A Key Renewal Scheme under the Power Consumption for Wireless Sensor Networks

Chien-Lung Wang[1], Tzung-Pei Hong[2], Gwoboa Horng[1] and Wen-Hung Wang[1]

[1]Department of Computer Science, National Chung-Hsing University
[2]Department of Electrical Engineering, National University of Kaohsiung

Abstract

In this paper, we propose a distributed, reliable and low-power-consumed key-renewal scheme for wireless sensor networks. The proposed scheme is divided into parts: the server part and the sensor-node part. The server will produce an appropriate key-generation function (KGF) for key renewal on sensor nodes under a power consumption constraint. It divides the function into slices and sends the slices to sensor nodes. The sensor nodes will assemble the slices to rebuild the key-generation function for key renewal as they receive the slices. The experimental results also show the effectiveness of the proposed key renewal scheme.

Keywords: key renewal, security, power consumption, permutation, wireless sensor networks.

1. Introduction

Sensor networks are a kind of ad-hoc networks [3] and are widely used in a variety of applications. In a sensor network, sensor nodes are deployed in different locations, responsible for perceiving local information and reporting to the server. In real applications, sensor nodes are usually deployed in a large number in order to cover a sufficiently large area. For instance, a military aircraft may scatter a lot of tiny sensor nodes over a certain terrain to gather information. When the amount of sensor nodes used in an application is large, it is better for these nodes to be provided as cheaply as possible. If all sensor nodes used in an application are the same, they can be manufactured uniformly and the production costs can be reduced. Specifically, "uniform" means that each sensor node is equipped with the same hardware, software, and initial settings.

When sensor networks are deployed in a malicious and hostile environment, security becomes extremely important since these networks may be maliciously attacked. For example, a malicious adversary can easily eavesdrop or interpolate the traffic communication information, and may intentionally provide misleading information to other nodes or impersonate one of the network nodes [3]. Developing appropriate security policy for sensor networks is thus desired.

A sensor node is usually limited by its computing ability, memory size, communication protocols and battery power. These constraints make public-key algorithms infeasible for sensor nodes. Over the past two decades, several security protocols [1], including key pre-distribution [2][4][5] and key renewal [7] schemes, were thus proposed for sensor networks.

It has been criticized that each sensor node with the same key is dangerous. An efficient key renewal scheme with efficient power control is thus necessary if all sensor nodes are initially equipped with the same key. The initially key should be updated immediately after the deployment of sensor nodes since a compromise of the initial key of a sensor node may collapse the entire sensor networks.

In this paper, we thus propose a distributed, reliable and low-power-consumed key-renewal scheme for wireless sensor networks. It is reliable in that the key-renewal function will be re-generated in each predefined interval, such that it is hard for an attacker to crack the function in time. It is power-efficient in that it only needs to consume the power under the power consumption constraint. The proposed key renewal scheme can thus not only meet the low-power constraint but also consider the security requirement.

The server will produce an appropriate key-generation function (KGF) for key renewal on sensor nodes under a power consumption constraint. It divides the function into slices and sends these slices to sensor nodes. As sensor nodes receive the slices, they will assemble the slices to rebuild the key-generation function for key renewal. Experimental results also show the effectiveness of the proposed scheme.

2. The Proposed Key Renewal Scheme

The proposed key renewal scheme is divided into two parts: server part and sensor part. The server part plays an important role in the process since it has to generate a low-powered function for key renewal. The process of the server is depicted in Figure 1. Each step will be described in the next section.

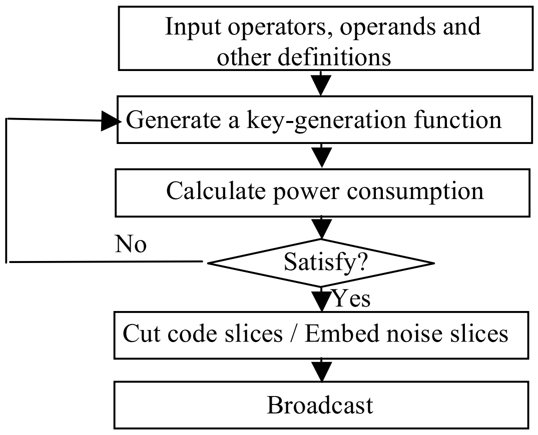
Input operators, operands and other definitions
Generate a key-generation function
Calculate power consumption
No
Satisfy?
Yes
Cut code slices / Embed noise slices
Broadcast

Figure 1: The server process

We assume here all sensor nodes are identical, with the same initial energy. The server and all sensor nodes initially share a common key k_c. Note that this key is only used to initialize encryption keys and should be annihilated as soon as the deployment is done. A time bound t is set to prevent the catastrophic consequence of compromising the sensor networks. Each sensor node will count down a timer immediately after its physical deployment. It will assemble the slices to rebuild the key-generation function for key renewal as it receives the slices.

3. The Sever Part

The details in the server part are described in the section.

3.1. Code slices and key-generation functions

The server must first prepare some predefined operators and operands for generating code slices, which will then be used to form key-generation functions. An operand can be any positive integer and an operator can be one of any existing or user-specified binary operators. Some examples for operators are addition, subtraction, multiplication, division, exponentiation, or, xor, shift and modulus, among others. A code slice is composed of an operand and an operator. For example, +2, xor9, *4, or6 and ^3 are possible code slices. With the five code slices in the example, 120 possible combinations (without repetition) may be generated. That is, there may be 120 possible functions. If there are *m* code slices, then there are *m!* possible functions. The package format of a code slice delivered from a server to a sensor node is designed as shown in Figure 2.

sn		*opr*	*opd*
rgn	*pb*		

Figure 2: The package format of a code slice

In Figure 2, three fields are included in the package. The first field, *sn,* stores the slice number, which is used to identify the code slice. It is unique in each run for forming a key-generation function. The *sn* field consists of g bits, which can be divided into two parts: a randomly generated number, *rgn,* and a poison bit, *pb*. The subfield *pb* occupies only one bit and may be located at any bit of the *sn* field. The actual location for *pb* is determined by the following function:

$$\text{location}(pb) = k_c \bmod g,$$

where k_c is the current key in the server. If *pb* is 1, meaning the code slice is a noise, the entire code slice becomes invalid and is not used to form a key-generation function; otherwise, the code slice is used as a part to form a function. The subfield *rgn* is composed of the remaining g-1 bits. It is a unique random number generated by the server. If a random number generated is the same as the previous one, it will be re-generated again. The other two fields, *opr* and *opd*, represent the operator and the operand, respectively. The code slice pool is formed from all combinations of *opr* and *opd*. A key-generation function (KGF) is composed of several code slices and a constant c, with c put at the front. It can be represented as:

$$KGF = c \text{ concatenate } code\_slice_1 \text{ concatenate } code\_slice_2 \text{ concatenate} \ldots \text{ concatenate } code\_slice_k,$$

where k is the number of valid code slices used to form the function and *concatenate* is a string operator to concatenate two strings together. In general, if there are m code slices, then there may be *m!* possible key-generation functions.

3.2. Power consumption for key generation

When a server generates a set of code slices, it distributes them to sensor nodes. Each sensor node then rebuilds the key-generation function from the valid code slices received and generates the new key from the function. This paper assumes each operator has its own power consumption. For example, let the operator "+" need one unit time. The operator "-" can be defined to need the same time as the operator "+" since they have the same logic operation. The operator "*" may be performed by several addition operators. The cost of "*" thus depends on the amount of the multiplication. Similarly, the operator "/" may be performed by several subtraction operators. The cost of "/" thus depends on the amount of the quotient. The calculation for "*opd1/opd2*" needs at last one time unit even when *opd1* < *opd2*. The priority of the operators is defined as follows. The "*" and the "/" operators have the same priority and have higher priority than the "+" and the "-" operators, which are higher than the logic operators.

As mentioned above, each sensor node is equipped with an initial amount of energy. The key renewal operation should not spend much power such that the sensor nodes can have enough energy to do their jobs. But if the key-generation function is too simple, the function will easily be cracked. Thus, an appropriate key-generation function is requested to satisfy the following power-consumption constraint:

$$E - t \leq PC(f) \leq E + t,$$

where $PC(f)$ is the power consumption for evaluating the function f, E is the desired energy consumption for a key renewal operation, and t is the energy tolerance. Thus, the energy of all the possible key-generation functions is bounded within $E \pm t$. When an operation takes more time units, it also needs more energy. The power consumption by each possible key-generation function in sensor nodes can thus be measured by the number of time units needed.

3.3. Sending functions to sensors

After the server finds out a key-generation function that satisfies the power consumption constraint, it decomposes the function into code slices and add some noise code slices selected from the code slice pool. The server then broadcasts these code slices to the wireless sensor nodes. For a predefined time interval, the server will re-generate a key-generation function for avoiding the previous function to be cracked.

4. The Sensor Part

In a sensor network, a sensor node can only communicate directly with its neighbors within a short range. A header is usually chosen from a subnet of sensor nodes as a relay. In the proposed scheme, the one with the most neighbors is elected as the header. At the first stage, each sensor node sends out its ID to notify its neighbors and counts the number of its neighbors. At the second stage, each sensor node announces this number. Based on this information, headers can be elected and located by their neighbors. This method is called an echo algorithm.

After headers are elected from sensor nodes, the next step is to form a secret program. Let the subnet of a header be defined as the network formed from the header itself and all its neighbors. Thus, there are many subnets in a wireless sensor network. Some subnets may overlap.

Each header then generates a random permutation of m objects, denoted by a string r_m, upon receiving code slices sn_1 *to* sn_m from the server. It then deliveries r_m to its subnet and routes a path to the server. With the permutation r_m, the code slices can then be assembled together to get a common program (function) P. The r_m should also be encrypted with the key K_c before it is sent out in order to prevent eavesdropping.

After an agreement of the program P has been made between the server and the subnet of a certain header, subsequent re-keying can be done by executing the program P on the previous key of the subnet, i.e. $k_j = P(k_{j-1})$. Initially, the key of all subnets is $k_c = k_0$. But after the rekeying procedure, each subnet should have its own key agreed with the server.

5. An Example

An example is given here to illustrate the above key renewal scheme. Assume the set of operators is {+, -, *, /} and the set of operands is {3, 4, 5, 6}. The pool of code slices is thus {+3, +4, +5, +6, -3, -4, -5, -6, *3, *4, *5, *6, /3, /4, /5, /6}. If the parameter (E) of the desired energy consumption for key-generation function is 10 time units and the parameter (t) for tolerance is 5. The energy of each desired key-generation function thus lies between 5 and 15. The energy of the function, f=+4*6-5, is 6, satisfying the constraints. The function can thus be used as the key-generation function. In additions, there are three code slices "+4", "*6" and "-5" in the function. Let the initial common key k_c be set as 20 and the length (g) for slice number (sn) be set at 8. In the code slice package, the poison bit is located at k_c *mod* g, which is 4. The four bit is thus a poison bit.

6. Experiments

Experiments were made to show the effectiveness of the proposed scheme. Assume one time unit of calculation needs a power unit. In the experiments, time units are thus used, instead of the power units. The proposed approach was compared with the one without considering the power-consumption constraint. The key-generation functions in the latter are randomly generated with a specific parameter setting. The parameters used in the experiments were shown in Table 1.

The two parameters E and t in the power consumption constraint are set at 100 and 10, respectively. The total power of each sensor node for key renewal is set at 10000, such that keys can be renewed about 100 times. The remaining power at each time of key renewal for the three settings in Table 1 is shown in Figure 3. It could be observed that the three settings caused obvious difference in power consumption. The proposed scheme is better than the others in power control. The former decreased smoothly, but the latter might consume all energy and remain no energy for next re-keying. In figure 4, It is shown that the consumption power of each rekeying. The proposed scheme is horizontal and the others are unstable curves. From the experiments, our scheme is a reliable and low-power-consumed key-renewal scheme for wireless sensor networks.

Table 1: The parameter setting in the experiments

	Methods		
	Proposed Approach	Random-1	Random-2
Integer num	1~100		
Num of *opr*	6	4	6
Num of *opd*	7	5	7

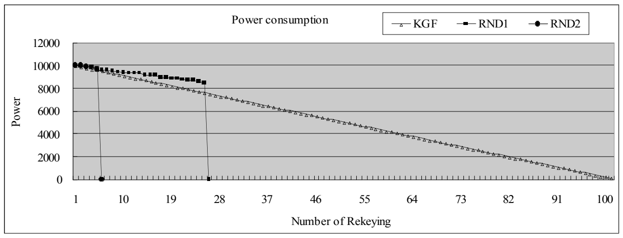

Figure 3: The remaining power at each time of key renewal

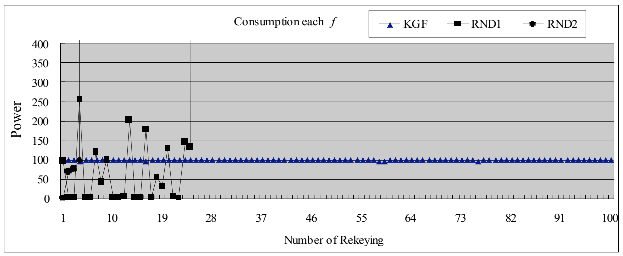

Figure 4: The consumption power at each time of key renewal

7. Conclusion

We have proposed a novel key-renewal scheme for wireless sensor networks. It can be divided into two parts: the server part and the sensor-node part. The server is responsible for generating appropriate key-generation functions under the power consumption constraint, and distributing them to sensor nodes. On the other hand, sensor nodes are responsible for electing headers, re-building the key-generation functions and actually renewing their keys. It is a trade-off between security and power consumption in wireless sensor networks.

Acknowledgement

This research was supported by the National Science Council of the Republic of China under contract NSC-94- 2213- E- 005- 028.

References

[1] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J. Tygar, "Security Protocols for Sensor Networks," *The Seventh ACM Annual International Conference on Mobile Computing and Networking*, pp.189-199, 2001.

[2] A. Price, K. Kosaka and S. Chatterjee, "A Key Pre-distribution Scheme for Wireless Sensor Networks," *The Wireless Telecommunications Symposium*, pp. 253-260, 2005.

[3] E. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks," *The Seventh International Workshop on Security Protocols*, pp.172-182, 1999.

[4] H. Chan, A. Perrig and D. Song, "Random Key Pre-distribution Schemes for Sensor Networks," *The IEEE Symposium on Security and Privacy*, pp. 197-213, 2003.

[5] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang, "Self-Securing Ad Hoc Wireless Networks," *The Seventh IEEE Symposium on Computers and Communications*, pp. 567-574, 2002.

[6] C. L. Wang, G. B. Horng, Y. S. Chen, and T. P. Hong, "An Efficient Key-update Scheme for Wireless Sensor Networks," *The International Conference on Computational Science*, pp. 1026-1029, 2006.

[7] G. Y. Lee and Y. Lee, "Efficient Rekey Interval for Minimum Cost on Secure Multicast System Using Group Key," *The IEEE Global Telecommunications Conference*, pp.1995-1999, 2002.