Dynamic Key Management Scheme in IoT

Po-Wen Chi[1(✉)] and Ming-Hung Wang[2]

[1] Department of Computer Science and Information Engineering, National Taiwan Normal University, Taipei, Taiwan, R.O.C.
neokent@gapps.ntnu.edu.tw

[2] Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan, R.O.C.
mhwang@fcu.edu.tw

Abstract. While IoT becomes more and more popular, security becomes an important issue when IoT deployment. Considering there are lot of mobile device, it is frequent for member joining and leaving. Therefore, traditional key agreement schemes are not suitable for dynamic IoT environments. In this paper, we propose a dynamic key management scheme to avoid key update overhead when membership changing.

Keywords: Internet of Things (IoT) · Group key management

1 Introduction

IoT (Internet of Things) has enabled a large number of connected devices to communicate with each other inside a private network, e.g., factory, farm, school, etc. These devices transit sensor records, machine status, and even sensitive data. Thus, security issues raised in communication between devices have become crucial and need to be addressed. Currently, many systems have leveraged the public key infrastructure to achieve a secure data transmission. However, in a private IoT, the computation power may be insufficient to support the key management of such a large number of individual machines with different keys. Moreover, there are more and more wearable devices and these devices move around with people. So we need a new key management scheme to support the dynamic member environment.

In this study, we propose to a dynamic key management scheme for IoT. We divide sensors into three groups. One group is for mobile sensors and other two groups are for static sensors. The difference between these two static sensors is their computational power. For example, when a traveler checks in a hotel, he may get a lightweight device which can access other hotel facilities. The device belongs to the first group. When the user enters a room, the device can access other room's field agents through a control unit. The control unit belongs to group 2 and field agents belong to group 3. In this paper, we move most heavy computational works to the static and powerful node to support other

C.-Y. Chang et al. (Eds.): ICS 2018, CCIS 1013, pp. 559–566, 2019.
https://doi.org/10.1007/978-981-13-9190-3_62

power constrained nodes' membership changing. We show that this scheme can decrease key update requirements even under frequent group 1 node handover events.

2 Background and Related Work

In this section, we briefly introduce the background techniques used in our scheme and some related works in this topic.

2.1 Blom's Key Pre-distribution Scheme [3]

Blom proposed a key pre-distribution scheme that allows any two nodes to find a secret key between them. Given total n nodes, one node does not require to store $n-1$ keys for other nodes. Instead, the node only takes $O(\lambda)$ memory space where λ is much smaller than n. The trade-off is that Blom's scheme is not fully resilient against the node capture attack. If an attacker compromises more than λ modes, the attacker can crack the system can get the pairwise key between any two nodes. This is called λ-secure. Blom's key pre-distribution scheme includes three phases which are briefly introduced as follows.

1. **Environment Setup.** Given total node number n, the trusted authority, which will be called **key server** later, first creates a generating matrix G of size $(\lambda+1) \times n$. λ is a security parameter described above. This matrix G is public information. Any $\lambda+1$ columns of G must be linearly independent. This can be done through a Vandermonde matrix[1]. The key server first selects a prime q where $q > n$ and then randomly picks a primitive element s of $GF(q)$ to generate the matrix.
2. **Key Space Setup.** The key server randomly generates a symmetric matrix D of size $(\lambda+1) \times (\lambda+1)$ over $GF(q)$. D is kept secret and is not disclosed to any nodes. The key server calculates $A = (D \cdot G)^T$. For a node i, the key server distributes i-th row of A to the node.
3. **Pairwise Key Agreement.** Suppose node i and node j want to come out a pairwise secret key. Since D is symmetric, we can show that $A \cdot G$ is also symmetric. The proof is as follows.

$$A \cdot G = (D \cdot G)^T \cdot G = G^T \cdot D^T \cdot G = G^T \cdot D \cdot G = G^T \cdot A^T = (A \cdot G)^T.$$

Node i calculates $k_{i,j}$, which indicates the element located on i-th row and j-th column of the matrix $A \cdot G$. Note that node i can derive $k_{i,j}$ with i-th row of A. Node j calculates $k_{j,i}$ of $A \cdot G$ similarly. Since $A \cdot G$ is a symmetric matrix, we can have $k_{i,j} = k_{j,i}$ and therefore $k_{i,j}$ can be used as the pairwise key.

[1] In Blom's work, it is not required to use a Vandermonde matrix. Here we use the Vandermonde matrix for convenience and the storage issue.

In 2005, Du et al. applied this idea with a random key distribution idea to build a sensor key management scheme [4]. In this paper, we base Blom's key pre-distribution work to build a dynamic key management scheme for IoT. Because of the dynamic characteristic in IoT, λ-security is not acceptable. So we enhance Blom's work to support more than λ changes in IoT.

2.2 Related Works

There are lots of works regarding secure and efficient key management in IoT and sensor networks. When considering the membership changing issue, most researchers use group key management to update keys efficiently. Logical Key Hierarchy (LKH) [5] is one of the most common group key management scheme. LKH uses a tree structure to represent users and their own keys. The update process takes $O(\log n)$ messages for n users. One way function tree is another efficient group key management scheme based on the tree architecture [2]. Park et al. proposed a group key management based on Chinese Remainder Theorem so that one encryption key can be used for multiple decryption keys [6]. Abdmeziem et al. separate devices into several groups to reduce key update overhead [1]. Veltri et al. decrease membership changing overhead through time partitioning techniques [7].

Unlike the above techniques, in this paper we drop the group key idea and use dynamic key generation concept for applying dynamic IoT environments. So in our scheme, the key update process is not necessary.

3 Proposed Scheme

In this section, we will introduce our scheme and show how it works when handling membership changes. First, we give an overview of the proposed scheme. Then we introduce the attack model. In Sect. 3.3, we show how the dynamic key agreement works. Finally, we use the dynamic key agreement approach to build a IoT key management system.

3.1 Overview

Our proposed scheme is a two-tier key management architecture with three different roles, **user**, **broker** and **device**. The user is an entity that will move around and handover between brokers. The broker is an entity that is located in a fixed-position and will not join or leave the system. The broker is in charge of forwarding data between users and devices. Here the broker is semi-trusted which implies the broker cannot read the content of packets that it relays. The device is an entity that is also on a fixed position and belongs to some broker. When a user wants to communicate with a device, the user needs to transfer data to the broker first and the broker then sends data to the device. Each device belongs to different functional group, like the temperature sensor, the humidity sensor, the door lock, the IP camera and so on. All entities have their own unique

identity in their role groups. There is an additional entity called **Key Distribution Center (KDC)** which is in charge of key management. The overall system model is shown in Fig. 1.

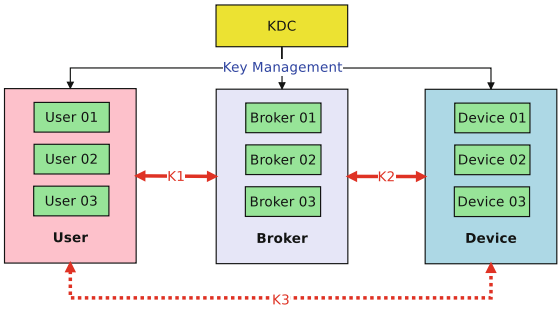


Fig. 1. System model.

In this model, there are three pairwise keys, K_1, K_2, K_3. They are used to protect the communication channels between the user and the broker, the broker and the device and the user and the device respectively. The dotted line in Fig. 1 implies the logical channel. When sending data to some device, the user will encrypt data through K_1 and K_3 and send the cipher to the broker. The broker then decrypts the cipher with K_1, re-encrypts with K_2 and forwards the new cipher to the device. The device finally derives data after decryption with K_2 and K_3.

Note that in this paper, the user also includes mobile devices, like wearable sensors.

3.2 Attacker Model

The attacker's goal is to get data access right without proper permission. For example, the attacker can be an outsider who is not allowed to get data but tries to decrypt data. Another example is that the attacker is a compromised device that attempts to get data which should be secret to it. In this case, undoubtedly, the compromised device can legally get data that it has right to access. So this case will not be considered as a successful attack.

3.3 Dynamic Pairwise Key Agreement Approach

In our system model, there will be three pairwise keys for three different channels. Here we just focus on the pairwise key agreement mechanism and the next subsection will show how this approach can be used to build a key management system in IoT. In this approach, we divide nodes into two groups and members belong to these two groups want to communicate with others. One group support dynamic membership while the other group is more static. There are five phases in this approach and are described below.

1. **Setup**(λ, m, n). Given two groups that have m and n members respectively. For simplicity, we call them group 1 and group 2. Let m is greater than n. KDC first creates a generating matrix G of size $(\lambda+1) \times (m'+n)$ where m' is much smaller than m. Any $\lambda+1$ columns of G must be linearly independent. KDC finds a transformation function ρ that maps from m to m'. We can use a hash function as the transformation function. KDC selects a prime q where $q > n$ and then randomly picks a primitive element s of $GF(q)$. The matrix will be:

$$G = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ s & s^2 & s^3 & \cdots & s^{m+n} \\ s^2 & (s^2)^2 & (s^3)^2 & \cdots & (s^{m+n})^2 \\ & & \vdots & & \\ s^\lambda & (s^2)^\lambda & (s^3)^\lambda & \cdots & (s^{m+n})^\lambda \end{bmatrix}.$$

   Each node only needs one element s to record G.
2. **Key Space Setup**. KDC randomly generates a symmetric matrix D of size $(\lambda+1) \times (\lambda+1)$ over $GF(q)$ with that all elements on the main diagonal are zeros. D is kept secret and is not disclosed to others. KDC also picks a pseudo random function σ. σ is secret to group 1 but is well known to group 2. KDC calculates $A = (D \cdot G)^T$. For node i in group 2, KDC distributes i-th row of A and σ to the node i. For a node j in group 1, KDC gets $(\rho(j)+n)$-th row of A as $\boldsymbol{v}$. Then KDC calculates $\boldsymbol{v'}$ as follows:

$$v'_k = v_k + \sigma(j,k) \cdot (s^{\rho(j)+n})^k, \forall k = 0, \ldots, \lambda.$$

   KDC distributes $\boldsymbol{v'}$ to the node j. This implies every node j in group 1 has its own unique secret matrix D_j where the element on the main diagonal is $\sigma(j,k), \forall k = 0, \ldots, \lambda$. Other elements are same with those in D. For simplicity, we use A_j to indicate $(D_j \cdot G)^T$.
3. **Pairwise Key Agreement**. Suppose node j in group 1 and node i in group 2 want to come out a pairwise secret key. Since D_j is symmetric, we can show that $A_j \cdot G$ is also symmetric:

$$A_j \cdot G = (D_j \cdot G)^T \cdot G = G^T \cdot D_j^T \cdot G = G^T \cdot D_j \cdot G = G^T \cdot A_j^T = (A_j \cdot G)^T.$$

   Node i calculates $k_{\rho(j)+n,i}$, which indicates the element located on $(\rho(j)+n)$-th row and i-th column of the matrix $A_j \cdot G$. As for node i in group 2, node i needs to calculate i-th row of A_j, $\boldsymbol{v}$', from i-th row of A, $\boldsymbol{v}$, as follows:

$$v'_k = v_k + \sigma(j,k) \cdot (s^i)^k, \forall k = 0, \ldots, \lambda.$$

   So node i can get $k_{i,\rho(j)+n}$ of the matrix $A_j \cdot G$. Because the matrix $A_j \cdot G$ is symmetric, $k_{i,\rho(j)+n} = k_{\rho(j)+n,i}$ and therefore node i and node j can share the same key.
4. **User Joining**. Here we focus on group 1 member joining. KDC assigns a new identity to the member and distributes the key space information for the new comer as described in the **Key Space Setup** phase. Note that existing deployed nodes do not need to do any changes.

5. **User Leaving**. Here we focus on group 1 member leaving. KDC simply broadcasts the node identity to the whole system. When the group 2 node receives the identity, it will simply add the identity to its revocation list. The group 2 member will reject the key agreement process with the group 1 member whose identity is on the revocation list.

In this pairwise key agreement approach, it is easily shown that computational cost required by the group 1 member is less than the group 2 member since the group 2 member needs additional $\lambda + 1$ pseudo random functions. The group 1 membership changing event costs almost nothing since the deployed environment does not require the key update process.

3.4 IoT Key Management Scheme

In our system model, there are three roles as shown in Fig. 1. In general, the user is dynamic while other two roles are static. That is, users join and leave frequently. As for the other two roles, they seldom changes once deployed. Besides, the broker is often more powerful than other two and the number of the user group is overwhelming.

K_1, K_2, K_3 are established from the approach described in Sect. 3.3. Note that there are three independent environments for three keys. According to the above characteristics, for K_1 and K_2, we make the broker in charge of the heavy computational work, which means the broker group is group 2 defined in the previous subsection. So the user and the device will not use too much computational power on pairwise key establishment. As for the K_3, since there are more users than devices, we prefer the device is group 2 and the user is group 1. To ease the device's computational burden, in our design, we make the broker to calculate $\sigma(j,k) \cdot (s^i)^k, \forall k = 0, \ldots, \lambda$ in the K_3 environment for the device. The broker attaches these elements to data when forwarding data to the device. That is, KDC puts σ of K_3 to the broker instead of the device. This can move some computational works to a more powerful entity. Note that the broker does not have any secret information D of K_3 and therefore it is impossible for the broker to derive the pairwise key K_3.

4 Evaluation

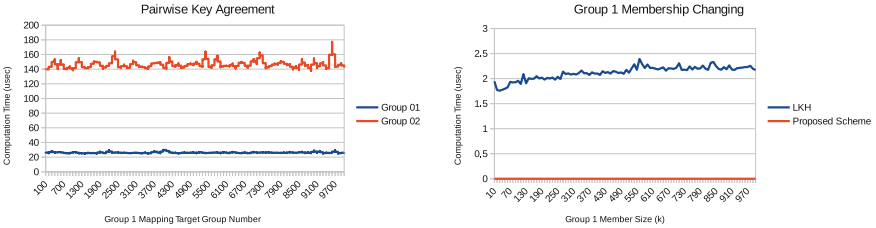
In this section, we discuss the computational overhead of our proposed scheme. We also discuss the probability of collusion attacks.

4.1 Computational Overhead

In this subsection, we will evaluate the computational costs required in our scheme. We implement our scheme on an Intel i7-7700 CPU including the user node, the group node and the device node. Though it is much more powerful than most IoT devices, here we just check the time difference. Since K_1, K_2, K_3 are from the same agreement scheme, here we just focus on K_1.

First, we check the pairwise key establishment part. Note that we do not care the environment setup phase and the key space setup phase run by KDC because KDC is much more powerful than other nodes and the setup process is infrequent. The evaluation result is shown in Fig. 2a. Group 1 is the user group and group 2 is the broker group. We can see that the group 1 node takes less computation time than the group 2 node. The reason is that group 2 node needs to recover the user's secret matrix from D. We use AES as our pseudo random function in our implementation. Note that the target group 1 number will not affect the computation time because it depends on $O(\lambda)$ instead of the mapping target group size.

Next, we compare the membership changing overhead in KDC between our method and LKH. The comparison result is presented in Fig. 2b. In the LKH scheme, when a node joins or leaves, existing nodes needs to update their keys related to the changing node. That is, given totally n nodes, KDC in LKH needs to encrypt $\log(n)$ times. As for our proposed scheme, KDC simply broadcasts the leaving node identity when a group 1 node leaves. For the node joining event, KDC distributes key agreement information to the newly coming node without informing existing nodes. That is, no encryption is required when membership changing. So the KDC computational cost in our scheme is almost zero.



(a) Pairwise key agreement. (b) Membership changing.

Fig. 2. Computational overhead.

4.2 Collusion Attack Analysis

In this subsection, we discuss how the collusion attack affect our key management system. There are three keys in our system and they are established through the same agreement approach. So here we just use K_1 as an example for analysis. Here are two groups here, the user group and the device group. Since any $\lambda + 1$ columns of the matrix G are linearly independent, an attacker must capture more than $\lambda + 1$ nodes with the same secret matrix D to recover D. However, since every node in the user group has its own secret, all colluded nodes cannot get secrets of other nodes.

Suppose one node in the broker group is compromised. That is, the secret pseudo random permutation function σ is released to an attacker. Because in our scheme, the user node identity is mapped to a smaller target group through a

mapping function ρ, compromising one user node implies compromising all user nodes that map to the same target. Given the target group size m' and a security parameter λ. Suppose m' is much smaller than m, which is the user group size. The probability that all K_1 pairwise keys are broken with n compromised user nodes and one compromised broker node is

$$P(n) = \begin{cases} 1 - \sum_{k=1}^{\lambda} \frac{1}{m'^n} (C_k^{m'} \cdot k^n), n > \lambda, \lambda < m' \\ \frac{1}{m'^n} \sum_{k=0}^{m'} (-1)^k \cdot C_k^{m'} \cdot (m' - k)^n, n > m', \lambda \geq m' \end{cases}.$$

5 Conclusion and Future Work

In this paper, we propose an IoT pairwise key management scheme that supports partially dynamic membership which means this system supports only some kind of devices joining or leaving. The overhead of membership changing costs less than other existing techniques. Considering some practical scenarios, we believe that the proposed scheme is suitable for the IoT environment.

Our next step will focus on the membership changing handling of the other group. Though in our construction, the proposed approach can support up to λ nodes leaving, it cannot support node joining[2]. Besides, λ may not be enough in a dynamic environment. We will try to solve this problem to make the scheme more applicable.

References

1. Abdmeziem, M.R., Tandjaoui, D., Romdhani, I.: A decentralized batch-based group key management protocol for mobile internet of things (DBGK). In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pp. 1109–1117, October 2015
2. Balenson, D., McGrew, D., Sherman, A.: Key management for large dynamic groups: One-way function trees and amortized initialization (1999)
3. Blom, R.: An optimal class of symmetric key generation systems. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 335–338. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39757-4_22
4. Du, W., Deng, J., Han, Y.S., Varshney, P.K., Katz, J., Khalili, A.: A pairwise key predistribution scheme for wireless sensor networks. ACM Trans. Inf. Syst. Secur. **8**(2), 228–258 (2005)
5. Harney, H.: Logical key hierarchy protocol. SMUG, March 1999
6. Park, M.H., Park, Y.H., Jeong, H.Y., Seo, S.W.: Key management for multiple multicast groups in wireless networks. IEEE Trans. Mobile Comput. **12**(9), 1712–1723 (2013)
7. Veltri, L., Cirani, S., Busanelli, S., Ferrari, G.: A novel batch-based group key management protocol applied to the internet of things. Ad Hoc Netw. **11**(8), 2724–2737 (2013)

[2] Of course, users can prepare a larger pool for new coming nodes.