*** in which each node broadcasts the key negotiation message to establish the pairwise shared key with its neighbors. To establish session keys with its neighbors, a sensor node **A** broadcasts the following message:

$$\mathbf{A \longrightarrow *: ID_A|N_A|MAC_K\ (ID_A|N_A)}$$

K is the same master key preconfigured in all the nodes prior to deployment. Upon receiving A's broadcast, a node B can construct the pairwise shared key $\mathbf{K_{AB}}$ by generating the Message Authentication Code (MAC) with the help of two nonces as:

$$\mathbf{K_{AB} = MAC_K (N_A|N_B)}$$

Similarly node **A** also receives the broadcast message from node **B** and constructs the key $\mathbf{K_{AB}}$ in the similar manner.

Zhu et *al.* (S. Zhu, 2003) proposed Localized Encryption and Authentication Protocols (***LEAP***) based on master key scheme. In ***LEAP***, an initial key k_I is preconfigured into each node. Each node u derives its master key as:

$k_u = f(k_I, u)$, where f is a secure one-way function. A node u establishes pairwise keys by broadcasting its identity as:

$$u \longrightarrow *: \quad u$$

After broadcasting node u waits for its neighbor's response. A node v sends the response as:

$$v \longrightarrow u: v, \mathbf{MAC}(k_v, u|v)$$

Upon receiving the response, Node u derives k_v as: $k_v = f(k_I, v)$ and authenticates the response using the derived key. The pairwise key for nodes u and v is $f(k_{u,v})$, for $u > v$, and $f(k_{v,u})$ otherwise. Each node erases the key k_I from its memory after the key setup phase but retain its own master or individual key.

The initial key k_I in ***LEAP*** is the single point of failure. If an adversary is somehow ever able to obtain k_I before it is erased, she will be able to compute all previously setup pairwise keys in the network as well as calculate all future session keys that may be established. In addition, the adversary can also inject any number of malicious nodes into the network.

Deng et *al.* (J. Deng et al.), described Opaque Transitory Master Key Scheme (***OTMK***), a pairwise key establishment scheme in which a master key M is preconfigured into each sensor node. To establish session keys, each node u broadcasts a request message as:

$$u \longrightarrow *: \quad JOIN|E_M(ID_u|nonce)$$

ID_u is the identity of u. Upon receiving this broadcast, a node v generates a random number $k_{v,u}$ and responds u as:

$$v \longrightarrow u: REPLY|E_M(ID_v|nonce+1|k_{v,u})$$

After receiving this message node u performs verification of nonce and designates node v as its verified neighbor. The pairwise key is either $k_{v,u}$ generated by v or $k_{u,v}$ generated by u. $k_{u,v}$ is used as shared key if $ID_u < ID_v$ else $k_{v,u}$ is used as shared key.

To enable new nodes to join, Deng et *al.* (J. Deng et al.), proposed another scheme in which a node v generates a new key $k_v = MAC(M, ID_v)$ and a number of verifiers containing two random numbers r_i and y_i where $y_i = f(M, r_i)$. A node v stores many verifiers and erases the master key M. When a new node u arrives, node v sends one of its random numbers r_i as a challenge to node u. Node u computes $z_i = f(M, r_i)$ and sends it to node v. Now node v compares z_i with y_i and if both are equal then it verifies node u. After that the pairwise key is established between u and v. A critical question is how many verifiers are to be contained by a node. A sensor node with a limited memory resource cannot store too many verifiers for authentication of newly joining nodes.

Dutertre et *al.* (B. Dutertre et al., 2004), depicted that many timing mechanisms can be used to reduce the probability of message collisions and proposed a protocol in which a secret group authentication key b_{k1} and a key b_{k2} to generate session key is preconfigured into all nodes. To establish pairwise keys a node **A** broadcasts a hello message as:

$$\mathbf{A} \longrightarrow *: \mathbf{HELLO|A|N_A|MAC}_{bk1}\mathbf{(HELLO, A, N_A)}$$

Upon receiving, a node **B** checks the validity of **A** and responds to **A** as:

$$\mathbf{ACK|A|B|N_B|MAC}_{bk1}\mathbf{(ACK, A, B, N_B, N_A)}$$

This acknowledgement proves to **A** that **B** knows b_{k1} and has received $\mathbf{N_A}$. After that **A** and **B** establish keys as $K_{AB} = \mathbf{G_{bk2}(N_A, N_B)}$. **G** is a keyed one-way hash function.

Perrig et *al.* (R. Anderson et al., 2004) paid attention to key distribution in commodity sensor networks where they did not assume a *global passive adversary* and proposed ***Key Infection***: a plaintext key exchange protocol, in which every sensor node sends plaintext to establish pair-wise keys with each of its neighbor nodes. In this scheme it was assumed that during the network deployment phase, the attacker can monitor only a fixed percentage a of communication channels. This assumption was in contrast to the previous work on key distribution for sensor networks, in which a strong threat model was assumed: it was assumed that the adversary is present both before and after the deployment of nodes, and can monitor all communications in the network at all times. It is assumed also that it is possible for an adversary to maliciously reprogram a small number of sensor

nodes. If this key setup time completes in short time, an adversary has very little time to eavesdrop on key setup. Compromising a node does not offer the adversary any added advantage in deducing these keys. ***Key Infection*** establishes key as: every sensor node simply decides a key and broadcasts it in plaintext to its neighbors. For example a node ***i***, after deployment, broadcasts a key k_i. Due to short-range nature of transmission, possibly half a dozen other nodes within range of ***i***, notice each other's presence and start self-organization. Another node ***j*** after hearing ***i***'s signal, produces a pair wise key k_j and sends it, along with its name, to ***i***:{ ***j***, k_{ji}} k_i. Minimum power essential for the link is used to transmit the packet. The key k_{ji} is used between ***i*** and ***j***. Perrig et *al.* argued that only 2.4% of links will be compromised if there is one hostile sensor node for every 100 white nodes, and there are four neighbors in the range of each node. Plaintext key exchange protocol does not provide protection for confidentiality, integrity, and node authentication. Another drawback of this approach is that an adversary can inject malicious nodes into the network, since there is no authentication mechanism to verify whether a sensor node is a valid member.

In *key pre-distribution* schemes, secret keys or secret information is distributed to every sensor node prior to deployment into the sensing area. Gligor et *al.* (L. Eschenaur and V. Gligor, 2002) proposed the probabilistic key pre-deployed scheme, which is regarded as basic scheme. In this scheme, key setup completes in three phases: key pre-distribution, shared-key discovery, and path-key establishment respectively.

Figure 4 shows the key pre-distribution phase in which each sensor node holds k distinct keys, randomly chosen from a big key pool having size P where P>> k. This set of k keys carried by each node is called key ring. An identifier is attached with each key. *Shared key discovery phase* starts with the deployment of nodes in which each node discovers its neighbors to share common keys in its radio range. Links are established between the nodes at the end of shared-key discovery phase. *Path-key establishment phase* is the last phase in which pairs of nodes are connected that want to establish a secure link but they are not sharing a common key. These pairs of nodes can be reachable by two or more hops. Perrig et *al.* (H. Chan et al., 2003) modified Gligor's scheme for proficient handling of bootstrapping problem. This scheme is more resilient against node capture. In this scheme, instead of using one common key for key establishment, q common keys are required for key setup with a hash function. Figure 5 demonstrates this scheme (B. Lai, 2002).

It is possible that many nodes in the sensor network can share that same key. If one of these nodes is compromised, all other nodes sharing the same key will also be compromised. Du et *al.* (W. Du et al., 2004) has proposed a model, in which the sensor nodes are deployed in groups, so in each group the nodes have high probability to be near to each other. So the basic idea is to let the nodes deployed near to each other select keys from sub-key pools that share more keys. In the scheme, because each node carries fewer keys, the memory efficiency and resilience are both improved.

3 SAKE: SECURE AUTHENTICATED KEY ESTABLISHMENT

In key establishment techniques based on master key schemes, master key is a single point of failure. Deng et *al*. (J. Deng et al.), demonstrated that time to deploy a sensor network is very important. The key setup phase may need to be extended until all nodes are activated, or arrive at their destination. In such cases, the master key would live for several minutes. If an adversary knows the memory area precisely from which to draw the keys, then the node compromise time is in the tens of seconds. So it is quite possible for an adversary to physically capture a node and compromise the master key during its key setup phase. The challenge in this scenario is to protect the network from its single point of failure. This challenge can be met if key establishment process would be made less dependent on master key. This can be done if the individual key of each sensor node would be made an active participant in key establishment process along with the master key. This load balancing approach will protect the master key from becoming a single point of failure. Moreover, in a situation when master key is compromised, it is required that a typical key establishment technique would not enable an adversary to compute all pairwise keys with the help of master key.

We propose Secure Authentication Key Establishment (***SAKE***) protocol, which is based on master key scheme. ***SAKE*** supports the establishment of pairwise shared keys between the nodes of a sensor network. ***SAKE*** is dissimilar from other master key schemes as it makes key establishment less dependent on master key. Firstly,

it is not essential for sensor nodes running ***SAKE*** to keep the master key in the memory for the entire key setup phase. In fact a node running ***SAKE*** erases the master key very early during its key establishment process. This reduces the chance of master key compromise if an adversary physically captures a node during its key setup phase. It also makes the insertion of malicious nodes in the network nearly impossible. Secondly, in case of master key compromise, an adversary cannot compute all pairwise keys in the network with the help of master key in contrast to ***LEAP*** protocol proposed by Zhu et *al.* (S. Zhu, 2003) in which an attacker can calculate all previously setup pairwise keys as well as is able to compute all pairwise keys that may be established in future.

3.1 Assumptions

It is assumed that the sensor nodes are static and a sensor node does not know its immediate neighbors in advance. T_{min} is a certain amount of time an adversary requires to compromise a node. T_{key} is the total key setup time of a sensor node and it may be less or greater than T_{min}. This is in contrast to other master key schemes ((J. Deng), (S. Zhu, 2003)), where T_{key} is assumed to be always less than T_{min}. T_{key} is divided into two time intervals namely T_{start} and T_{estab}. T_{start} is the smaller portion of T_{key} that at maximum is less than half of T_{key} so it is smaller than T_{estab}. It is assumed that being a very small portion of total key setup time; T_{start} is always less than T_{min}. In other words it is believed that a node that requires T_{key} time to complete its key setup process cannot be compromised in its T_{start} phase.

Following notations have been used in the presentation of the protocol

- ***a*** and ***b*** are communicating sensor nodes.
- Ek_m **(M)** means encrypting message **M** with master key k_m.
- Ek_s **(M)** means encrypting message **M** with subordinate key k_s.
- **x**|**y** means concatenation of message **x** with **y**.

Since communication is taking place between a typical sensor node and its immediate neighbors, our scheme establishes pairwise keys that are shared by a node and its immediate neighbors.

In this scheme, each sensor node is preconfigured with a master key k_m and subordinate key k_s. Moreover each sensor node has also its unique individual key. A node keeps the master key k_m during its T_{start} phase only. A node erases k_m when T_{start} expires but retains k_s for entire key setup phase.

3.2 Functional Architecture

Key setup process of a sensor node starts when a node is deployed. Initially the node ***a*** is in its T_{start} phase in which it performs following tasks.

- Encrypts its individual key k_a with both k_m and k_s.

$$y_1 = Ek_m\ (a|k_a)$$
$$y_2 = Ek_s\ (a|k_a)$$

- Broadcasts a ***HELLO*** message to discover its immediate neighbors.

$$a \longrightarrow *: HELLO|a|n_a|\ Ek_m\ (a|n_a|k_a) \quad (1)$$

a is the identity of node ***a***. n_a is the nonce or random number and k_a is the unique individual key of ***a***. T_{start} interval is made so small that a node ***a*** can perform the above tasks only. As soon as T_{start} expires, node ***a*** erases k_m but keeps y_1, y_2 and k_s. From **(1)** it is clear that k_m is used for authentication purpose because while receiving this broadcast, another node ***b*** is assured of the identity of node ***a***. Now following conditions can occur.

3.2.1 Node b is in T_{start} Phase

Use 15-point type for the title, aligned to the center, linespace exactly at 17-point with a bold font style and all letters capitalized.

If node ***b*** is in its T_{start} phase, it keeps both k_m and k_s. Besides its own ***HELLO*** broadcast, if it receives ***HELLO*** message from node ***a*** during T_{start}, it can decrypt **(1)** with k_m and gets k_a. After getting k_a, it can generate pairwise key as

$$k_{a,b} = f(k_a, b) \text{ if } a > b \text{ else}$$
$$k_{a,b} = f(k_b, a) \text{ if } a < b \text{ and sends a response to } a$$
$$b \longrightarrow a: REPLY|b|Ek_a(a|b|k_b|n_a+1) \quad (2)$$

f is a secure one-way function. Upon receiving this reply, ***a*** decrypts with its individual key k_a and validates the identity of ***b***. After that it generates pairwise key with ***b*** as ***b*** has generated above. It does not matter that whether node ***a*** is in its T_{start} phase or in T_{estab} phase, because it is not dependent on any master key to decrypt and process **(2)**. T_{estab} is the second time interval of T_{key} which lasts till the expiration of T_{key}. Node ***a*** only needs its individual key k_a to generate pairwise keys.

3.2.2 Node b is in T_{estb} Phase

If node ***b*** is not in T_{start} phase, it will be in T_{estab} phase where it has erased k_m. Node ***b*** will remain in T_{estab} phase until T_{key} expires. We know that most of the requests will be processed after T_{start}, because T_{start} is so small that a node hardly broadcasts its on

HELLO request. Now node ***b*** is not in position to decrypt **(1)**. Node ***b*** computes pair wise key as $k_{a,b}= f(k_b,a)$ only if $a<b$, and replies to node ***a*** as

$$b \longrightarrow a: REQUEST|b|a|n_a+1|y_1|y_2 \quad (3)$$

Upon receiving *REQUEST* message, if node ***u*** is in T_{start} phase, it can decrypt y_1 to verify the identity of ***b*** as well as to get k_b. Now node ***u*** computes the pairwise key as $k_{u,b}= f(k_b,u)$. If node ***u*** is in T_{estab} phase, it can do all the above tasks with the help of y_2. Because the ***HELLO*** message cannot be authenticated by node ***b***, resource consumption attacks can be launched. Solutions to these attacks have been suggested by (S. Zhu, 2003) which can successfully be adopted. Authentication process of newly coming nodes in ***SAKE*** is very simple and consumes less memory because a certain node has to store only y_1 and y_2 than the scheme proposed by Deng et *al*. (J. Deng), in which a number of verifiers are stored in memory for the same purpose.

Node ***b*** cannot compute $k_{a,b}= f(k_a,b)$ if $a>b$, because it cannot decrypt **(1)**, so in this case it simply sends **(3)** to ***a***. Upon receiving this message, if node ***a*** is still in T_{start} phase (although the chances are very rare), since it keeps k_m and k_s, it decrypts y_1 with k_m to verify the identity and to get individual key of node ***b*** and ignores y_2. But if node ***a*** is also in T_{estab} phase, y_1 is useless for it. It decrypts y_2 with the help of k_s to get ***b***'s identity and its individual key. So it has not only verified the identity of ***b*** but also generated key $k_{a,b}$. Node ***a*** verifies the identity of ***b*** and generates pairwise key $k_{a,b}= f(k_b,a)$ if $a<b$ and does not send any notification message to ***b***, otherwise generates $k_{a,b} = f(k_a,b)$ if $a>b$ and sends following notification message to ***b*** as

$$a \longrightarrow b: REPLY|a|b|Ek_b(a|b|k_{a,b}) \quad (4)$$

Node ***b*** decrypts this message and gets the pairwise key. An interesting situation arises if node ***a*** has received ***HELLO*** broadcast from ***b*** during its T_{start} phase and generated pairwise key $k_{a,b}$. But ***a***'s ***HELLO*** broadcast was received by ***b*** in ***b***'s T_{estab}. Now ***b*** generated $k_{a,b}$ and responded with **(3)**. When ***a*** receives this message, it checks the identity of ***b*** and discards immediately because it has already established the pairwise key.

When T_{key} expires, each node erases k_s and individual keys of its immediate neighbors with which it has established pairwise keys, but it retains y_1 and y_2 because they are used by a node to establish pairwise keys with newly coming nodes that come after the expiration of T_{key}. Depending upon the policy, if internal network processing (data aggregation and passive participation) is required, a node can be programmed to keep the individual keys of its immediate neighbors collected during key setup.

3.2.3 Adding New Nodes

If a new node ***u*** arrives, it will broadcast ***HELLO*** message as:

$$u \longrightarrow *: HELLO|u|n_u|\ Ek_m(u|n_u|k_u)$$

Suppose a node ***b***, which was deployed earlier and has completed its key setup process, receives this broadcast. Node ***b*** cannot decrypt the broadcast since it has erased k_m. Node ***b*** computes pair wise key as $k_{u,b}= f(k_b,u)$ and replies to node ***u*** as

$$b \longrightarrow u: REQUEST|b|u|n_u+1|y_1|y_2$$

3.3 Security Analysis of SAKE

The prime objective of ***SAKE*** is to protect a sensor network from master key compromise during key setup phase hence making the insertion of malicious node difficult. During the first time interval (T_{start}) of key setup phase, a typical node broadcasts the ***HELLO*** message encrypted with the master key k_m to identify its neighbors. T_{start} being a very small time interval of total key setup time ends after this broadcast and each node erases k_m from its memory though the key setup phase is not completed yet. According to our assumption, it is impossible for an adversary to compromise a node during T_{start}. Therefore an adversary has a very less time and chance to get k_m and it is clear that a sensor node must require k_m to be authenticated and to establish pairwise keys. In T_{estab}, which is the second and the larger time interval of key setup phase, if an adversary however compromises a node, she will get the subordinate key k_s only. Subordinate key k_s alone is useless for her, since the malicious nodes dropped by her need to have k_m for authentication as well as to establish pairwise keys with the legitimate nodes. Eventually the pairwise key is established when each node receiving the ***HELLO*** broadcast from each initiating node, sends ***REPLY*** message encrypted with the individual key of the initiating node rather than with the master key. This feature makes possible the establishment of pairwise key between the initiating and responding node even after the expiration of key setup phase of the initiating node when it has erased k_m. Initiating node will decrypt the ***REPLY*** message with its individual key which is

stored in it permanently. In other schemes ((J. Deng et al.), (S. Zhu, 2003), (S.Seys), (B. Lai, 2002), (A. Perrig et al., 2001), (B. Dutertre et al., 2004)), since master key is kept and required for entire key setup phase, a skilled adversary gets more time to compromise a node and to get the master key. Moreover the compromise of master key in LEAP (S. Zhu, 2003) allows adversary to determine all pairwise keys in the network. It is not possible in ***SAKE*** because master key k_m is only used for node authentication and cannot be used to compute all pairwise keys in the network.

4 CONCLUSIONS

We have proposed Secure Authenticated Key Establishment (***SAKE***) protocol that establishes pairwise keys shared between neighboring nodes of a wireless sensor network. ***SAKE*** is an efficient protocol in terms of memory, processing and communication. It makes a sensor network more resilient to master key compromise and makes the insertion of malicious nodes extremely difficult. We plan to implement and validate this security protocol in a simulator in near future.

REFERENCES

R. Anderson, H. Chan, and A. Perrig, "Key infection: Smart trust for smart dust," *In 12th IEEE International Conference on Network Protocols,* Berlin, Germany, 2004.

J. Deng, C Hartung, R. Han, and S. Mishra, "A Practical Study of Transitory Master Key Establishment for Wireless Sensor Networks," University of Colorado at Boulder, Boulder, CO, 80309-0430.

S. Zhu, S. Setia, and S. Jajodia, "Leap: Efficien security mechanisms for large-scale distributed sensor networks," *In 10th ACM Conference on Computer and Communication Security,* Washington D.C, USA, 2003

L. Eschenaur and V. Gligor, "A key-management scheme for distributed sensor networks," *In Proceedings of the 9th ACM Conference on Computer and Communication Security*, pp. 41-47, 2002.

S. Sarsan and V.G. Adadda, "Analysis and Evaluation of Key Management Schemes in Wireless Sensor Networks," 2004.

H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *In IEEE Symposium on Research in Security and Privacy,* 2003.

Du W, Deng J, Han Y S, Chen S, and Varshney P K, "A key management scheme for wireless sensor networks using deployment knowledge," *In Proceedings of IEEE INFOCOM'04*, IEEE Press, Hong Kong, 2004.

S.Seys, "Key Establishment and Authentication Suite to Counter DoS Attacks in Distributed Sensor Networks" *unpublished manuscript, COSIC.*

Lai, B., Kim, S., and Verbauwhede, I, "Scalable session key construction protocol for wireless sensor networks," *In IEEE Workshop on Large Scale RealTime and Embedded Systems (LARTES)*, 2002.

S. Basagni, K. Herrin, E. Rosti, D. Bruschi, "Secure Pebblenets," *In Proc. of MobiHoc* 2001.

D. Carman, P. Kruus and B. Matt, "Constraints and approaches for distributed sensor network security," *NAI Labs Technical Report No. 00010* , 2000.

A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "SPINS: Security Protocols for Sensor Networks," *In Proc. Of Seventh Annual ACM International Conference on Mobile Computing and Networks (Mobicom 2001),* Rome Italy, 2001.

Dutertre, B., Cheung, S., and Levy, J, "Lightweight key management in wireless sensor networks by leveraging initial trust," *Tech. Rep. SRI-SDL-04-02, System Design Laboratory,* 2004.

S. A. Camtepe and B. Yener, "Key Distribution Mechanisms for Wireless Sensor Networks: a Survey," *Rensselaer Polytechnic Institute*, 2005.