Efficient Geometric Group Key-sharing Scheme

Koki Nishigami and Keiichi Iwamura
Tokyo University of science, 6-3-1, Niijuku Katsushikaku, Tokyo 125-8585, Japan

Keywords: Group Key, Key Sharing, Common Key Cryptography, IoT Network, Euclid Distance.

Abstract: In recent years, Internet of things (IoT) technologies that are used to collect various data have advanced significantly. In such networks, devices communicate with each other to exchange data. It is efficient for group-encrypted communication in IoT networks when devices in a group communicate with a common group key. However, if one device in the group is analyzed by an attacker, the group key is leaked. Therefore, a group key-sharing scheme that can change the group composition including a device deletion is required. Hamasaki et al. proposed a group key-sharing scheme using geometric characteristic. In this scheme, each device stores only the unique key and the key generates the coordinates for key sharing. The group composition can be changed easily using this scheme. However, the scheme incurs considerable computation cost when a large number of devices exist in the group. Therefore, we propose an efficient scheme that improves on Hamasaki et al.'s scheme for group key sharing and compare it with other methods.

1 INTRODUCTION

In recent years, Internet of things (IoT) has become fundamental for advancing businesses including e-business. In IoT networks, many devices (nodes) collect various data and send them to a base station for analysis. Generally, an IoT network comprises a base station (BS) and many nodes. A single BS is included in every network and serves as the center of the network. Many nodes are present in the network and they collect environmental data. A BS must be tamper resistant and include sufficient computational ability and power supply capacity because if the BS malfunctions, the network will be affected. Nodes are placed in physically unsafe places, and the number of nodes can be large in the network. Therefore, nodes are not tamper-resistant and CPU performance is poor, because the cost to nodes is minimized.

Communications in the IoT network include unicast, group, and broadcast communications. Unicast communication involves one-to-one device communication. Group communication involves three or more nodes. Broadcast communication involves all nodes in the network. Broadcast communication is considered as group communication because "all nodes in the network" are one of the groups in the network.

It is necessary to encrypt all communications in an IoT network to ensure a secure communication. The following factors highlight the importance of encrypted communication in IoT networks because the node performance can be poor, and efficient key sharing is critical for establishing encrypted communication in IoT networks:

- Low communication cost

  Communication cost must be low because of the considerable electric power required for communication, and each node has limited power storage.

- Low computational cost

  Large computational costs involving complex calculations such as the public key cryptosystem may not be suitable for IoT networks because of poor CPU performance of nodes.

- Low storage requirement

  It is possible to share keys using many pre-distributed element keys without using a unique key for each node. However, node storage requirements are expected to be low because memory storage for most nodes is low.

- Key renewal

  Encrypted communication using group keys for every communication group is efficient in IoT networks. However, it is desirable to renew a

Nishigami, K. and Iwamura, K.
Efficient Geometric Group Key-sharing Scheme.
DOI: 10.5220/0007965802090214
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 209-214
ISBN: 978-989-758-378-0

group key easily because if the same group key is used continuously, all communication leaks when an attacker analyzes a device, and the key can be leaked as well.

- Security

  It is important to ensure that an attacker analysis performed for one node does not leak the keys of other nodes.

Hamasaki et al. proposed a geometric method in which a group key is shared using the unique key of each node. In this scheme, node coordinates are generated based on the unique key of a node, and the center coordinate is calculated, which is equally distant from all node coordinates. The BS broadcasts the center coordinate to all nodes, and the distance between the center coordinate and every node coordinate is used as the group key. The method uses characters of circles and spheres where the center coordinate is equally distant from the circumferences of a circle and a sphere. The method realizes an efficient key renewal process. We evaluated this method based on the five factors defined above. The evaluation demonstrated good results except for computational cost when the number of nodes in a group is large. This can be attributed to the calculation of multidimensional coordinates according to the number of nodes. For example, a 999-times calculation for multidimensional coordinates is required when 1,000 nodes exist in the network.

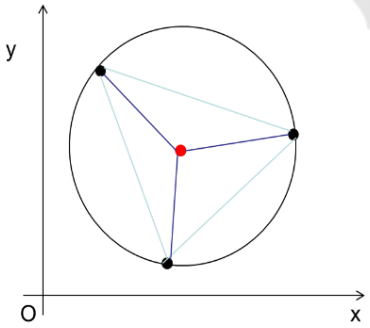

Figure 1: Central point of a circle.

Therefore, we propose an efficient scheme for group key sharing, in which the group is divided into clusters of three nodes each.

The related works are presented in Section 2. Section 3 provides a detailed description of the proposed scheme. Section 4 presents an evaluation of the proposed scheme, and Section 5 summarizes the conclusions of the paper.

2 RELATED WORKS

2.1 LEAP

The Localized encryption and authentication protocol (LEAP) was introduced to maintain secure key establishment by providing four keys per each node: individual key, group key, cluster key, and pairwise key. This scheme is based on a common key cryptosystem.

The individual key is shared with the BS, the group key with all nodes in the network globally, the cluster key with all the neighbor nodes jointly, the pairwise key with the direct neighbor nodes individually. In terms of LEAP, a cluster key is a key used for group communication, whereas a group key is a key used for broadcasting. Concept of the cluster key-sharing scheme used in LEAP is explained in detail in the subsequent paragraphs.

The pairwise key sharing is required to be established first; it is then followed by the cluster key sharing phase. The pairwise key is generated by exchanging IDs between each neighbor nodes and encrypting these IDs using the initial keys set in all nodes. The individual key is deleted after pairwise key-sharing phase is completed.

The cluster key-sharing process is established in a rather straightforward way. Let us consider the case where a node *u* attempts to establish a cluster key with all immediate neighbors *v*1, *v*2, …, *vm*. The node *u* first generates a random key Kcu as a group key, and subsequently encrypts this key with the pairwise key shared with each neighbor individually, followed by transmitting the encrypted key to each neighbor *vi* (i = 1, …, m). Therefore, the node *u* sends the encrypted key and the node ID *vi* to identify the addressee node. The node *vi* decrypts the group key Kcu and subsequently stores it. When one of the neighbors is revoked or added, the node *u* generates a new cluster key and transmits it to all the neighbors similarly.

2.2 ECCKSS

The ECC-based key-sharing scheme (ECCKSS) protocol is based on the elliptic curve cryptography (ECC) operation, namely, based on public key cryptosystems. This protocol consists of an initiator that wishes to share a group key and a responder that is a multicast group member. First, we present the steps of the initiator for sharing the group key.

1. The initiator decides the composition of the multicast group $U = \{U_1, U_2, U_3, \ldots, U_4\}$.

2. The initiator generates a random value *r* and computes R = rG (G is the base point generator with an order of p, which is a prime used in the elliptic curve (EC) parameters). The same R value can be reused.
3. The initiator calculates the EC points S_js (j = 1 to n-1) using r and the public keys Q_j of the group members: $S_j = d_jQ_j + R$, where d_j is its private key.
4. The initiator encodes the EC point $S_j = (x_j,y_j)$ into the point (u_j,v_j) as follows: $u_j = h(x_j)$; $v_j = h(y_j)$.
5. The initiator computes the values $\bar{u_j} = \{\bigoplus_{i\neq j} u_i\} \oplus v_j$, $\forall j \in \{1, \dots, n-1\}$ and denotes the set $P = (\bar{u}_1 || \dots \bar{u}_{n-1})$.
6. The initiator calculates the group key as $k = h(\bigoplus_i u_i)$.
7. The initiator calculates the Auth code as follows: $Auth = h(k||R||P)$.
8. The initiator transmits (Auth; C; R; U; P), where *C* is the counter value. Additionally, the digital signature is appended to preserve the message's authenticity and integrity.

Next, we present the steps of the other nodes for sharing the group key.

1. The responder verifies whether it is included in the multicast group U. Subsequently, the digital signature and counter C are verified.
2. If both are verified correctly, the responder computes S_j using the received random value R and the node's private key d_j: $S_j = d_jQ_i + R$.
3. The responder converts the EC point S_j to the point (u_j, v_j) using the same encoding as in Step 4.
4. The responder calculates the group key k: $k = h(\bar{u_j} \oplus u_j \oplus v_j)$.
5. The responder verifies whether $Auth = h(k||R||P)$. If this is verified correctly, then the group key *k* is authenticated.

Finally, we present the concluding step for sharing the group key:

1. Each sensor node should send an acknowledgment message $h(k, Q_j)$ to complete the handshake.
2. By verifying the acknowledgment message, the initiator can ensure the authenticity of the particular group member and the accurate derivation of the group key k.

2.3 SSKM

To maintain the channel secure for delivering shares, the secret sharing-based key management (SSKM) scheme adopts a discrete logarithm in the finite field under the decisional Diffie–Hellman difficulty assumption.

This scheme shares one cluster key (a group key) between the neighboring nodes. Below the detailed description of the scheme phases is presented.

[Initial phase]

Assume that $m-1$ clusters exist, and each cluster has a cluster head and n ($n \geq k$) member nodes. In this phase, the BS sets the parameters for key sharing. This process is discussed systematically in subsequent paragraphs.

Step 3-1:

The BS chooses two large primes: $p1$ and $q1$; let $p = 2p1 + 1$ and $q = 2q1 + 1$, $N = pq$; it is computationally intractable to solve the factor N without p and q. Meanwhile, the BS selects a generator g ($g \in [N^{1/2},N]$) and another prime Q ($Q > N$). Subsequently, it broadcasts the triple (N, g, Q) to the sensors in the network.

Step 3-2:

The BS randomly and uniformly chooses a polynomial *f(x)* of (k−1) degree for each cluster as follows:

$$f(x) = s + a_1x + a_2x^2 + \cdots + a_{k-1}x^{k-1}$$

Step 3-3:

The BS independently selects a session key K_{CH} from GF(Q) in the finite field Q and encrypts the session key Z_{CH} with the secret key S_{CH}, namely, $Z_{CH} = K_{CH} + S_{CH}$.

[Cluster key management]

Step 4-1:

The cluster head (CH) chooses its own key *xch* randomly, which is relatively prime with *p-1* and *q-1*, and sends it to the BS. Thereafter, the BS counts the $y_{CH} = g^{xch}$ and sends (ID_{CH}, y_{CH}) to a sensor node in the cluster; meanwhile, the sensor node selects a node's private key *xi* randomly, which is relatively prime with *p-1* and *q-1*, computes $yi = g^{xi} \bmod N$, and subsequently sends (*Indi, yi*) to the BS. The BS ensures that, if $IDi \neq IDj$, $yi = yj$ should not exist; otherwise, it reselects a node until it succeeds. Furthermore, the BS utilizes the CH's ID_{CH} and the members' *IDi (i = 1, ..., k)* to count the shares $f_{CH}(ID_{CH})$ and $f_{CH}(IDi)$.

Step 4-2:

The CH selects a group of users $Vl = \{ID1, \dots, IDk\}$, while the BS unicasts $(ID_{CH}, f_{CH}(ID_{CH}) \cdot (yi)^{xch}$

mod N) to a sensor node in the cluster and sends ($IDi, f_{CH}(IDi) \cdot (y_{CH})^{xi} \bmod N$) to the CH.

Step 4-3:

The BS informs Z_{CH} not to share the K_{CH}.

[Secret recovery]

Depending on the received information from the BS, e.g., the public generator, a node's private key x, and the CH's own key x_{CH}, the CH and members can obtain their shares through the following formulas:

$$\frac{f_{CH}(ID_{CH})(y_i)^{x_{CH}}}{(y_{CH})^{x_i}} = f_{CH}(ID_{CH}) \quad (1)$$

$$\frac{f_{CH}(ID_i)(y_{CH})^{x_i}}{(y_i)^{x_{CH}}} = f_{CH}(ID_i) \quad (2)$$

2.4 Hamasaki et al. Scheme

In this scheme, all nodes share the group key through a BS. First, the steps executed by the BS to perform the group key sharing are discussed. All nodes store each own unique key and the BS knows all unique keys. The BS and all nodes keys are calculated with *GF(p)*. The process is described in detail below:

1. The BS generates a pseudorandom number *r*, and the number *r* is shared with all nodes in the group *n*.
2. The BS generates a pseudorandom number based on each node's unique key and a random number *r* to set node's coordinates in *n-1* dimensions $a_i\ (a_{i1}, a_{i2}, a_{i3}, \cdots, a_{i(n-1)})$.
3. If all node coordinates are set in *n-2* dimensions, the BS returns to step 1.
4. The BS calculates the coordinates of the center of the circle using the coordinates of all nodes.
5. The BS broadcasts *r, n* and the coordinates of the center of the circle.

Next, the steps of sharing the group key for a node *i* are presented:

1. The node *i* calculates a pseudorandom number $a_i\ (a_{i1}, a_{i2}, a_{i3}, \cdots, a_{i(n-1)})$ based on a random number *r* and a node's unique key.
2. The node *i* calculates the distance from the received center of the circle to its own node coordinates using equation (3) presented below. However, we do not perform the square root operation because of its lower computational cost.

$$S = (o_1 - a_{i1})^2 + (o_2 - a_{i2})^2 + \cdots + \left(o_{n-1} - a_{i(n-1)}\right)^2 \quad (3)$$

3 PROPOSED SCHEME

We propose an efficient geometric group key-sharing scheme based on the Hamasaki et al. scheme.

3.1 Efficient Hamasaki et al. Scheme

The Hamasaki et al. scheme incurs a high computational cost if there are many nodes in a group because *n-1* times generating pseudorandom number is required (*n* is the number of nodes in the group). Because the result of equation (3) is a group key, the bit length of *p* includes the group key size. Therefore, the length of the pseudorandom number is 128 bits when the key size is 128 bits; therefore, the random number can be generated by encryption in that the bit length is 128 bits, for example, AES, etc. If 1,000 nodes are in the group, a 999-times encryption is required to share the group key in this method; therefore, the computational cost becomes high.

A large number of nodes causes this problem. The group is divided into clusters of three nodes each to decrease the number of nodes *n*. A cluster key is generated in each cluster by the Hamasaki et al. scheme, and a group key is sent to every cluster with the cluster key. We present the operation to share the group key. First, we present the BS steps. All nodes store each unique key and the BS knows all unique keys and IDs. The description of the scheme steps is provided below:

1. The BS divides the IDs of nodes in the group into clusters $c_j\ (j = 1, 2, 3, \ldots, m)$ of three nodes in ascending order.
2. The BS generates a random number *r*, and calculates each node coordinates $a_i(a_{ix}, a_{iy})$ based on the random number *r* and each unique key using pseudorandom number generator.
3. The BS calculates the center of the circle $O_j(O_{jx}, O_{jy})$ of each cluster with the Hamasaki et al. scheme.
4. The BS calculates the radius for cluster key s_j in each cluster by equation (4).

$$s_j = \left(o_{jx} - a_{ix}\right)^2 + \left(o_{jy} - a_{iy}\right)^2 \quad (4)$$

5. The BS generates group key *S* and encrypts group key *S* with cluster key s_j into g_j.
6. The BS broadcasts the following information. cID_j is the smallest ID in cluster c_j.

$$r, (cID_j, O_j(O_{jx}, O_{jy}), g_j)\ (j = 1,2,3, \cdots, m)$$

Next, we present the steps for a node *i*.

1. If its node ID_i is in the received $cID_j \sim cID_{j+1}$, its cluster is c_j.
2. The node i calculates pseudorandom number $a_i(a_{ix},a_{iy})$ based on random number r and its own unique key.
3. The node i calculates the distance from the received center of circle O_j to its own node coordinate a_i by equation (4).
4. The node i decrypts g_j by s_j to obtain group key S.

4 EVALUATION

The proposed scheme is compared with the conventional methods discussed in Section 2 based on the five factors presented in Section 1. The results of comparison are presented considering computational cost, communication cost, and storage requirement, the performance evaluation process is described in detail in Section 4.1. We present the results of the evaluation based on key renewal and security as outlined in Section 4.2.

4.1 Performance Evaluation

The performance indicators in terms of computational cost, communication cost, and storage requirement for each node are evaluated for all considered schemes. Table 1 lists the order for all used indicators and schemes. The BS performance is not evaluated because a BS must be tamper-resistant and include sufficient computational ability and power supply capacity.

To perform a simple evaluation, we set the length of constants such as counter value, random value, and ID of a node or a key, equal to *L1*; the length of data that are encoded by a common key cryptosystem, hash, or ECC operations to *L2*; and the length of data that are processed by a discrete logarithm to *L3*. Generally, $L1 < L2 << L3$.

For computational cost, we set the computational complexity of addition and multiplication equal to *C1*; that of using a common key cryptosystem, hash, pseudorandom number generation, and simultaneous equation processing including the secret-sharing scheme to *C2*; and that of ECC operation and the discrete logarithm to *C3*. Generally, $C1<C2<<C3$.

Meanwhile, LEAP, ECCKS, and SSKM have an initiator. However, Hamasaki et al.'s scheme and the proposed scheme do not have one. It is possible to assume an initiator for Hamasaki et al.'s scheme and for the proposed scheme to keep the settings of nodes in the group consistent (SSKM has a BS as well). Therefore, a group consists of one initiator and m peripheral nodes. In this case, each peripheral node sends its own ID to the initiator.

Table 1: Comparison of results.

	(a)	(b)	(c)	(d)	(e)
(1)	$mL2$	m^2L2	$L2$	$mL1$	$mL1$
(2)	$L1$	$L1$	$L3$	$L1$	$L1$
(3)	$2mC2$	$mC3$	$C3$	$mC2$	$2C2$
(4)	$3C2$	$C3$	$2C3$	$mC2$	$2C2$
(5)	$L2$	$L2$	$L2$	$L2$	$L2$

(a) LEAP
(b) ECCKS
(c) SSKM
(d) Hamasaki et al. scheme
(e) Proposed scheme
(1) Communication cost (initiator)
(2) Communication cost (peripheral node)
(3) Computational cost (initiator)
(4) Computational cost (peripheral node)
(5) Storage requirement

LEAP shares a group key (cluster key) using pairwise keys. Therefore, during the pairwise key-sharing phase, the initiator needs the communication cost of $mL1$ to send its own ID to each node, then the peripheral nodes send their own ID to the initiator, whereas the group key-sharing phase requires that of $m(L1+L2) \fallingdotseq mL2$ for the initiator as it has to transmit the node ID and enciphered key.

In LEAP, the initiator performs one encryption for its own ID, and one encryption for every node ID in the pairwise key sharing. In addition, the initiator performs m encryption for the peripheral nodes. Therefore, the initiator requires the computational cost of $(1+2m)C2 \fallingdotseq 2mC2$, and each peripheral node performs two encryptions and one decryption for pairwise and group key-sharing. Therefore, that of peripheral node is $3C2$.

ECCKSS uses an ECC-based operation. The initiator transmits (*Auth; C*; *R*; *U*; *P*) for each node. Therefore, it requires the communication cost of $m\{(m + 1)(L1 + L2) + L2\} \fallingdotseq m^2L2$. The communication cost required by the digital signature and Ack was not included in this evaluation as these objects are used for verification.

In ECCKSS, each node performs an ECC operation for S_j, a hash for u_j and v_j, and two additions to calculate the group key. Considering the initiator, processing the ECC operation for R and addition for $\overline{u_j}$ and k are added. Therefore, each node requires the computational cost of almost $C3 + 2C2$

$+ 2C1 \fallingdotseq C3$, and the initiator requires the computational cost of almost *mC3*.

SSKM applies a discrete logarithm. The initiator sends *xch* to the BS. A peripheral node sends (*IDi, yi*) to the BS. y_{CH} and *yi* are extremely large numbers. Therefore, the communication cost of the initiator is *L2* and that of each peripheral node is almost *L3*.

In SSKM, the nodes except the initiator calculate $yi = g^{xi} \bmod N$ and equation (1) to obtain a share. The nodes solve a simultaneous equation processing to obtain the group key. Therefore, each node requires the computational cost of almost $2C3+C2 \fallingdotseq 2C3$ and the initiator requires that of almost $C3+C2 \fallingdotseq C3$.

Hamasaki et al.'s scheme performs calculations to obtain the center of a circle by the BS. Therefore, each peripheral node sends its own ID to the initiator, and the initiator sends the IDs of the peripheral nodes to the BS. Each node receives the coordinate of the center of the circle from the BS. Therefore, the initiator requires the communication cost of *mL1,* and each peripheral node requires that of *L1*.

In Hamasaki et al.'s scheme, each node generates *m* pseudorandom numbers and calculates *S* using addition and multiplication. Therefore, each node requires $mC2+C1 \fallingdotseq mC2$.

The scheme proposed in this paper forms multiple clusters, which allows keeping *m as a relatively small number*. The communication cost for each node is the same as that of Hamasaki et al.'s method. However, the computational cost of the proposed scheme is the smallest comparing to other consider methods, because *m* is maintained small. Therefore, each node requires the computational cost of $2C2+C1 \fallingdotseq 2C2$.

5 APPLICATION EXAMPLE

The proposed scheme is characterized by easy group key update. Therefore, we can present applications where only authorized people can receive service in unpartitioned space. To build an analogy, let us consider a hypothetical room where people who have a communication device frequency enter and exit. However, it is not necessary for a room to be surrounded by a wall, which would cut off communication. Let us consider that a service, which is operated in the room is e-sports, established such as only authorized people sharing a common group key can use it. The game of e-sports is projected to a big screen in the room, and only the people in the room can share the information. When a person exits the room, the group key is immediately updated to remove that person from the group. Therefore, he loses access to the service at the moment when he leaves the room. Subsequently, when someone enters the room, the group key is updated to include this new person and he can obtain access to the service. Therefore, it is easy to respond to services that are frequently updated by participants using the proposed scheme.

6 CONCLUSION

We proposed an efficient key-sharing scheme, which implies dividing into clusters to decrease computational cost. In future work, we would like to consider pairwise key sharing, and propose a method using pairwise key sharing and group key sharing.

REFERENCES

Estrin, D. et al, 1999. Next century challenges: scalable coordination in sensor networks. In *Mobile Computing and Networking (Mobi- Com'99)*, Seattle, WA, pp.263-270

Hamasaki, J. et al, 2016 "Geometric Group Key-Sharing Scheme," Global Conference on Consumer Electronics Pages 667-668.

Zhu, S. et al, 2003. LEAP: Efficient Security Mechanisms for Large-ScaleDistributed Sensor Networks, In *ACM Conference on Computer and Communications Security*, pp.62-72.

Zhu, S. et al, LEAP+: Efficient security mechanisms for large-scale distributed sensor netwoeks, In *ACM Trans. On Sensor Networks (TOSN)*, Vol.2, Issue 4, pp. 500-528, 2006.

Porambage, P. et al, 2015. Multicast Communication in Wireless Sensor Networks Deployed for IoT Applications, In *IEE Access* pp. 1503–1511.

Zhang, Y. et al, 2013. A Secret Sharing-Based Key Management in Hierarchical Wireless Sensor Network, In *International Journal of Distributed Sensor Networks Volume*.