IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 23, 2022, accepted May 18, 2022, date of publication May 23, 2022, date of current version June 1, 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3177270*

# Hyperledger Fabric-Based Lightweight Group Management (H-LGM) for IoT Devices

**JUHYUN MAENG, YOONNYOUNG HEO, AND INWHEE JOE**
Department of Computer Science, Hanyang University, Seoul 04763, South Korea

Corresponding author: Inwhee Joe (iwjoe@hanyang.ac.kr)

This work was supported in part by the Institute for Information and Communications Technology Promotion (IITP) funded by the Korean Government through the Ministry of Science, ICT (MSIT) and Future Planning (MSIP) (Development of the Technology to Automate the Recommendations for Big Data Analytic Models that Define Data Characteristics and Problems) under Grant 2020-0-00107, and in part by the National Research Foundation of Korea (NRF) funded by the Korean Government (MSIT) under Grant NRF-2019R1A2C1009894.

**ABSTRACT** Internet of Things (IoT) is a network that can communicate between devices without human intervention. In this network, IoT devices collect sensor data and provide them for various applications. However, data collected from the device may include sensitive information. If this information is leaked, several serious problems will occur. Therefore, access control is presented to allow only authorized users to access important data. This paper proposes a model for organizing and managing groups with the group key (GK) so that only the authorized users can share data for access control. In particular, rekeying overhead is reduced. Because the IoT devices are resource-constrained, the network lifetime may be reduced as the rekeying overhead increases. There is a problem when group communication between multiple users is not used. If multiple users can easily access all data, the leakage of sensitive information will increase. To solve this problem, only users who need to share data join in a group based on the hyperledger fabric. Our approach groups users with the same GK and protects sensitive data by using secure communication links within the group. In addition, a trusted agent for rekeying sends a new GK to users in the group. Therefore, the security of data is guaranteed, and the network lifetime is extended. We analyze the storage cost, delay, and processing time for rekeying, and we also compare them according to the number of users and the depth of the key tree. The results of the performance analysis show that the proposed hyperledger fabric-based lightweight group management (H-LGM) outperforms the existing method in terms of storage cost, delay, and processing time.

**INDEX TERMS** Group management, group key, rekeying, agent, hyperledger fabric, IoT device.

## I. INTRODUCTION

Internet of Things (IoT) is a network that can communicate between diverse devices without human intervention [1]. IoT can collect and exchange data and convert it into information [2]. In addition, IoT devices produce various data and share information by accessing the internet [3]. This IoT provides a wide variety of applications [4]–[7], it has changed our daily life considerably [1]. However, data collected from IoT devices may include sensitive information [4]. If information is leaked, several serious problems will occur [7], [8], Therefore, access control is presented to protect important data from unauthorized users [9]. Access control of important data may be performed using a secure communication link between users in the group. Where a pre-agreed key is required for the communication link [10]. The pre-agreed key is held only by users in the group. This key is the group key (GK). Only users with GK can access important data. GK is generated by the trusted agent among the users and transmitted to users in the group. Therefore, securing the reliability of the agent that makes this GK is very important. This is because if the agent is malicious, the agent sends GK to an external user, so important data cannot be protected from unauthorized users. In this paper, in order to secure the reliability of GK and prevent its leakage, we secure the reliability of the agent by applying the open-source hyperledger fabric [11]. Authentication can be effectively implemented using MSP,[1] an element of hyperledger fabric [12]. The proposed hyperledger fabric-based lightweight group management (H-LGM) generates a GK that manages

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaolong Li.

[1] https://hyperledger-fabric.readthedocs.io/en/release-2.2/msp.html

the group through an agent previously authenticated by the MSP. Since the agent is certified by the MSP, the reliability of GK is secured and its leakage may be prevented. Whenever each user leaves or joins, the agent rekeying and sends the key to users in the group. Data is then shared between users in the group. As a result, important data may be protected from unauthorized users. Also, in order to extend the lifetime of a network composed of IoT devices, rekeying overhead must be reduced. This is because IoT devices are seriously resource-constrained [13] If the storage cost, delay, and processing time increase for rekeying in this device, the network lifetime may be reduced. The proposed lightweight rekeying reduces the key tree depth of the GK to reduce the update overhead of the GK. As a result, the lifetime of the network composed of the resource-constraint IoT device may be extended.

Under these considerations, we implement H-LGM to protect important data from unauthorized users. In addition, lightweight rekeying was studied to extend the lifetime of the network consisting of resource-constrained IoT devices. Our contributions can be summarized as follows:

- We propose a lightweight rekeying to reduce the update overhead of GK and H-LGM to manage access control of important data.
- We use GK to organize and manage groups. Because if users use a secure communication link within a group while authenticating with the same GK between users, access control of important data can be performed.
- We use an agent for lightweight rekeying. Because if the agent re-generates the GK and sends it to all users in the group to reduce the rekeying overhead, storage cost, delay, and processing time can be reduced. Also, in order to secure the reliability of the agent, it is authenticated by the MSP, an element of hyperledger fabric, and registered in a private network.
- In performance evaluation, we demonstrate the effectiveness of H-LGM. Since storage cost, delay, and processing time according to rekeying are reduced compared to the existing method, the lifetime of the network composed of IoT devices may be extended.

The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces the hyperledger fabric-based lightweight group management (H-LGM). Section 4 analyzes various cases of the H-LGM in terms of storage cost, delay, and processing time. Section 5 presents the limitations of the study. Finally, Section 6 concludes the paper.

## II. RELATED WORK

Recently, Internet of Things (IoT) devices have been used in various areas [14]. Since IoT devices in diverse fields may communicate with each other, important data may be leaked to any user. Therefore, it is necessary to protect important data by forming a group between IoT devices. Under these considerations, several studies have been conducted to enhance data security. In [15], the author proposed a blockchain-based authentication and dynamic

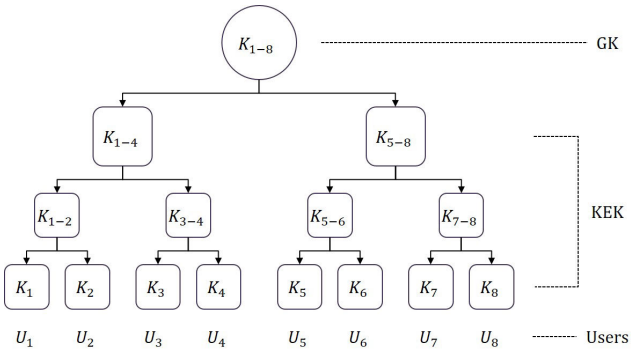



**FIGURE 1.** The key tree of the logical key hierarchy scheme [20].

group key agreement protocol to improve some shortcomings in the existing group key agreement protocol. Authentication of each member can be performed by authenticating the left neighboring member. The literature [16] considers group sensor communication protocols to address various vulnerabilities in wireless network communication. This protocol performs basic arithmetic and logical operations for sensor node authentication and data transmission. In [17], group key management (GKM) for the management of multiple devices was studied. The proposed GROUPIT is a two-tier GKM architecture. In GROUPIT, each device is included in a pre-determined group, and key management is performed within or between each group. [18] studies group key management (GKM) protocols for dynamic IoT environments. In this protocol, device groups can join or leave the network, also user groups may be created and dismantled. In [19], the author proposed master-key-encryption-based multiple group key management (MKE-MGKM) for multiple multicast groups. MKE-MGKM uses a master key and multiple slave keys. As shown in Fig. 1, key trees can be used to manage groups. In this logical key hierarchy (LKH),[2] the user owns a key on the path from leaf node to root node when in a group [20], and if a new user joins the group or an existing user leaves, all keys on the path are updated [21]. In addition, all users in the group share the group key [22]. There are also one-way function tree (OFT) and LKH++, which manage groups through the key tree. In the OFT, each node v has a node secret $x_v$ and a node key $k_v$. If the key tree changes, the node secret $x_v = \mathrm{f}(x_l) \oplus \mathrm{f}(x_r)$ and node key $k_v = \mathrm{f}(x_v)$ from node v to root node are updated. The node secret of the updated root node is used as the group key. where l and r are left and right children of node v. F is one-way functions and $\oplus$ is bitwise exclusive-or [23]. In addition, in LKH++, the user has a key from leaf node to root node. If a member of the group changes, the key in the path to the root node is updated. The group is managed with the key of this root node [24], [25]

Since IoT devices are resource-constrained, it is necessary to reduce storage cost, delay, and processing time according

[2]http://cgi.di.uoa.gr/~halatsis/Crypto/Bibliografia/Crypto_Lectures/Stinson_lectures/lec20.pdf

to rekeying. Therefore, this paper proposes hyperledger fabric-based lightweight group management (H-LGM).

## III. PROPOSE METHOD

### A. SYSTEM MODEL

In this paper, we assume that, as shown in Fig. 2, the proposed system consists of the fabric-CA root server, hyperledger fabric, and a number of users expressed by $U_i, \forall_i \in \{0, 1, 2, \ldots, K\}$ Hyperledger fabric consists of membership service providers (MSP), agent and fabric-certificate authority (CA) intermediate server including enrollment (E)CA and transport layer security (TLS) CA. Under this configuration, the procedure for registering and verifying any user in the group as an agent of the system is as follows. The Fabric-CA intermediate server requests verification from fabric-CA root server. Fabric-CA root server verifies hyperledger fabric and then issues a certificate to the fabric-CA intermediate server. Next, the MSP requests verification from the fabric-CA intermediate server. Fabric-CA intermediate server verifies the MSP and then issues a certificate to the MSP. As a result, preparations for registration and verification of the agent are completed. When the agent requests registration from the fabric-CA intermediate server, the (E) CA of the Fabric-CA intermediate server verifies the agent and then issues the certificate to the MSP and the agent. Also, the (TLS) CA issues a certificate to the MSP and the agent to establish a communication link between the MSP and the agent. Finally, the MSP verifies the certificate owned by the agent, and each user in the group has a shared key (PK) expressed by $PK_i, \forall_i \in \{0, 1, 2, \ldots, K\}$ Each user sends a PK to the agent when requesting to join the group. This PK is used when the agent is rekeying; therefore, multiple groups can be created and operated simultaneously. Agents in each group manage the group after completing registration and verification as described above. MSP records verified agnet in block via orderer. By confirming the records periodically, the status of the agent is monitored. In addition, the agent for each group is periodically changed. This is to stably manage the rekeying of each group.[3]

As shown in Fig. 3, the agent authenticated by the MSP manages the group. GK is generated to manage the group and then sent to all users in the group. All users with GK communicate with each other using a secure group channel.

A reliable agent should be selected because the agent performs an important role in managing the group. Therefore, as shown in Fig. 4, the selected agent is verified by the fabric-CA intermediate server and MSP. In order to issue a certificate for the MSP, the MSP generates a shared key. Next, a certification request for MSP is generated, including information on organizations, organizational units, cities and regions, states, provinces, and countries within the MSP. After the created shared key and authentication request are sent to the CA, the CA issues the certificate and sends it to the MSP. This procedure is implemented in the same way

[3]https://en.wikipedia.org/wiki/Single_point_of_failure#cite_note-1

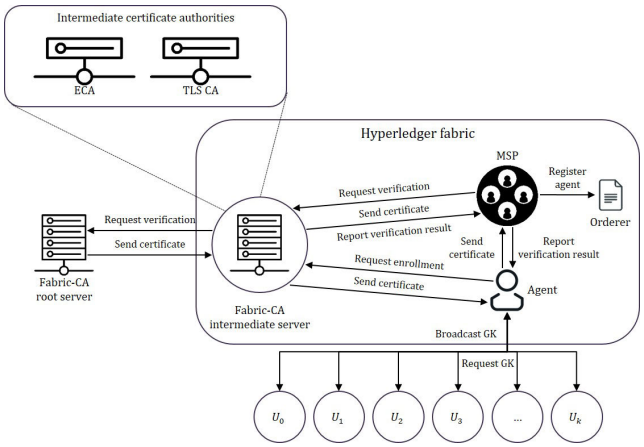



**FIGURE 2.** General system model: Hyperledger fabric-based lightweight group management (H-LGM).

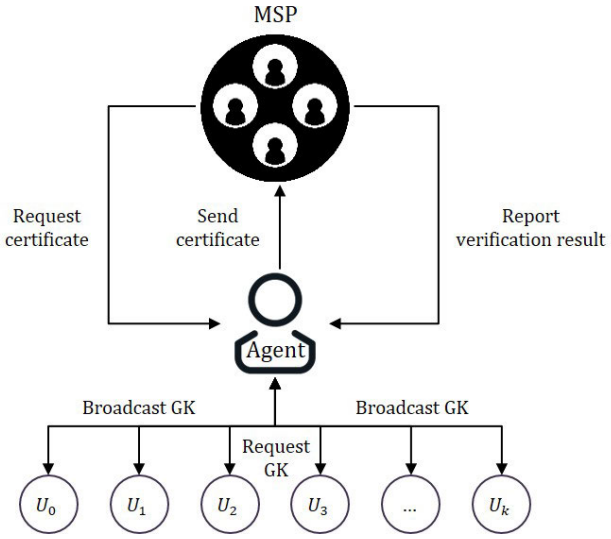



**FIGURE 3.** Core system model: Hyperledger fabric-based lightweight group management (H-LGM).

for issuing certificates for other nodes. MSP and orderer already receive a certificate from fabric-CA intermediate server and are mutually authenticated. The agent requests a certificate from the fabric-CA intermediate server that can prove itself. After verifying the agent, the fabric-CA intermediate server generates a certificate for the agent and sends it to the agent. The agent then requests registration from the MSP. The MSP requests a certificate from the agent, which receives the request and sends the certificate issued by the fabric-CA intermediate server to the MSP. Upon receipt of the certificate, the MSP confirms the previous registration and integrity of the certificate. If there is no problem with the registration of the certificate, MSP requests the certificate from the fabric-CA intermediate server that issued the certificate to the agent. The fabric-CA intermediate server sends the certificate requested by the MSP, and the MSP confirms the previous registration and integrity of the certificate as before. The MSP then confirms the verification

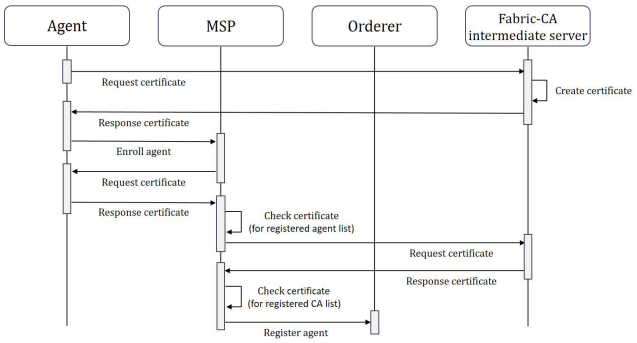



**FIGURE 4.** Flow chart: The user leaves the group.

of the agent through the received certificate and notifies the orderer of the result. Registered and validated agent manages the group. The MSP records the agent in the block through the orderer.

The following procedure is required for the user to receive the GK from the agent. The user sends a message to the agent that includes random numbers, session ID, and cipher suite information generated by the user. Upon receiving it, the agent transmits the random numbers and session ID information generated by the agent to the user. The agent then sends information to the user for the exchange of his certificate and key, and the agent requests certificate of the user. The user sends information to the agent for the exchange of certificate and key, and sends information about the verified certificate to the agent. The user and agent update the existing information by exchanging updated parameters in a series of processes, so that communication between the user and the agent becomes reliable.[4]

### B. GROUP MANAGEMENT

In this section, we introduce the proposed hyperledger fabric-based lightweight group management (H-LGM). When a user leaves or joins a group through H-LGM, a procedure in which GK is re-generated and the group is re-established is described.

#### 1) USER LEAVE

As shown in Fig. 5, $U_5$ leaves the existing group. First, $U_5$ sends $PK_5$ to the agent and requests to leave the group. The agent confirms the request from $U_5$, discards the $PK_5$ of $U_5$, and then informs $U_5$ of the result. Next, GK is re-generated with the $PK_{0\sim4}$ of $U_{0\sim4}$ in the existing group and the new random string. GK is transmitted to $U_i, \forall_i \in \{0, 1, 2, 3, 4\}$in the group.

Except for $U_5$ which left the group, the agent re-generate GK with the $PK_{0\sim4}$ of $U_{0\sim4}$ and the new random string. It is expressed as follows.

$$GK = SHA - 256(PK_0 + PK_1 + PK_2 + PK_3 + PK_4 + randomstring) \quad (1)$$

[4]https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Math/random

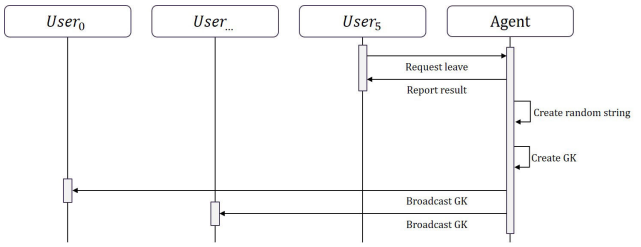



**FIGURE 5.** Flow chart: The user leaves the group.

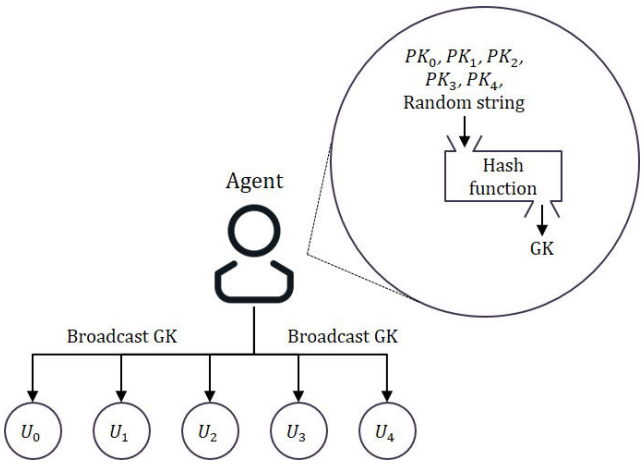



**FIGURE 6.** The user leaves the group.

The random string is 15 random numbers. It is randomly selected between English alphabets and numbers from 0 to 9. There are 62 in total and are listed in order. The random string is generated as follows. First, the randomly selected number between 0 and 1 is multiplied by 62. Then the characters in the order corresponding to the resulting value are selected, which is repeated 15 times. Since the rekeying includes a random string, it is possible to reduce the possibility that the leaked or lost GK and the re-generated GK are the same. Here, SHA-256 represents a hash function, as shown in Fig. 6, a new GK is obtained by the input values $PK_{0\sim4}$ and a random string.

#### 2) USER JOIN

As shown in Fig. 7, $U_5$ joins the existing group. First, $U_5$ sends the $PK_5$ to the agent and then requests a group join. The agent confirms the request from $U_5$ and requests $PK_5$ from $U_5$. $U_5$ sends the $PK_5$ to the agent and waits. The agent re-generates GK with $PK_{0\sim4}$ of $U_{0\sim4}$ and $PK_5$ of $U_5$ in the existing group and a new random string. Then, the GK is provided to the $U_i, \forall_i \in \{0, 1, 2, 3, 4, 5\}$ in group.

The agent re-generate the GK with the $PK_5$ of the new $U_5$, the $PK_{0\sim4}$ of the existing $U_{0\sim4}$, and the new random string, and it is expressed as follows.

$$GK = SHA - 256(PK_0 + PK_1 + PK_2 + PK_3) \quad (2)$$

As shown in Fig. 8, a new GK is obtained by inputting $PK_{0\sim5}$ and random string.

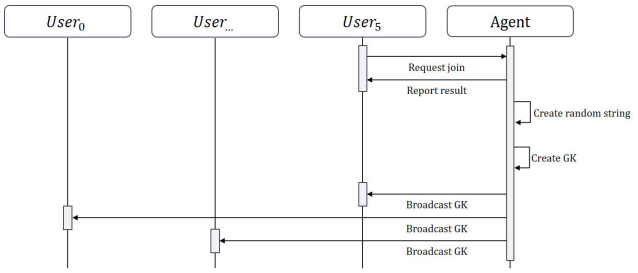



**FIGURE 7.** Flow chart: The user joins the group.

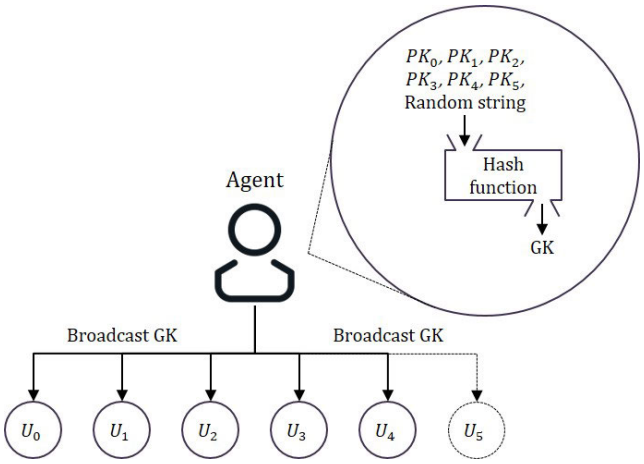



**FIGURE 8.** The user joins the group.

**TABLE 1.** Hardware and software environments.

| | |
|---|---|
| Hardware | |
| CPU | AMD Ryzen 7 1700 3.0GHz |
| Memory | 32GB |
| Software (Vmware) | |
| Virtual machine | Workstation 16 |
| OS | Ubuntu 16.04.7 LTS |
| CPU | 4 CORE |
| Memory | 8GB |
| Hyperledger fabric | Version 2.3.2 |
| Node | Version 14.16.1 |

## IV. PERFORMANCE EVALUATION

In this section, we show the results of analyzing the proposed hyperledger fabric-based lightweight group management (H-LGM). The environment in which H-LGM is implemented is listed in Table 1.

The value used in the experiment is the same as TABLE 2, and each value is actually measured in the implemented environment. The results are calculated using generalization equations from TABLEs 3 to 6, and the mean values of 50 times are shown as graphs. The generalized equation uses $h \approx \log_d(n)$ of LKH [26].

Where, u is the number of all users in the group, and d represents the depth of the key tree for rekeying, and $k_l$ is the size of a key for rekeying, and $t_r$ is the time to make random strings. Also, $t_e$ is a time for generating GK by PKs of all users in the group and random string, $t_x$ is the time for XOR operation. $t_d$ is the time for decryption, $t_b$ represents the time at which all users in the group receive GK.

**TABLE 2.** Simulation parameter's values.

| Symbol | Value | Description |
|---|---|---|
| $u$ | Variable | Number of users |
| $d$ | Variable | Key tree depth |
| $k_l$ | 32 bits | Key length |
| $t_r$ | 0.0389 msec | Generation time of random string |
| $t_e$ | 0.1396 msec | Encryption (SHA-256) time |
| $t_x$ | 0.0510 msec | XOR operation time |
| $t_d$ | 0.1396 msec | Decryption (SHA-256) time |
| $t_b$ | 0.1134 msec | Broadcasting time |

**TABLE 3.** Storage cost when the user leaves the group.

| | H-LGM | LKH | OFT | LKH++ |
|---|---|---|---|---|
| Client | $2k_l$ | $(\log_2 u+1)k_l$ | $(\log_2 u+1)k_l$ | $(\log_2 u+2)k_l$ |
| Server | $k_l$ | $(2\log_2 u-1)k_l$ | $(2\log_2 u-1)k_l$ | $(\log_2 u+1)k_l$ |

**TABLE 4.** Storage cost when the user joins the group.

| | H-LGM | LKH | OFT | LKH++ |
|---|---|---|---|---|
| Client | $2k_l$ | $(\log_2 u+1)k_l$ | $(2\log_2 u+1)k_l$ | $(\log_2 u+2)k_l$ |
| Server | $k_l$ | $(2\log_2 u)k_l$ | $(2\log_2 u+1)k_l$ | $(\log_2 u+1)k_l$ |

From TABLE 3 to TABLE 6, the client represents the user. The server for H-LGM is an agent, and the server for LKH is a key distribution center (KDC). Also, the server for OFT is a manager, and the server for LKH++ is a center.

When a user leaves or joins a group, the storage cost of the client and the server by the model are calculated by using the generalized equation, as shown in TABLE 3 and 4. First, when a user leaves or joins a group, in H-LGM, the client stores GK and PK, and the server stores only GK. Next, when one user leaves or joins a group, in LKH, the client stores d KEK and one GK. However, the storage cost according to the leave and join of the server is different. During leave, 2(d-1) KEKs and one GK are stored, and during a join, 2d-1 KEKs and one GK are stored. In OFT, if a user leaves or joins a group, the client stores unblended key and blind node secrets of d+1 and unblended key and blind node secrets of 2d+1 along the path to root, respectively. The server stores 2d-1 and 2d+1 respectively. In LKH++, if a user leaves or joins a group, the client stores key, private key, and random number of d+2 along the path to the root. The server stores d+1.

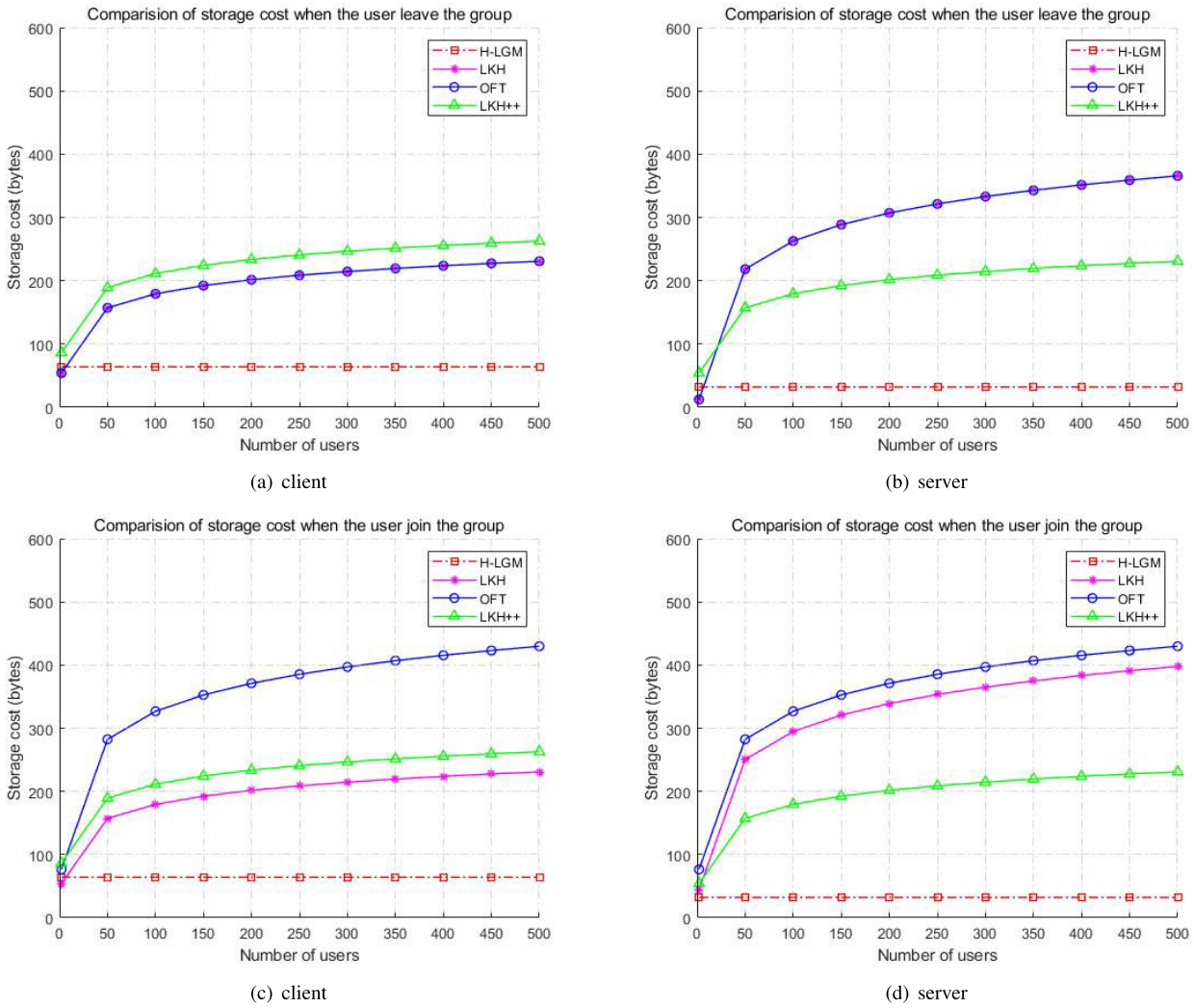



**FIGURE 9.** Comparison of storage cost when the user leaves and joins the group.

### A. STORAGE COST

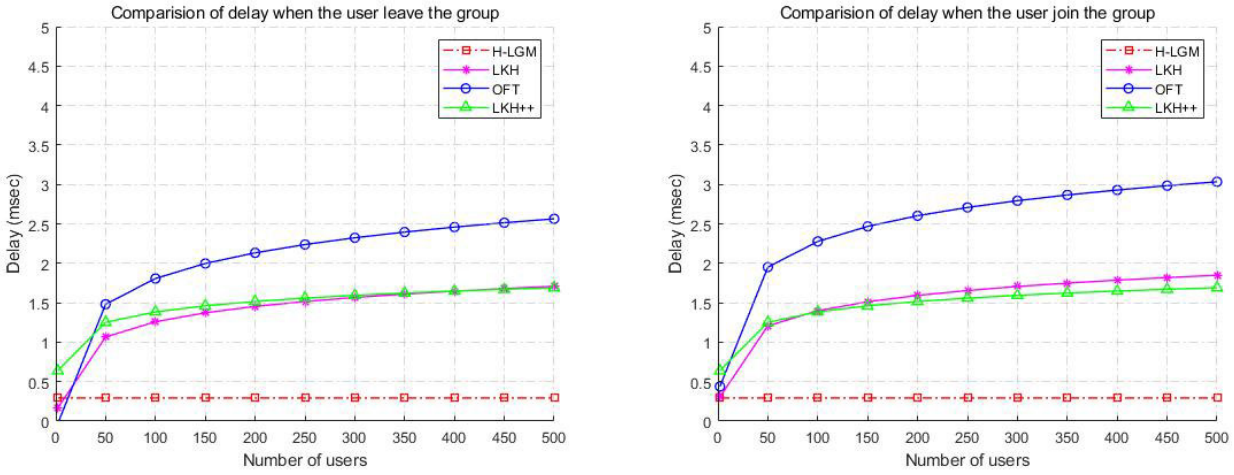
Fig. 9 is the result of comparing the storage cost of the client and server. Each (a), (b), and (c), (d) represent the result of the storage stored by each user for each rekeying according to the number of users when one user leaves and joins a group. H-LGM reduces storage than the existing method. This is because, when rekeying, each user only has an individual PK and a new GK, and an agent only has a re-generated GK. Since the storage stored by each user and agent is reduced compared to the existing method, the network lifetime consisting of resource-constraint IoT devices can be extended. When a user leaves or joins a group, the delay of the client for each model is calculated by a generalized equation, as shown in TABLE 5. In H-LGM, when a user leaves or joins a group after the agent completes encryption, users receive GK. However, the delay according to the leave and join of the KDC is different. At the time of leave, after 2d-1 encryption is completed, users receive GK. During join, after completing 2d encryption, users receive GK. In OFT, if a user leaves, the users receive GK after 3(d-1) encryption and d-1 xor operations, if a user joins, 3d encryption and d xor operations, and obtains GK. In LKH++, if a user leaves or joins, the users receive a GK after d+1 encryption, one decryption, and d xor operations.

**TABLE 5.** Delay of client.

| | H-LGM | LKH | OFT | LKH++ |
|---|---|---|---|---|
| Leave | $t_r+t_e$ $+t_b$ | $(2\log_2 u\text{-}1)t_e$ $+t_b$ | $3(\log_2 u\text{-}1)t_e$ $+(\log_2 u\text{-}1)t_x+t_b$ | $t_e+t_d+\log_2 u(t_e+t_x)$ $+2t_b$ |
| Join | $t_r+t_e$ $+t_b$ | $(2\log_2 u)t_e$ $+t_b$ | $\log_2 u(3t_e+t_x)+t_b$ | $t_e+t_d+\log_2 u(t_e+t_x)$ $+2t_b$ |

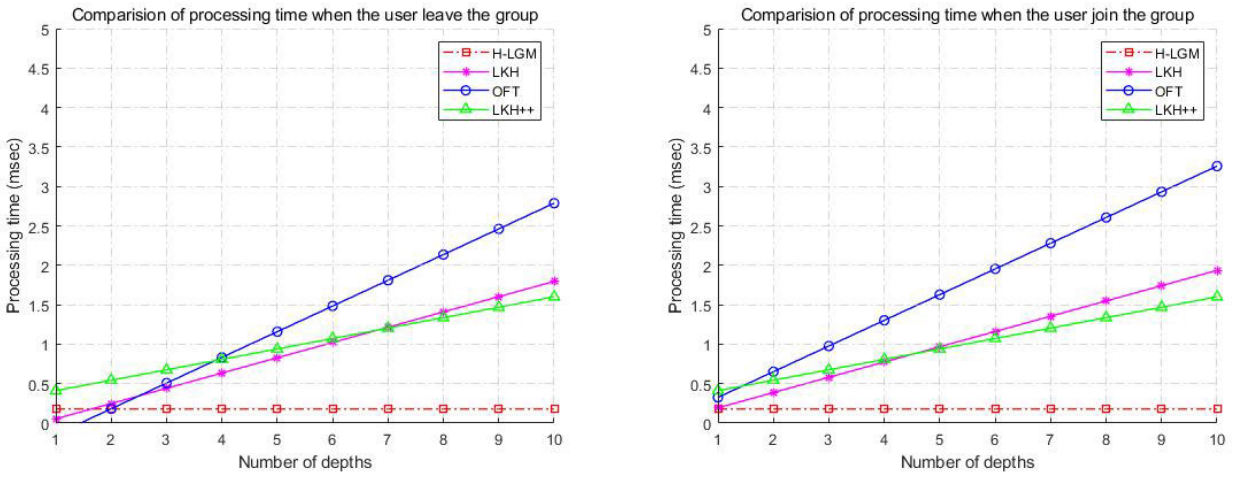
### B. DELAY

Fig. 10 is the result of comparing the delay of clients. (A) and (b) show the results of the time each user receives a new GK after rekeying according to the number of users when




(a) client (b) client

**FIGURE 10.** **Comparison of delay when the user leaves and joins the group.**




(a) server (b) server

**FIGURE 11.** **Comparison of processing time when the user leaves and joins the group.**

a user leaves and joins a group. H-LGM reduces delay than existing method. This is because the agent reduced the update overhead of the GK by re-generating the GK and transmitting it to all users. Since each user has reduced the time to receive a new GK compared to the existing method, faster group communication will be possible.

When a user leaves or joins a group, the processing time of the server by the model is calculated by the generalized equation, as shown in TABLE 6. In H-LGM, when a user leaves or joins a group, the server performs one encryption. However, the processing time according to the leave and join of the KDC is different. When leaving, 2(d-1) encryption is performed, and when joining, 2d encryption is performed. In OFT, if one user leaves, 3(d-1) encryption and d-1 xor operations are performed, and if one user joins, 3d encryption and d xor operations are performed. In LKH++, if one user leaves or joins, d+1 encryption, one decryption, and d xor operations are performed.

**TABLE 6.** **Processing time of server.**

| | H-LGM | LKH | OFT | LKH++ |
|---|---|---|---|---|
| Leave | $t_r+t_e$ | $(2\log_2 u\text{-}1)t_e$ | $3(\log_2 u\text{-}1)t_e$ $+(\log_2 u\text{-}1)t_x$ | $t_e+t_d+\log_2 u(t_e+t_x)$ |
| Join | $t_r+t_e$ | $(2\log_2 u)t_e$ | $\log_2 u(3t_e+t_x)$ | $t_e+t_d+\log_2 u(t_e+t_x)$ |

### C. PROCESSING TIME

Fig. 11 is the result of comparing the processing time of the server. (A) and (b) show the results of the overhead of the server for each rekeying according to depth when a user leaves and joins a group. H-LGM reduces the processing time

compared to existing method. This is because the depth of the key tree and the update overhead of GK were reduced. As a result, the processing time for agents is reduced compared to the existing method, so lightweight group management can be implemented.

## V. LIMITATIONS OF THE STUDY

Our research has several limitations. First, it is necessary to apply hyperledger fabric-based lightweight group management (H-LGM) to IoT devices and prove its effectiveness. This is because the IoT is a promising technology that can change the everyday life of people [27]. However, IoT devices are resource-constrained [28], in order to extend the network lifetime, resource consumption of the device must be reduced. Therefore, it contributes to the implementation of various applications by proving that the resource constraint of the IoT device is improved through the proposed H-LGM. To this end, H-LGM is implemented in IoT development kits and resource consumption is evaluated. Next, it is necessary to consider the possibility that the same GK will be generated again. To create a new GK, H-LGM generates GK using random string and each PK. Since random strings can generate $62^{15}$ random numbers, it is difficult to re-generate the same GK. However, since GK performs a role in protecting important data, it is very important to manage GK. Therefore, the system stability is improved by further reducing the possibility of generating the same GK. To this end, an algorithm that generates more than $62^{15}$ random numbers is developed and performance is evaluated.

## VI. CONCLUSION

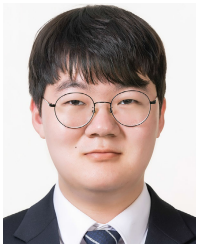
In this paper, we knew the problem of increasing the storage cost, delay, and processing time for the existing method that manages groups using GK. This problem is because as the number of users increases, the depth of the key tree is increased, and the rekeying overhead is increased. Therefore, we studied hyperledger fabric-based lightweight group management (H-LGM). H-LGM reduces the depth of the key tree and the rekeying overhead. In addition, it secures the reliability of the new GK. Organizing a group using H-LGM can protect important data from external users. As the number of users in existing method increases, the storage cost, delay, and processing time increase, so the network lifetime composed of IoT devices may be reduced. This is because IoT devices are resource-constrained. According to the number of users and the depth of the key tree, the storage cost, delay, and processing time of the proposed H-LGM are analyzed. Storage decreased because the key size of the user and agent was reduced for each rekeying. Also, delay and processing time was reduced because the rekeying overhead was reduced. Therefore, even if the number of users increases, hyperledger fabric-based lightweight group management (H-LGM) reduces storage cost, delay, and processing time, so it can be applied to resource-constraint IoT devices to ensure security of important data.

## REFERENCES

[1] K. Gulati, R. S. K. Boddu, D. Kapila, S. L. Bangare, N. Chandnani, and G. Saravanan, "A review paper on wireless sensor network techniques in Internet of Things (IoT)," *Mater. Today*, vol. 51, pp. 161–165, May 2022.
[2] B. Javed, M. W. Iqbal, and H. Abbas, "Internet of Things (IoT) design considerations for developers and manufacturers," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2017, pp. 834–839.
[3] H. Liu, D. Han, and D. Li, "Fabric-IoT: A blockchain-based access control system in IoT," *IEEE Access*, vol. 8, pp. 18207–18218, 2020.
[4] H. Lee, R. Chow, M. R. Haghighat, H. M. Patterson, and A. Kobsa, "IoT service store: A web-based system for privacy-aware IoT service discovery and interaction," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2018, pp. 107–112.
[5] M. Syafrudin, G. Alfian, N. Fitriyani, and J. Rhee, "Performance analysis of IoT-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing," *Sensors*, vol. 18, no. 9, p. 2946, Sep. 2018.
[6] M. Wazid, A. K. Das, R. Hussain, G. Succi, and J. J. P. C. Rodrigues, "Authentication in cloud-driven IoT-based big data environment: Survey and outlook," *J. Syst. Archit.*, vol. 97, pp. 185–196, Aug. 2018.
[7] H. Hejazi, H. Rajab, T. Cinkler, and L. Lengyel, "Survey of platforms for massive IoT," in *Proc. IEEE Int. Conf. Future IoT Technol. (Future IoT)*, Jan. 2018, pp. 1–8.
[8] X. Huang, R. Fu, B. Chen, T. Zhang, and A. W. Roscoe, "User interactive Internet of Things privacy preserved access control," in *Proc. Int. Conf. Internet Technol. Secured Trans.*, 2012, pp. 597–602.
[9] M. Dammak, S.-M. Senouci, M. A. Messous, M. H. Elhdhili, and C. Gransart, "Decentralized lightweight group key management for dynamic access control in IoT environments," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 3, pp. 1742–1757, Sep. 2020.
[10] N. Ferrari, T. Gebremichael, U. Jennehag, and M. Gidlund, "Lightweight group-key establishment protocol for IoT devices: Implementation and performance analyses," in *Proc. 5th Int. Conf. Internet Things, Syst., Manage. Secur.*, Oct. 2018, pp. 31–37.
[11] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.* New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–15.
[12] J. Jeong, D. Kim, S.-Y. Ihm, Y. Lee, and Y. Son, "Multilateral personal portfolio authentication system based on hyperledger fabric," *ACM Trans. Internet Technol.*, vol. 21, no. 1, pp. 1–17, Feb. 2021.
[13] Y. F. D. Rosário and X. Fafoutis, "A constrained monitoring protocol for the Internet of Things," *J. Signal Process. Syst.*, vol. 94, no. 1, pp. 45–64, Apr. 2021.
[14] M. P. K. Reddy and M. R. Babu, "Energy efficient cluster head selection for Internet of Things," *New Rev. Inf. Netw.*, vol. 22, no. 1, pp. 54–70, Jan. 2017.
[15] Z. Xu, F. Li, H. Deng, M. Tan, J. Zhang, and J. Xu, "A blockchain-based authentication and dynamic group key agreement protocol," *Sensors*, vol. 20, no. 17, p. 4835, Aug. 2020.
[16] H. Kim and J. Kang, "Dynamic group management scheme for sustainable and secure information sensing in IoT," *Sustainability*, vol. 8, no. 10, p. 1081, Oct. 2016.
[17] Y.-H. Kung and H.-C. Hsiao, "GroupIt: Lightweight group key management for dynamic IoT environments," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5155–5165, Dec. 2018.
[18] A. Kabra, S. Kumar, and G. Kasbekar, "Efficient, flexible and secure group key management protocol for dynamic IoT settings," *EAI Endorsed Trans. Internet Things*, vol. 7, no. 25, Apr. 2021, Art. no. 168862.
[19] M. H. Park, Y. H. Park, H. Y. Jeong, and S. W. Seo, "Key management for multiple multicast groups in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 9, pp. 1712–1723, Sep. 2013.
[20] N. Liu, S. Tang, L. Xu, and D. He, "Analyses of several recently proposed group key management schemes," *Secur. Commun. Netw.*, vol. 8, no. 2, pp. 136–148, Jan. 2015.
[21] J.-J. Lee, C.-Y. Huang, L.-Y. Lee, and C.-L. Lei, "Design and implementation of secure communication channels over UPnP networks," in *Proc. Int. Conf. Multimedia Ubiquitous Eng. (MUE)*, 2007, pp. 307–312.
[22] A. S. Pande and R. C. Thool, "Survey on logical key hierarchy for secure group communication," in *Proc. Int. Conf. Automat. Control Dyn. Optim. Techn. (ICACDOT)*, Sep. 2016, pp. 1131–1136.
[23] A. T. Sherman and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," *IEEE Trans. Softw. Eng.*, vol. 29, no. 5, pp. 444–458, May 2003.

[24] R. D. Pietro, V. L. Mancini, and S. Jajodia, "Efficient and secure keys management for wireless mobile communications," in *Proc. 2nd ACM Int. Workshop Princ. Mobile Comput.* New York, NY, USA: Association for Computing Machinery, 2002, pp. 66–73.
[25] W. Yao, S. Han, and X. Li, "LKH++ based group key management scheme for wireless sensor network," *Wireless Pers. Commun.*, vol. 83, no. 4, pp. 3057–3073, Aug. 2015.
[26] A. De Salve, R. D. Pietro, P. Mori, and L. Ricci, "A logical key hierarchy based approach to preserve content privacy in decentralized online social networks," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 1, pp. 2–21, Jan. 2020.
[27] A. P. Plageras, K. E. Psannis, C. Stergiou, H. Wang, and B. B. Gupta, "Efficient IoT-based sensor BIG data collection–processing and analysis in smart buildings," *Future Gener. Comput. Syst.*, vol. 82, pp. 349–357, May 2018.
[28] A. Nadeem, M. Hussain, A. Iftikhar, and S. Aslam, "Narrowband IoT device to device pairing scheme to save power," in *Proc. IEEE 23rd Int. Multitopic Conf. (INMIC)*, Nov. 2020, pp. 1–5.

**JUHYUN MAENG** received the B.S. degree in information communications engineering from Sejong University, Seoul, South Korea, in 2011, and the M.S. degree in electronic computer and communications engineering from Hanyang University, Seoul, in 2013, where he is currently pursuing the Ph.D. degree in software engineering. From 2013 to 2017, he was a Researcher with the Center of Human-Centered Interaction for Coexistence. His research interests include wireless sensor networks, wireless powered communication networks, blockchain, and non-terrestrial networks.



**YOONNYOUNG HEO** received the B.S. degree in smart system software engineering from Hyupsung University, South Korea, in 2021. He is currently pursuing the M.S. degree in software engineering with Hanyang University, Seoul, South Korea. From 2017 to 2020, he has participated in various IoT projects and application development projects. Since 2021, he has been working with a company in the field of computer vision. His research interests include computer vision, machine learning, and distributed systems.

**INWHEE JOE** received the B.S. and M.S. degrees in electronics engineering from Hanyang University, Seoul, South Korea, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 1998. Since 2002, he has been a Faculty Member with the Division of Computer Science and Engineering, Hanyang University. His current research interests include mobile internet, cellular systems, PCS, wireless sensor networks, mobile ad-hoc networks, multimedia networking, and performance evaluation.