Evaluation of Ticket-based Access Control Method Applied to IoT Data Distribution

Masaki Yoshii\textsuperscript{1a)}, Ryohei Banno\textsuperscript{2}, and Osamu Mizuno\textsuperscript{1}

\textsuperscript{1}Graduate School of Electric and Electronics Engineering, Kogakuin University
1-24-2, Nishi-Shinjuku, Shinjuku-ku, Tokyo 163-8677, Japan

\textsuperscript{2}Faculty of Informatics Department of Information and Communication Engineering, Kogakuin University, 2665-1, Nakanomachi, Hachioji-shi, Tokyo 192-0015, Japan

\textsuperscript{a)} cm20053@ns.kogakuin.ac.jp

Abstract: Fog computing provides various new services with distributing Internet of Things (IoT) data. To distribute IoT data, we apply the publish/subscribe messaging model to a fog computing system. However, the user's private data may be distributed without his or her permission. In this paper, we propose using Ticket-based Access Control to distribute IoT data. It enables detailed control over a user’s data access. Because we need to understand the performance of the ticket authentication and authorization infrastructure, we simulate it. As a result, the average processing delay time is nearly the same when the number of authentication and authorization channels is 5 and 10. In addition, the average processing delay time is reduced by approximately 61\% when compared to the case with one channel.

Keywords: IoT Data Distribution, Access Control, Fog Computing

Classification: Network System

References

[1] Masaki Yoshii, Ryohei Banno, Osamu Mizuno, “Performance evaluation of table-based access control list applied to IoT data distribution method using fog computing,” Proceedings of the IEICE International Conference on Emerging Technologies for Communications (ICETC), E1–1(Dec. 2nd), 2020. DOI: 10.34385/proc.63.E1-1.

[2] Masaki Yoshii, Ryohei Banno, Osamu Mizuno, “Evaluation of table-based access control in IoT data distribution method using fog computing,” IEICE Communications Express 10(10) 822–827. DOI: 10.1587/comex.2021xb0134.

[3] MySQL, https://www.mysql.com/jp/ (accessed Sept. 16th, 2021).

[4] Redis, https://redis.io/ (accessed Sept. 16th, 2021).

[5] Aulia Arif Wardana and Riza Satria Perdana, “Access Control on Internet of Things based on publish/subscribe using authentication server and secure protocol,” Proceedings of the 10th International Conference on Information
1 Introduction

We proposed an IoT data distribution method using fog computing [1]. The problem with this method is that the private data of players who install IoT devices and use IoT services may be distributed to other players unnecessarily because service providers (SPs) assign IoT data acquisition IDs to a variety of IoT devices. To solve this issue, a Table-based Access Control List (ACL) applies to the user’s fog nodes (UFNs) [2].

However, Table-based ACL requires the user to set many access control parameters according to the number of services. In addition, because of the fixed table format, it is not easy to set access control according to user requirements, such as “use IoT data only for one hour” or “use IoT data only in case of disaster.”

This paper proposes Ticket-based Access Control (TAC) mediated by Ticket Authentication and Authorization Infrastructure (TAAI). The performance of TAAI is shown by simulation.

2 Ticket-based access control

Fig. 1. Overview of the operation of Ticket-based Access Control.

We assume three players: the users, SP, and TAAI operator. Users collect IoT data with the help of IoT devices and their UFNs. SP owns the cloud servers and operates the service through their fog nodes (SFNs). TAAI includes a mechanism for authenticating and authorizing tickets, as well as the maintenance of contractual relationships with users and SPs.

Fig. 1 shows an overview of the operation of TAC. As a prerequisite, it is assumed that the Tag ID, which is an ID for IoT data acquisition, is shared among SPs. SP₁ is a service provider that resolves the naming convention of Topic, which is the IoT data acquisition ID, for the user with whom it has a contractual relationship; it sends the Tag ID for naming convention resolution to the user via their SFN (noted as SFN₁).

SPₙ, which does not have a direct contractual relationship with the user, is used via SFNₙ to generate a ticket. The ticket includes the time of use of IoT data and its Tag ID. The generated ticket is sent to TAAI. TAAI authenticates the ticket and
sends it to $UFN_1$ if it is a legitimate ticket. $UFN_1$ sends IoT data directly to $SFN_n$ based on the content of the ticket.

TAAI uses three kinds of databases. One is the Tag ID management database to manage Tag IDs for each user. Second is the client management database to manage Clients who maintain contractual relationships with TAAI. The client in this case is both the SP and the user. The third component is the ticket management database, which is used to manage tickets. When SFN sends a ticket, TAAI refers to the client management database to check the identifier of SFN. Assume TAAI and SP have a contractual agreement. In that case, the ticket’s Tag ID is compared to TAAI’s Tag ID management database. If Tag ID is already registered in the database, the ticket is discarded; otherwise, Tag ID is registered in the ticket management database. When a UFN obtains a ticket, it first submits a ticket request to TAAI. TAAI checks the client management database, and if the ticket is registered in the ticket management database, the ticket will be sent to UFN.

3 Simulation evaluation
3.1 Parameters of the TAAI database

The proposed method is evaluated by simulation. To benchmark databases, MySQL and Redis register dummy data. The Tag ID management and client management databases use MySQL version 14.14 Distribution 5.7.35 [3]. The ticket management database uses Redis 6.2.5 [4], a key-value database system.\footnote{Note that the measured values are for reference only, as they vary depending on CPU performance and HDD read/write performance.} We set the number of simultaneous connections as 1, 2, 4, ..., 128. The load time of the database is assumed to be 60 s. The overall time taken to retrieve the data (delay time) is used as an evaluation parameter in the simulation.

We assume that the ticket is stored in a String type. The data size varies from 1000, 2000, 4000, ..., 32000 bytes. The number of threads varied from 1, 2, 4, ..., 128. Then, we evaluate the PUSH/SET delay time of the ticket.

3.2 Simulation parameter

It is necessary to quantitatively evaluate the performance of TAAI and analyze how it affects the IoT system. Therefore, in this simulation, we evaluate the average processing delay time when a ticket transfers from SFN to TAAI and registers in each database.

Table 1 shows the simulation parameters. As a simulation model, we use $M/M/k/c$. Here $c$ is the number of TAAI channels, which indicates the number of tickets that can be processed simultaneously. $k$ is TAAI’s queue length.

First, SFN sends the ticket to TAAI at an average interval of 20 ms. After the TAAI queue stores the ticket, TAAI processes the ticket in the available channel and sends it to UFN. UFN requests and receives the ticket from TAAI. Based on the content of the ticket, the UFN receives the ticket and sends IoT data to the corresponding SFN. Because the goal of this simulation is to determine the average processing delay between SFN and TAAI, the average processing delay between TAAI and UFN, as well as the average processing delay between UFN and SFN,
has no direct effect on the results. Eq. (1) shows the evaluation formula. \( T_d \) is the total processing delay time between SFN and TAAI.

\[
T_d = D_t + D_{tk} + T_t + T_s
\]  
(1)

\( D_t \) is the communication delay between TAAI and SFN. \( D_{tk} \) is the transmission interval of the Ticket sent from the SFN to the TAAI. \( T_t \) is the average response time of TAAI. \( T_s \) is the average response time of SFN. The following equation gives the response time for each of TAAI and SFN.

\[
T_t = W_t + P_t + P_{db}
\]  
(2)

TAAI average response time \( T_t \) is the sum of the ticket queuing delay \( W_t \), the average processing time \( P_t \), and the total response time \( P_{db} \) of each database. Since the SFN generates and transmits tickets, there shall be no queuing delay, and the average processing time \( P_t \) of the SFN shall be the same as the response time \( T_s \).

\[
P_t = T_s
\]  
(3)

The processing time of the database obtained by benchmark beforehand is calculated to follow an exponential distribution between the minimum and maximum values in the simulation.

### 3.3 Simulation results and discussion

Fig. 2 shows the simulation results. The maximum average processing delay time was approximately 520 ms when TAAI channels were one. The average processing delay time grows exponentially as the number of threads increases. Because there is only one channel, tickets are held in the queue until the channel’s processing is completed.

| Parameter                                  | Value          |
|--------------------------------------------|----------------|
| Simulation model                           | M/M/c/k        |
| TAAI queue length (k)                      | 1000           |
| Number of TAAI                             | 1              |
| Number of SFN                              | 2, 4, 8, ..., 128 |
| Number of UFN                              | 2, 4, 8, ..., 128 |
| Number of IoT                              | 96             |
| Average process time [ms]                  | 1.0            |
| Average interval for sending a ticket [ms]  | 20.0           |
| Transmission delay [µs]                    | 50.0           |
| Delay time of Tag ID management database (min) [ms] | 56–325         |
| Delay time of Tag ID management database (max) [ms] | 333–3881       |
| Delay time of client management database (min) [ms] | 50–142         |
| Delay time of client management database (max) [ms] | 183–951        |
| Delay of ticket management database [ms]   | 0.0083         |
| Ticket size [byte]                         | 4000           |
| Number of TAAI channels (c)                | 1, 5, 10       |
| Simulation time [s]                        | 3600.0         |
| Simulation trials                          | 30             |
When the number of channels is five, the maximum average processing delay time is approximately 200 ms. When compared to a single channel, the average processing delay time is reduced by approximately 61%. The number of tickets that can be processed concurrently increases as the number of channels increases. As a result, the time spent waiting for tickets is reduced.

Even with 10 or 5 channels, the average processing delay is almost the same. The number of channels is 5, indicating that the TAAI is fully capable of processing tickets simultaneously.

Scalability is desired for most of the IoT services; although the requirements vary depending on the usage environment of the user of the IoT service, we consider the proposed method with respect to processing delay time as an indicator. By increasing the number of TAAI channels, the proposed method can reduce the processing delay time. The proposed method can reduce processing delay time by increasing the number of TAAI channels, and because it has a contractual relationship with each player, it can adjust the number of channels based on each player’s ticket transmission/request. It can be said that it is effective for users of IoT services to reduce processing delay time by changing the number of TAAI channels. In the future, we will check the scalability of the user side.

4 Related works

Wardana and Perdana [5] proposed the application of authentication servers and tokens for authentication in IoT systems. As part of their proposal, they set up an authentication server within the MQTT broker that issues tokens to manage and verify the secure payload. In addition, SSL certificates are used to secure communication in the MQTT protocol. The IoT data and token are sent simultaneously from the Publisher to the gateway and authenticated. The traffic also increases as the token becomes dominant. In our method, TAAI mediates the ticket and can obtain it when the UFN requests the ticket. Thus, the traffic can be reduced.

5 Conclusion

In this paper, we proposed a TAC applied to the IoT data distribution method. Simulations show that increasing the number of TAAI channels can reduce the average processing delay time by approximately 61%.
In the future, we will quantitatively show the average processing delay time between the user and TAAI and the reduction of traffic by access control.

Acknowledgments

Part of this research was supported by JSPS KAKENHI Grant Number JP18K11276.