Secure and Energy Efficient Key Management Protocol for Ad hoc Sensor Network

Dr.Dhafer R. Zaghar*, Dr.Bassim Abdulbaki Juma & Yaarob M.Nafel***

Received on:6/8/2008
Accepted on:4/6/2009

Abstract

Recent advances in wireless communications and electronics have helped to develop sensor nodes which are low-cost, low-power, multifunctional, small in size and communicate in short distances. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks. Ad hoc sensor network is a multihop network made of hundreds of sensor nodes.

This Paper presents a proposed secure and energy efficient decentralized key management protocol. The proposed protocol combines three schemes; key establishment, key update and new node addition scheme. The energy consumption of the proposed key management is analyzed and compared with those of the formal protocols. The analysis shows an advantage in term of energy consumption over the previous work.

Keywords: Ad Hoc sensor network, key management, compromised node.

نظام آمن ذو استهلاك قليل للطاقة لادارة مفاتيح شبكات التحسس اللاسلكية

الخلاصة

التقدم الحاصل في مجال الالكترونيات و الاتصالات ساهم في تطوير اجهزة تحسس رخيصة الثمن و اقتصادية في استهلاك الطاقة و متعددة الوظائف و صغيرة الحجم و ذات مدى اتصال قصير.

هذه المتحسسات الصغيرة الحجم التي تتكون من وحدة معالجة البيانات و جهاز اتصال لاسلكي, ساعدت في بلورة فكرة شبكة المتحسسات اللاسلكية. تتكون هذه الشبكات من المئات من المتحسسات التي تعتمد في الاتصال فيما بينها على مبدا القفز المتعدد. ان هذا النوع من الشبكات يواجه العديد من التحديات واهمها هو محدودية مصدر الطاقة التي يعتمد عليها المتحسس. حيث انها لا تعتمد على مصدر طاقة مستمر في عملها بل على البطاريات. و لضمان اداء هذه الشبكات بشكل كفوء يجب ان يكون هناك انظمة امنية تقوم بحماية التطبيقات التي تعمل من اجلها هذه الشبكات. من اهم المتطلبات هو ضرورة تقييم احتياجات الطاقة.

هذا البحث يقترح نظام لامركزي آمن و كفوء لادارة المفاتيح و يتضمن هذا النظام ثلاث مهام فرعية وهي: Key establishment, Key update and New node addition. جرى دراسة هذا النظام مع اجراء مقارنة مع النظم المتعارف عليها و اظهرت النتائج كفاءة النظام المقترح في استهلاك الطاقة مقارنة بالنظم السابقة.

***Computer and Software Engineering Department, University of Al-Mustansiriya/ Baghdad**
**** Computer Engineering and Information Technology Department, University of Technology/Baghdad**

1. INTRODUCTION

Advances in hardware and wireless network technologies have placed us at the doorstep of a new era where small wireless devices will provide access to information anytime, anywhere as well as actively participate in creating smart environments. One of the applications of smart spaces is *sensor networks*, networks that are formed when a set of small sensor devices that are deployed in an ad hoc fashion cooperate in sensing a physical phenomenon. Sensor networks hold the promise of revolutionizing sensing in a wide range of application domains [1], because of their reliability, accuracy, flexibility, cost-effectiveness, and ease of deployment.

An ad hoc sensor network (ASNs) is composed of a large number of sensor nodes, which are densely deployed either inside the phenomenon or very close to it [2]. The limited energy and processing power of nodes make the use of public key cryptography nearly impossible. Instead, most security schemes make use of symmetric key cryptography. One thing required in either case is the use of keys for secure communication. Managing key distribution is not unique to ASNs, but again constraints such as small memory capacity makes centralized keying techniques impossible. Straight pair-wise key sharing between every two nodes in a network does not scale to large networks with tens of thousands of nodes, as the storage requirements are too high. A security scheme in ASNs must provide efficient key distribution while maintaining the ability for communication between all relevant nodes [3].

Above all these issues, to ensure the availability of the ASN, the sensor nodes should protect its resources from the unnecessary processing of key management messages in order to minimize energy consumption and extend the life of the network [4].

This paper presents a proposed key management protocol for an ad hoc sensor network. The proposed scheme involves three important schemes to improve the ASN functionality, which are:

- Secure key establishment.
- Secure key updates, (including compromised node isolation)
- Secure new node addition.

All three functions are analyzed in terms of energy consumption and security issues and compared with previous work. The analysis involves studying both computational and communicational energy consumption required for the proposed scheme based on the TinySec Security protocol [5] as a link layer security protocol. The analysis utilizes the AVR Studio® software tool kit to determine the energy consumption.

2. RELATED WORKS

Zhu, Setia, Jajodia (2004) [6], present a key management scheme (LEAP), where the pair-wise key is generated from a preloaded global shared master key and two neighbor's node identifiers, after a neighbor discovery phase. A cluster key is generated by a node itself and sent to each one of its neighbors, encrypted by their respective pair-wise keys, so far established.

Where *pair-wise key*, is shared between a node and each one of its neighbors, used in pair-wise communication among them; *cluster key*, is shared between a node and all its neighbors, used in local broadcast communication.

Prajapati (2005) [7], proposed key update scheme, the key is rotated circularly. Depending on the initial value of the preloaded key, the updated key will have various combinations of 1's and 0's. The operations required for the whole process are; setting the bit in the length field and performing circular rotation. Setting the length field bit is to inform the node that received the packet about the update process, and so to perform the rotation on the old key. The scheme allows for 64 times update process until the key repeats itself. Prajapati does not address the security requirements for this scheme.

Oliveira, Wong and Nogueira (2007) [8], present the (NEKAP) scheme, which is unlike the formal work; where each node broadcasts its master key, then each node generates the cluster key for each one of its neighbors based on the master key of its neighbor.

The pair-wise key for two nodes A and B is a function of all master keys of their neighbor nodes they have in common.

To generate the pair-wise key, A and B must know the common neighbors. For that, each node broadcasts its list of neighbors. This scheme overcomes the security defects in LEAP.

Landstra, Zawodniok, Jagganathan (2007) [9], proposed a subnetwork key management strategy, it allows the dynamic creation of high security subnetworks within the wireless sensor network and provides subnetworks with a mechanism for dynamically creating a secure key. Dynamic keys techniques minimize the number of keys that must be stored by nodes. This protocol depends on forming (subnetwork head node), that is responsible for creating the group key and broadcasting it to the entire nodes group. But forming the ASN into subnetwork (cluster) is inefficient, since cluster based approaches usually create single points of failure.

3. DEFINITIONS AND ASSUMPTION

A. *Definitions*

$E_K(M)$	Message M Encrypted Using key K
K_A	Node A Link key
K_{AB}	Shared Pair-Wise key of nodes A and B
K_m	Master Key
K_{mA}	Master Node key of node A
MAC (E_k (M, *K*)	Message M Authenticated Using Key *K*
Hash(M)	Message M hashed using SHA-1 hash algorithm

The proposed key management allows for establishing link and pair-wise keys with each direct, one-hop neighbor. The link key is used for local broadcast and shared with each one-hop neighbor node, while the pair-wise key is used to protect peer to peer communication and each node has a unique key with every adjacent one-hop neighbor node.

B. *Assumption*

A master key scheme depends on an assumption, by which the node can

discover its immediate neighbors. The required keys are derived from the master key which is preloaded in each node before deployment. The master key is erased in a time t_{est}, expecting t_{est} to be in the order of several seconds. So t_{est} must be shorter than time t_{min} required by the attacker to compromise a sensor node and extract the key materials. This will prevent the attacker from computing the keys that are used in other regions of the network [6].

However, during the nodes deployment process some nodes cannot initiate due to hardware problems. So keeping the master key imposes risks to the network.

If any of these nodes is tampered with and its keys are captured, the aggressor will have access to the master key and thus will be able to get all the link and pair-wise keys of the network [8].

4. PROPOSED KEY MANAGEMENT PROTOCOL

A distributed key management protocol is proposed in this work; it has minimal energy consumption and overcomes the security and technical defects in the previous protocols. It does not require tamper resistant sensor node hardware and suites the TinySec link layer security platform.

A. *Key establishment step*

Each node is preloaded with master key K_m and K_s, where K_m is used for generating the link and the pair-wise keys, while K_s is used to authenticate a broadcasted list by the task manager (section 4.C).

The first action is to compute the link-wise node key. For node X, it first generates a random number R_X and stores it. There are several methods for generating random numbers; one of the old methods, the random number can be directly generated from existing physiological signals collected by the sensors, with proper sampling rate and quantization resolution.

A simple method of random binary sequences generation is to seriate the least significant bits of quantifying values [10].

Another efficient and simple method dedicated for the sensor nodes which is based on a free running timer, a random number is generated by XORing the current value of the timer with a key.

After generating a random value, one's complement of the timer value becomes the new key and the one's complement of the new random number becomes the next timer value [11].

After generating the random number, the sensor node applies the hash function (SHA-1 hash algorithm) to the K_m and R_X and its identification X (The node ID according to the TinySec Protocol is 16-bit):

$$K_{mX} = \text{Hash}\ (K_m)$$

Node X link-wise key K_x = Hash (K_m || X || R_X)

K_{mX} will be stored and used to authenticate the new key released by the task manager in the key update step (section 4.B), and also used to communicate with new nodes inserted to the network (section 4.C).

Then node X announces itself to its neighbor by sending the announcement message, which contains its link-wise key K_x, encrypted and authenticated using K_m:

Node X announce: MAC (E_{Km} (K_x), K_m)

Each node in the network performs the same steps which is a neighbor

discovery process, after a duration each node will receive a number of the announcements messages from its one-hop neighbor.

Now node X starts to compute its neighbor's pair-wise keys depending on the received message. For each received announcement from node Y, node X will first verify its authenticity using the master key K_m. Then it stores node Y link-wise key K_y (which is also generated in the same process in node X) and saves node Y ID.

After that, it calculates their pair-wise key:

Node X,Y pair-wise key:

$$K_{xy} = \text{hash} (K_m \| K_x \| K_y)$$

Since node Y receives node X announcement, then it's capable of computing their pair-wise key in the same way.

The order of ($K_m \| K_x \| K_y$) depends on node's ID, if X > Y then this order is true, but if X < Y then the order will be ($K_m \| K_y \| K_x$).

Finally each node deletes the master key K_m. At that time each node successfully establishes a secure link with each neighbor node.

Figure (1) represents the establishment process flowchart.

B. *Key update step*

The proposed scheme depends on generating different master keys in each hop, which will limit the effect of master key exposure.

As in the previous methods, the task manager already generates a key hash chain. The key chain is a sequence of numbers, $K_0, K_1, \ldots, K_{n-1}, K_n$ such that for all L, $0 < L \leq n$, $K_{L-1} = \text{Hash}(K_L)$. Figure (2) demonstrates the process.

where K_0 is the first master key that is preloaded into the sensor nodes. The next key that is updated is K_1.

When each node receives K_1 it will compute the hash of K_1. If Hash (K_1) equals K_0, then this key is authenticated and its source is the base station [6][12].

The task manager initiates the key update process either to replace the existing keys to achieve the semantic security [5], or because of detection of a compromised node. In turn the controller has to replace the existing key to invalidate of the exposed key.

Once the task manager sends the new key, each node will receive the new key and computes the new link and pair-wise keys based on the new master key and forward the key to its neighbor.

Each node produces the next hash image of the key and forwards this image to its neighbor. Besides the new key, the task manager sends a list containing the IDs of the compromised nodes such that each node which received the list will check its neighbor IDs list. If it does exist, it will delete the related keys to the compromised node and inform the other nodes using pair-wise keys.

When the node receives the new key, it will authenticate this key. If K_{old} equal Hash_L (K_{new}), then the key is authenticated. Hash_L means performing the hash function L times on the key.

As an example, node B receives the new key, the level L and the list of the compromised node IDs (S) from its neighbor node A, the message is $\{K_{new}, L, S\}$. Level L identifies the number of hashes made to the received new key K_{new} since for each passed hop a new hash will be performed at the received key and the level is incremented by one.

First task carried by node B is authenticating the key and then computes the link key of node A:

$$K_A = \text{hash}\ (K_{new} \parallel A \parallel K_{A\text{-old}})$$

The next operation is incrementing the received level by 1, and computing the link key:

$$L = L + 1$$

$$K_{mB} = \text{Hash}\ (K_{new})$$

$$K_B = \text{hash}\ (K_{mB} \parallel B \parallel K_{B\text{-old}})$$

K_{mB} represents the master level key. This key will be used in the node addition process and for authenticating the new key updated received from the task manager.

The pair-wise key with node A will be computed:

$$K_{AB} = \text{hash}\ (K_{new} \parallel K_{AB\ old} \parallel K_A \parallel K_B)$$

When the computations are done, the node checks its stored list of neighbor nodes and compares it with the received list of compromised nodes. If a difference exists, then the node has to distribute the new key by authenticating and encrypting it with the pair-wise keys of the nodes that are not compromised.

The compromised node will be out of operation, since it does not have the pair-wise keys.

Now node B master key K_{mB} is the new key K_{new} which will be forwarded to the next hop neighbor nodes ($K_{new} = K_{mB}$) .

If no compromise node exists:

Node B announce: MAC (E_{KB} (K_{new}, L, S), K_B)

And if compromised node exist

Node B sends: MAC (E_{KBX} (K_{new}, L, S), K_{BX})

Where K_{BX} , is the pair-wise key between node B and nodes X.

After duration of time t, node B receives the announcements of all neighbors' nodes (since node B can not directly compute the link and the pair-wise keys for its neighbors as in the establishment step). In order to overcome the problem of mismatch keys, the exact level of each node will be maintained.

For a message received from node X {K_{new}, L, S}, node B will compute the link and pair-wise keys:

$$K_X = \text{hash}\ (K_{new} \parallel X \parallel K_{X\text{-old}})$$

$$K_{BX} = \text{hash}\ (K_{new} \parallel K_{BX\ old} \parallel K_B \parallel K_X)$$

Since two pairs of nodes may have different levels, a conflict in the level that K_{new} is based on may occur. Then choosing the level that K_{new} is based on follows the higher level between the two nodes as in computing the pair-wise keys. The last action taken by node B is deleting the received new master key K_{new} and all its old neighbor links and pair-wise keys.

Figure (3) simplifies the process in node B after receiving the new key K_{new} from node A, and then node B forwards the key to its neighbor's nodes X.

Packet loss problem could be easily solved in this scheme. Since each node waits for time t to compute the link and pair-wise key with each neighbor node, then it could easily compare the received message sender ID with its stored list that defines the neighbor nodes. Once a node finds that one of its neighbor nodes did not respond, then it has to resend the new key using the old shared pair-wise key.

C. *New node insertion*

The proposed key management simply allows adding new nodes to the network which is simply done through a process similar to the establishment step. Since the task manager is involved in this process (it takes the decision for adding new sensor nodes to the networks, either to compensate for the failing nodes

or to increase the network size). Then the base station (Task Manager) has first to broadcast a message that announces the list of IDs of the new sensor nodes. Then each sensor node stores the received list for a limited time *t*. One reason for broadcasting IDs list is to make this step of ASN key management immune to HELLO attack. The task manager broadcasts the ID list of the new nodes after deployment.

An adversary that captures a node will initiate this attack as a new node wants to join the network [13]. But when the base station informs the ASN nodes about the new nodes IDs, only the specified nodes will be allowed to join the network.

Also the broadcasted list contains K_{s+1} the next key of the hash chain (as in Figure 2). The old node verifies the authenticity of the received list in the same procedure with the key update step by checking if K_s equals Hash (K_{s+1}), where K_s is already preloaded as K_m but not deleted since its exposure makes no threat on the network operation. This verification will prevent an adversary from sending a decoy message that contains a list of malicious nodes ID's to add them to the network as new nodes.

Since the base station broadcasts can reach the entire network [4], and then the task manager can utilize this property to broadcast this list. Even if captured by an adversary, it will have no threat.

Each new node added to the network must be preloaded with the master key. When the node joins the network it first announces JOIN message which contains its ID. The link-key encrypted with its master level key, and only level L is authenticated in order to allow the receiving nodes to compute the master level key of the new node according to its announced level.

Each node receives the JOIN message will check the IDs list received from the task manager. If the ID node exists then it will continue with the process of adding new nodes. Else, a replication node attack will be detected and the base station is informed about the attack by sending the sensor node ID of the node that initiate the JOIN message.

After ensuring the existence of the joined node ID, the nodes will compute the pair-wise keys with new node according to its link key. Then they send the messages that contain their levels (not encrypted) and their links keys (encrypted with their master level key) to permit the joined node to compute its neighbor master level keys for each one and authenticate and decrypt the message.

For example, the new node A wants to join the network. It first broadcasts ID A, K_A and level L. As mentioned earlier, the ID included in the packet is represented by the source address:

$A \rightarrow *$: JOIN: MAC ((A, L_A, $E_{KmA}(K_A)$), K_{mA})

Where * denotes the broadcast process. The K_{mA} is generated by:

$$K_{mA} = \text{Hash}_L \text{ (preloaded } K_m)$$

Where the level L takes the highest value that could occur in the network.

K_A will be generated by computing the hash function to the master level key, the random number and the ID of node A:

$$K_A = \text{Hash } (K_{mA} \| A \| R_A)$$

Then each node X received this announcement checks the ID list of the new joined nodes received from the base station. When it finds the ID,

node X computes the pair-wise keys with the new node:

K_{AX} = Hash (K_{mA} || K_A || K_X)

After that it responds to the JOIN message that is initiated by new node. The JOIN message contains the link key (encrypted with master level key of node X) and the level (not encrypted):

X: MAC ((L_X, E_{kmX} (K_X,)), k_{mX})

Once node A receives this message, according to L_X, it will compute node X master level key and authenticate and decipher the received message. Finally it will compute the related pair-wise key with node X delete the master key.

K_{AX} = Hash (K_{mA} || K_A || K_X)

Figure (4) depicts this process.

Time *t* is the time that the nodes wait in order to get the addition of new nodes. It varies from network to network depending on the deployment strategy.

5. PERFORMANCE ANALYSIS

AVR Studio® [14] is used to evaluate the execution time and number of cycles of the impact of the cipher functions (encryption and authentication process using the RC5 cipher algorithm) taken by the microcontroller to carry out the implementations of TinySec protocol and SHA-1 hash algorithm. The implementations were tested over AVR microcontroller Atmega128 4MHz with 2.1mA and 3.0V [15] utilizing AVR Studio® simulator. The estimation of the energy consumption for the microcontroller depends on the average power dissipated while running a program [16] [17]:

$$P_{SW} = I_{AVE(m)} * V_{DD} \quad \text{..... (1)}$$

where $I_{AVE(m)}$ is the average current drawn by the microcontroller in amperes and V_{DD} is the supply voltage in volts. The associated energy in joules is given by

$$E_{SW} = P_{SW} * t_{SW} \quad (2)$$

Where t_{SW} is the execution time of the software program in seconds

To maintain good results about the approximate energy value that is consumed during each step of the key management scheme, two factors must be considered, the communication and computation part. In the communication part, the energy consumption for sending messages depending on the scheme must be calculated. The energy consumed by computing the keys and ciphering messages must also be evaluated.

The next sections will compare the proposed key management with the previous works on this field. Two key management schemes are analyzed and involved in the comparison, LEAP [6] and NEKAP [8].

A. *Key management analysis*

Table (1) shows the comparison of the cost required by each node with 20 neighbor nodes for the proposed key management with LEAP and NEKAP.

B. *New node insertion cost*

In this section a comparison between the proposed key management and LEAP and NEKAP is made. Although the defect is encountered in LEAP for computing pair-wise keys, because the computation depends on the master key but the old nodes have already erased this key and will not be able to compute this key [7]. If the node has the master keys, the total cost for the new node addition step will be the same for the cost of

establishing step as mentioned in Table (1). Also the same problem applies to NEKAP, the old nodes have already deleted the master keys used in the link and pair-wise keys establishment so that they will not be able to compute the required keys.

The comparison between the proposed key management with the LEAP and the NEKAP cost for the new node is shown in Table (2).

Table (3) shows the cost required for the old nodes that respond to the new deployed node.

6. Discussion

The proposed key management scheme reduces the (neighbor discovery) phase time t_{est}. This will block the chance against the attacker to compromise a node and revealing its master key.

Tampering during the initialization, the attacker can reveal the global key and all exchanged keys at the eavesdropping zone. These attacks can reveal pair-wise keys and cluster keys exchanged in that point of network. This is the case with all the previous and proposed key management.

In all communications mode, first thing is to check the source address of the received packet (meaning the ID) and according to this ID, the related cluster or pair-wise keys is fetched, then the received message will be authenticated.

The generation of the random number by each node in the key establishment step makes the entire sensor network cluster and pair-wise keys independent of each other. Thus the revealed keys from a compromised node affect only the neighbor nodes. Also clone node attacks are prevented since the other parts of the network will have a different set of keys. Once initiated the attack will be detected because of the original nodes depend on their stored list of the neighbor node IDs and keys.

With the LEAP scheme, if one node is tampered with and its keys are captured, the aggressor will be able to get all the pair-wise keys of the network.

For the new node insertion scenario of NEKAP scheme, each node has to send its neighbor node ID list. If one tampered node exists, then the attacker can easily compute the old pair-wise keys shared between the old nodes since the derivation of these keys is based on the ID list that is sent by each old node to the new node for establishing pair-wise keys.

7. Conclusions

The proposed key management reduces the dependency on the base station in order to minimize the communication overhead. The analysis in this work shows that the communication has the highest impact on the sensor node energy consumption, such that the cost for sending one packets is 66 times higher than the cost of the cipher computation performed on the packet. Then each step of the proposed key management (key establishment, key update, new node addition) relies on one message sent by each node to complete each of the required step which results in a light key management.

The proposed key scheme reduce t_{est} which limits the attacker from gaining session keys, making this scheme more immune than the previous ones.

References

[1] Sameer Tilak, Nael B. Abu-Ghazaleh and Wendi Heinzelman, "A Taxonomy of Wireless Micro-Sensor Network Models" ACM SIGMOBILE Mobile Computing and Communications Review, 2002.

[2] Ian f. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam and Erdal Cayirci, "A Survey on Sensor Networks", IEEE Communications Magazine, Aug. 2002.

[3] Yong Wang, Garhan Attubery, AND Byrav Ramamurthy, " A Survey of Security Issues in Wireless Sensor Networks", IEEE Communications Surveys & Tutorials, 2nd Quarter 2006.

[4] Fei Hu, Neeraj K. Sharma, "Security considerations in ad hoc sensor networks", Ad Hoc Network, Elsevier, 2003.

[5] Chris Karlof, Naveen Sastry and David Wagner, "TinySec: A Link-Layer Security Architecture for Wireless Sensor Networks," SenSys '04: Proc. 2nd Int'l. Conf. Embedded Networked Sensor Systems, New York: ACM Press, 2004.

[6] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, "LEAP: Efficient Security Mechanisms for LargeScale Distributed Sensor Networks", Tech. report, ACM, Aug. 2004.

[7] Kamini B. Prajapati, "An Efficient Key Update Scheme for Wireless Sensor Network", Master Thesis, Washington state University, 2005.

[8] Sérgio de Oliveira, Hao Chi Wong, José Marcos Nogueira, "NEKAP: Intruder Resilient and Energy Efficient Key Establishment in Sensor Networks", IEEE, 2007.

[9] Tim Landstra, Maciej Zawodniok, S. Jagannathan, "Energy-Efficient Hybrid Key Management Protocol for Wireless Sensor Networks", 32nd IEEE Conference on Local Computer Networks, IEEE Press, 2007.

[10] Shu-Di Bao, Yuan-Ting Zhang, Lian-Feng Shen, "A Design Proposal of Security Architecture for Medical Body Sensor Networks", Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks, IEEE 2006.

[11] Deva Seetharam and Sokwoo Rhee, "An Efficient Random Number Generator for Low-Power Sensor Networks". http://alumni.media.mit.edu/~deva/papers/emnets.pdf

[12] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar, "SPINS: Security Protocols for Sensor Networks", ACM Press, 2001.

[13] Chris Karlof, David Wagner, " Secure routing in wireless sensor networks: attacks and countermeasures", Elsevier Press, 2003.

[14] http://www.atmel.com/

[15] Chulsung Park, Kanishka Lahiri, Anand Raghunathan, "Battery Discharge Characteristics of Wireless Sensor Nodes: An Experimental Analysis", IEEE, 2005.

[16] William Fornaciari, Paolo Gubian, Donatella Sciuto, Cristina Silvano, "Power Estimation of Embedded Systems: A Hardware/Software Codesign Approach", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 6, NO. 2, JUNE 1998.

[17] Adam Dunkels, Fredrik - Osterlind, Nicolas Tsiftes, Zhitao He, "Software-based On-line Energy Estimation for Sensor Nodes", ACM Press, 2007.

Table (1) Comparison of Key establishment cost schemes.

Scheme	No. of Required Preloaded Keys	Computation Cost(µJ)	Communication Cost(mJ)	Total Cost(mJ)	No. of Required message
LEAP	1	98.271	9.3215	9.4197	1+D
NEKAP	2	46.179	1.4992	1.54537	3
Proposed	1	150.169	0.4563	0.6064	1

Table (2) Comparison of new node insertion cost schemes (new node side).

Scheme	Computation Cost (µJ)	Communication Cost (mJ)	Total Cost(mJ)	No. of Required message
LEAP	98.271	9.3215	9.4197	1+D
NEKAP	46.179	1.4992	1.54537	3
Proposed	147.901	0.4725	0.6204	1

Table (3) Comparison of new node insertion cost schemes (old node side).

Scheme	Computation Cost	Communication Cost	Total Cost	No. of Required message
LEAP	7.749 µJ	0.4563mJ	0.4637mJ	1
NEKAP	32.508 µJ	1.9555mJ	1.988mJ	4
Proposed	9.765 µJ	0.4725mJ	0.4822mJ	1

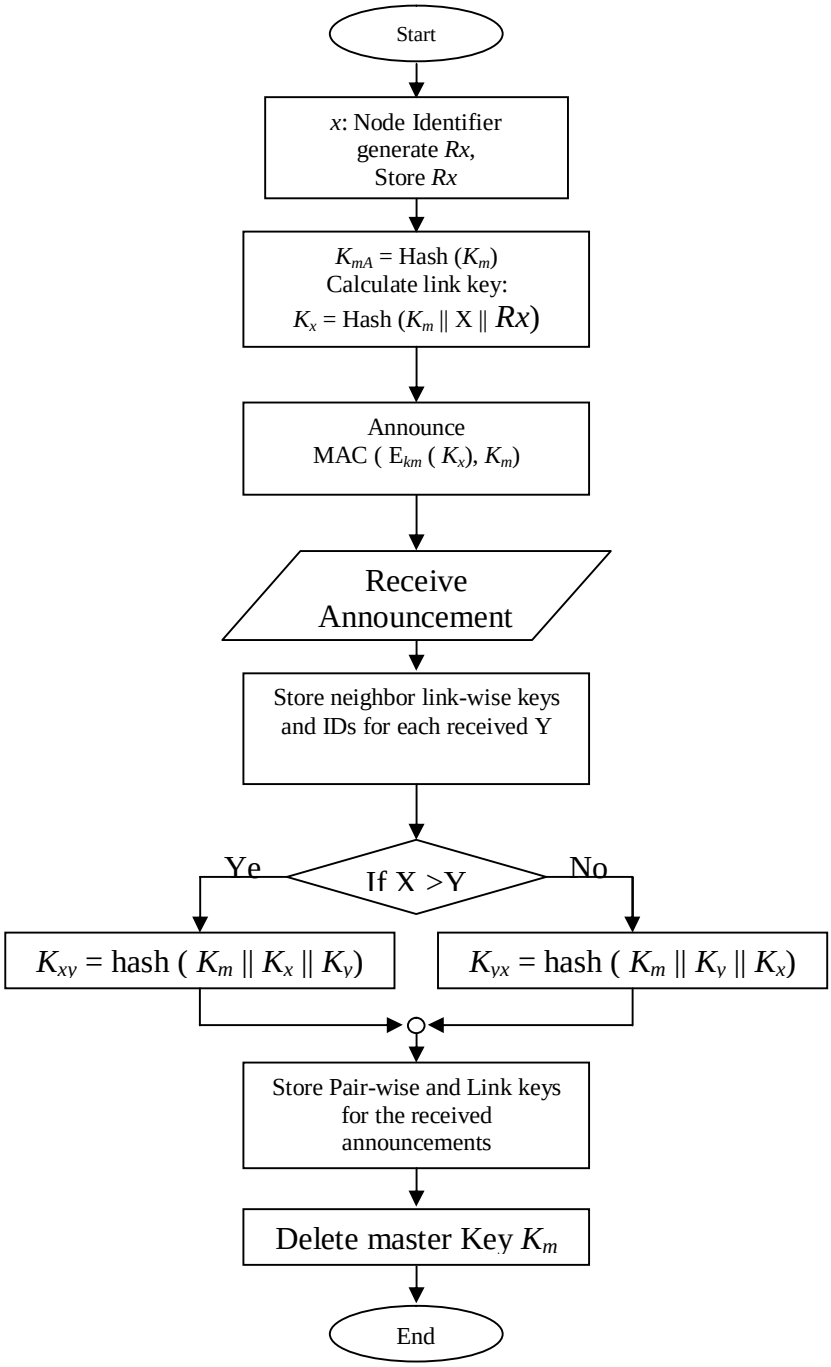

Figure (1) Key establishing flowchart.

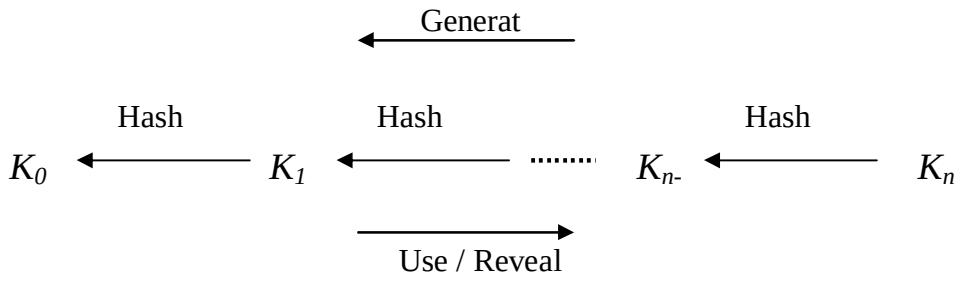

Figure (2) Key hash chain.

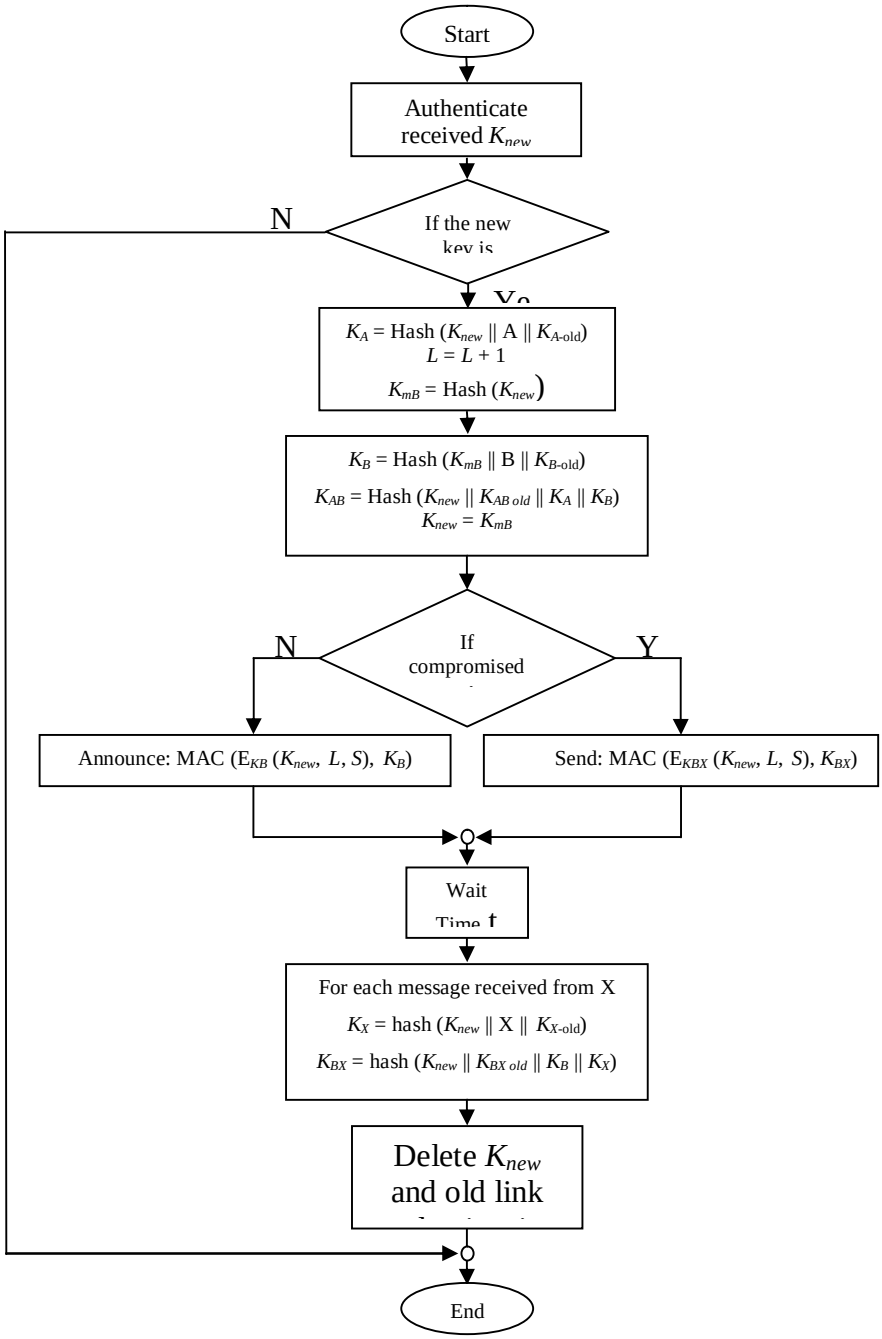

Figure (3): Flow Chart of key update process.

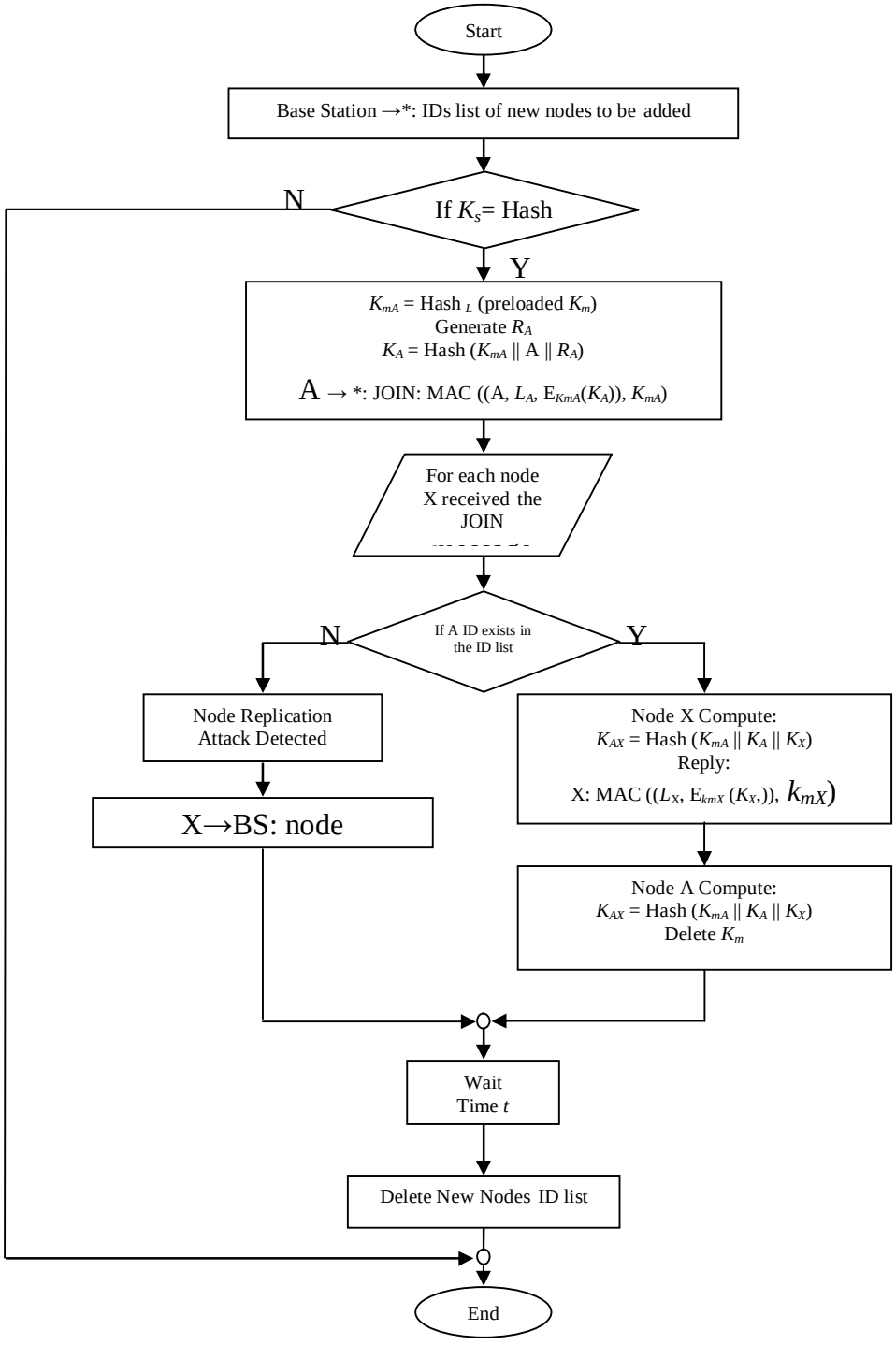

Figure (4): Flow Chart of new node insertion process.