Regular Paper

Automatic Roaming Consortium Discovery and Routing for Inter-federation Wireless LAN Roaming System

Kazunari Irie[1] Hideaki Goto[2,a)]

Received: July 6, 2019, Accepted: April 8, 2020

Abstract: Next Generation Hotspot (NGH) is being introduced as a new standard that brings higher security and usability to Public Wireless LANs. There has been progress in recent years toward connecting some Roaming Consortia (RCs) by NGH to develop large-scale Wireless LAN roaming systems. However, a communication routing problem has been identified in the identity federation network when an RC consisting of multiple operators is connected. A typical example is eduroam, the roaming system for research and education institutions, which has thousands of different realms. It is hard for wireless Service Providers (SPs) to find which RC the authentication requests should be sent to without having a large realm-RC mapping list. In addition, introducing a new system for realm list exchanges among the operators is not easy as some RCs have difficulties in modifying their existing systems due to operational constraints. To deal with these problems, we developed a Hub Proxy with automatic RC discovery and routing features. In this study, the effectiveness of the system was confirmed through experiments using a virtual inter-federation roaming network.

Keywords: next generation hotspot (NGH), eduroam, wireless LAN roaming, RADIUS, roaming consortium

1. Introduction

Public Wireless LAN (WLAN) is now quite popular at various places such as cafes, airports, train stations, and tourist sites. However, it has been noted that most Access Points (APs) currently providing so-called "Free Wi-Fi" have a wide variety of security problems including eavesdropping and evil twin AP attacks. A standard known as Next Generation Hotspot (NGH) was developed in order to improve the security and usability of Public WLANs and is being introduced worldwide [1]. The NGH standard is based on the specification known as Hotspot 2.0 or Passpoint [2], which is based on the IEEE 802.1X standard [3].

NGH-enabled Public WLAN systems are expected to play important roles not only in the tourism industry but also to support the rapid growth of telecommunications traffic. Many telecom companies are interested in deploying NGH in cities and so are some governments and enterprises in the tourism industry. A good example can be seen in the City Wi-Fi Roaming trial conducted by the Wireless Broadband Alliance (WBA) [4]. Although international roaming used to be available in some conventional commercial WLAN services, development of a roaming system for secured Public WLANs has just begun. Besides progress within industries, one of the most successful WLAN roaming systems is "eduroam" [5], which was developed for research and education institutions. Another example is "govroam" for public-sector and government organizations [6], which was developed based on the eduroam architecture [7]. Eduroam, as well as govroam, is a WLAN Roaming Federation. A Roaming Federation is a group of operators that is also referred to as a Roaming Consortium (RC).

There has been great interest in enabling eduroam/govroam services on Public WLANs at various places. Since eduroam architecture is based on the IEEE 802.1X standard, we can expect that deploying the NGH systems will greatly facilitate the adoption of eduroam/govroam and will also contribute to better public services in the digital era. We have been developing an NGH-based identity federation system interconnecting not only various Public WLAN systems worldwide but also eduroam/govroam [8]. The current Passpoint/NGH systems also have the benefit of being able to provide backward compatibility for 802.1X-only devices. If an AP is connected to the identity federation network, we can enable eduroam/govroam just by adding their SSIDs. A communication routing problem was identified as follows.

On the identity federation network, a network Service Provider (SP) has to deliver the authentication request from the user's device to the authentication server at the user's home operator or institution. RADIUS (Remote Authentication Dial-In User Service) [9] is the most popular protocol for authentication requests and reply exchanges. In general, the realm name attached to the user ID is used to find the route to the authentication server. If the number of roaming operators is small, each SP can easily identify the user's home operator, also referred to as Identity Provider (IdP), by using a small list consisting of realms, roaming partner operator names, and their RADIUS server addresses.

Suppose an SP provides roaming services for some large RCs as shown in **Fig. 1**, and each RC has many realms. For example,

[1] Graduate School of Information Sciences, Tohoku University, Sendai, Miyagi 980–8579, Japan
[2] Cyberscience Center, Tohoku University, Sendai, Miyagi 980–8578, Japan
a) hgot@cc.tohoku.ac.jp

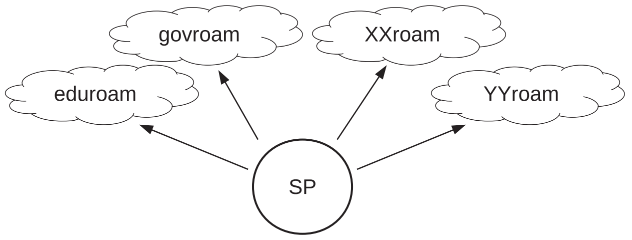


Fig. 1 Routing problem in a roaming system connecting large RCs.

eduroam is an RC spanning thousands of universities worldwide. The SP needs to send the authentication request to the RC to which the user belongs. However, the SP cannot always find the correct destination by simply looking at the realm because the RC name is not usually embedded in the realm.

Some methods have been developed to deal with this routing problem. Yamaki et al. proposed a method using an Organization Identifier (OI) [10], as defined in the Hotspot 2.0 specification [2]. The OI is also known as RCOI since it may show the RC. The user's device handles prepending of the OI to the username enabling the AP to find the correct RC. This method requires specially designed APs which are not currently available in the market. In addition, changing all existing usernames is required at every single IdP, and many eduroam member universities would find this difficult. A similar idea can be seen in the "Roaming Consortium Selection element" newly introduced in Hotspot 2.0 Version 3.0 [11]. Neither of these methods can provide backward compatibility for 802.1X-only devices, while Hotspot 2.0 systems can normally provide backward compatibility.

We have developed a partial solution to the routing problem in our previous work [8] along with an inter-federation roaming architecture based on the eduroam architecture. A regional Hub Proxy placed in each country has a realm list consisting of the realm-RC pairs in the region and then handles authentication request routing. To our knowledge this architecture is the only one available and tested internationally with eduroam [4] as of the time of this writing. However, compilation of the realm list remains to be addressed and it currently must be done manually by a human operator. System automation for labor reduction is one of our requirements for developing a practical system incorporating eduroam.

In this paper, we focus on realm-based routing at the Hub Proxy. The inter-federation roaming system should accommodate both the 802.1X-only and Hotspot 2.0 environments because we aim at prompt adoption of eduroam on the NGH. We have developed an automatic realm-RC mapping mechanism that works on the Hub Proxy and that does not require any modification on the RC side. This condition is crucial since we are aiming to include as many eduroam member institutions as possible. It would be technically possible to develop a realm list synchronization system among the RCs instead of this automatic mapping mechanism. However, our long experience in the operation of the eduroam infrastructure shows that asking other operators to introduce an additional function is not always easy for various reasons that include the difficulty of system modification, limited budgets, and operational constraints.

This paper is organized as follows. Section 2 presents the system architecture and proposes a new method for RC discovery and routing. The security of the proposed system is also analyzed. In Section 3, our system implementation is described and the effectiveness of the system is confirmed through some experiments. Although we presented the basic idea of RC discovery in our conference paper [12], the details were necessarily omitted as it was a short work-in-progress report and, more importantly, our early data structure and algorithms worked under only some limited conditions and we did not fully prove the effectiveness of our approach. The evaluations were done only on a simulator and no actual device was used. A new system developed from scratch and its evaluation results using actual user devices are presented in this paper. Conclusions are given in Section 4.

2. Automatic Roaming Consortium Discovery and Routing

2.1 RADIUS-based User Authentication and Roaming

IEEE 802.1X is a user authentication standard for network access control which is often implemented using the RADIUS protocol [3], [9]. The 802.1X standard supports the Extensible Authentication Protocol (EAP), which provides various authentication methods [13]. Some representative examples are EAP-TLS, EAP-TTLS, and PEAP. EAP messages are exchanged between the user device and the authentication server via the AP and some RADIUS proxies. EAP-TLS is a certificate-based mutual authentication protocol in which the user device and the server authenticate each other by using digital certificates issued by a Certificate Authority (CA). EAP-TTLS is an asymmetric mutual authentication protocol in which the user device first authenticates the authentication server using a server certificate, and after the authentication succeeds, the server authenticates the user by using a user ID/password pair. PEAP is similar to EAP-TTLS. Both PEAP and EAP-TTLS utilize an end-to-end Transport Layer Security (TLS) tunnel established between the user device and the authentication server to protect sensitive information such as real user IDs and passwords from a wide range of attacks including eavesdropping, falsification, and replay. No device between the user device and the authentication server can see or modify the information in the tunnel. The RADIUS proxies and the APs can see only the outer identity of EAP method and some RADIUS attributes. In real roaming applications, such as eduroam, server authentication has been strongly recommended for security reasons.

These EAP methods consist of two-part conversations: server authentication part followed by user authentication part. The user device sends Access-Request packets and the server sends Access-Challenge packets until the authentication is complete. The sequence of the RADIUS packets for each authentication can be tracked by using attributes such as "State" and "Proxy-State" at the RADIUS proxies. The beginning of an authentication sequence can be detected by checking the absence of the State attribute in the Access-Request packet.

When server authentication fails, the user device will not proceed to the user authentication part because this might disclose the user's credential to an incorrect authentication server. In the RADIUS protocol, the authentication server returns an Access-

Reject packet to the user device immediately if any value of the received response is not acceptable. However, the user device does not need to send an Access-Reject packet back to the authentication server when the presented server certificate is not acceptable. Indeed, some devices including iPhone silently terminate the conversations without sending an Access-Reject packet to the authentication server. Therefore, in such a case, the authentication process might be suddenly restarted by a new authentication sequence instead of receiving a response.

A network roaming system can be realized by creating an identity federation network connecting the RADIUS servers and proxies of some operators. Realm names have been quite widely used for identifying the IdPs in roaming environments. The username consists of two parts, "user ID" and "realm," concatenated using the @ symbol in many cases. The realm part is used to represent the name of the operator to which the user belongs. For eduroam and govroam, the realm is aligned with the Domain Name System (DNS) for routing convenience. For example, a student named Alice at Tohoku University may have "alice@tohoku.ac.jp" as a username. We can easily see in this example that the user belongs to Tohoku University in Japan, and the authentication requests should therefore be forwarded to the university. Note that the RADIUS proxies can see only the outer identity that often carries an anonymized username like "anonymous@tohoku.ac.jp," while the inner identity of EAP carries the actual username.

A group of WLAN operators on an identity federation network may form an RC. Note that each RC has its own identity federation network isolated from the other roaming federations. The realm does not always provide a hint about the users' home RC. We therefore need a new method to realize an inter-federation roaming system.

2.2 Inter-federation Roaming System

Suppose there are a small number of large RCs and an inter-federation roaming system is developed by connecting those RCs somehow. The eduroam architecture [7] uses the identity federation network logically isolated from other RCs and allows all member operators to trust each other without authenticating the peer IdPs or SPs. If such a federation network were simply connected to another one, and if the authentication requests became routable between the networks, anyone could get on any AP even without a roaming agreement between the different RCs. For example, the APs on the eduroam network would accept users from foreign RCs, but use of the APs should normally be restricted to the research and education community. An access control mechanism is required in order to avoid such an over-federation problem. As explained in Section 1, it is not always easy for the SP to determine the destination RC by simply looking at the realm. As long as the destination RC remains unknown, implementing access control is also difficult.

In our previous work [8], we proposed an inter-federation roaming architecture in which every country or territory has their own Hub Proxy server accommodating all local operators using the corresponding ccTLD (country code Top-Level Domain) in the realms. **Figure 2** shows the proposed architecture. The Hub Proxy handles the realm-based routing. The Hub Proxy collects all the realms from the local operators of the RCs to compile a regional realm list in advance. Thus, the workload of compiling the realm list, as well as the roaming system operation, can be dispersed around the world, and each regional realm list would be of a manageable size. This distributed architecture is similar to the eduroam's that contributed to the high scalability of the system. Since eduroam is much larger than commercial WLAN roaming systems with respect to the number of operators, any inter-federation roaming system accommodating eduroam naturally develops into a large scale.

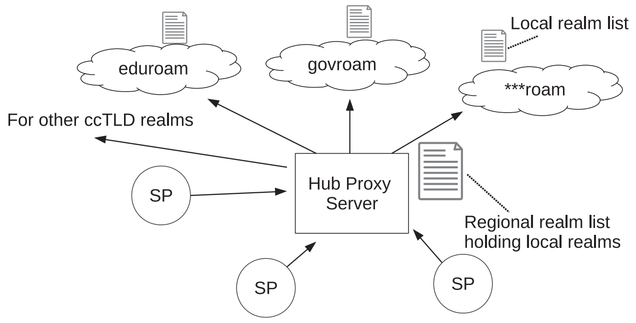


Fig. 2 Inter-federation roaming system using regional Hub Proxies [8].

The authentication requests from the local SPs are initially sent to the Hub Proxy and then forwarded to the correct RC. If the realm has a foreign ccTLD, the request is transferred to the Hub Proxy in the corresponding region. The Hub Proxy also handles the access control according to roaming agreements.

For example, in eduroam, there are roughly 970 IdPs in Germany, 640 in the US, 360 in the UK, and 270 in Japan, as of writing. There has not been any RC as big as eduroam, and the regional realm list would not be so large. However, in reality, handling 1,000 institutions manually is not an easy task for the human operator of the regional hub.

Although the number of institutions is limited to some extent in a region, each member institution may use some sub-realms, making the table larger. We may be able to make the regional realm list smaller by using *a priori* knowledge in regions where special realms are used.

In the case of eduroam, for example, the following base realms are used in Japan.

- <inst_name>**.ac.jp**
- <inst_name>**.eduroam.jp**
- <inst_name>**.f.eduroam.jp**
- <inst_name>**.v.eduroam.jp**
- <inst_name>.go.jp
- <inst_name>.jp

All the realms including .ac.jp and the .eduroam.jp sub-realms, shown in bold, obviously belong to eduroam. Authentication requests with these realms can be easily captured using regular expressions (regex). Since most of the eduroam members in Japan are using these realms, the regional realm list can be made much smaller. If the Hub Proxy sees one of them, it ignores the institution name part and forwards the Access-Request packet to a proxy in the eduroam RC immediately.

Making a group of eduroam institutions by using regex may not be practical in some countries where the use of general-purpose

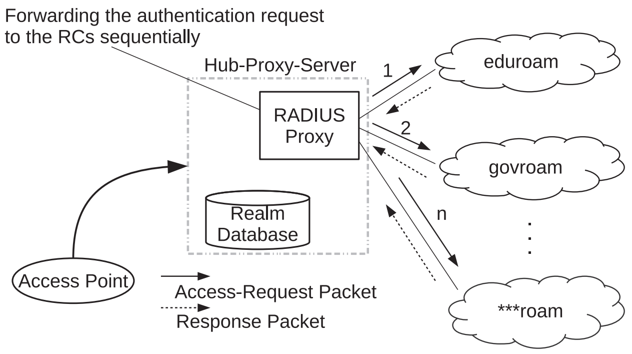


Fig. 3 Overview of the proposed system.

ccTLD domains is popular.

Although Japan has not joined govroam as of this writing, we assume that the govroam consortium exists. The .go.jp realms may be used by eduroam and govroam. Some institutions in eduroam use .jp realms aligned with general-purpose JP domains. The .jp realms may be used by eduroam, govroam, and other RCs. Therefore, each of these realms including the institution name needs to be registered separately on the regional realm list.

2.3 Basic Idea of Automatic RC Discovery

As described in Section 1, the realm collection and list compilation tasks must still be done manually by a human operator, and the labor aspect cannot be ignored as in our earlier work [8]. We think that these problems can be solved by introducing a mechanism in which the Hub Proxy automatically finds the forwarding address of the received request. This paper proposes a new system for automatic RC discovery and routing. Although we have already described the basic idea of the system in our conference paper [12], some important details have not yet been described. This section reviews the idea and provides its details.

Figure 3 shows an overview of the system. Suppose there are *n* RCs and their representative proxies are registered in a buffer of maximum size *n*. When the Hub Proxy has received an Access-Request packet from a user device, it forwards the request to one of the RCs. If the authentication is successful, the proxy stores the realm–RC pair and will reuse it in future processes. If the authentication fails, then the proxy switches the forwarding address to another RC. After a certain number of authentication attempts the Hub Proxy will eventually find the correct forwarding address as long as one of the users has a valid credential. The proxy will be able to register the address so that the proxy can forward the authentication requests with the same realm immediately afterward.

Using EAP and enabling server authentication are important parts of ensuring sufficient security in all WLAN systems. As described in Section 2.1, a TLS tunnel is used in some EAP methods. Therefore, no proxy can duplicate the authentication request and send the copied requests to multiple RCs to find the correct RC. RC discovery is challenging because the proxy cannot send RADIUS messages as broadcasts or probe requests even though they are popular in various network protocols. The Hub Proxy is unable to see which realm belongs to which RC until a valid authentication request goes through it. One of our key ideas is

Table 1 Proxy table containing fixed/learned realms.

realm name (exact or regex)	forwarding address
*.ac.jp	eduroam proxy
*.eduroam.jp	eduroam proxy
*.example1.jp	XXroam proxy
*.example2.jp	govroam proxy
...	...

Table 2 Realm tracker table for RC discovery.

inst. base realm	current RC ID
example3.go.jp	1 (eduroam)
example4.jp	4 (XYroam)
...	...

Table 3 Authentication tracker table.

inst. base realm	forwarding address	tracking key (State attribute, etc.)
example3.go.jp	eduroam proxy	...
example4.jp	eduroam proxy	...
example4.jp	XXroam proxy	...
...	...	...

to use the authentication retry function which all user devices are equipped with to find the RC. User devices normally repeat authentication trials when an authentication fails.

We consider the RC sending back an Access-Accept packet to be the correct one. The user authentication succeeds only when a valid user credential is delivered to the user's home authentication server in the correct RC. Once the Hub Proxy has received the Access-Accept packet for the given realm, it stores the realm–RC pair. In this paper, to "learn" means that the Hub Proxy stores the realm–RC mappings found by the RC discovery. The security features of the EAP methods ensure that the authentication requests sent to incorrect RCs always fail, with or without an explicit Access-Reject reply.

2.4 System Design for RC Discovery and Routing

This section presents the system design for RC discovery and routing. Since our early data structure and algorithms [12] failed to fully demonstrate the effectiveness of our proposed method, we have designed a new system from scratch.

The Hub Proxy has a proxy table used for realm-based routing just like the normal RADIUS proxy. The proxy table, shown in **Table 1**, may initially contain some pre-defined entries for known base realms such as .ac.jp and .eduroam.jp. New entries are added during automatic RC discovery. Once a realm has been registered in the table, proxy operation for the same realm takes place immediately and skips the RC discovery.

The Hub Proxy uses two more tables, a realm tracker table (**Table 2**) and an authentication tracker table (**Table 3**), for controlling automatic RC discovery. These tables are initially empty. One entry is inserted into the realm tracker table when the Hub Proxy has received an authentication request with an unknown realm. Sub-realms are stripped off if present, and only the base realm including the institution name is recorded. Each entry corresponds to an individual ring buffer containing the RCs in the same base realm class. One entry may be shared by multiple user devices that run the authentication sequences with the same institution base realm. All RCs are numbered, and each is represented by an integer RC ID (IDentifier). The initial RC ID is chosen randomly, although the selection algorithm is not so important since it is implemented to give each RC the same opportunity in terms of the authentication retry count. We apply the Least Recently Used (LRU) algorithm to remove the old entries with failed realms that accumulate and might fill up the table.

Each entry on the authentication tracker table is created for each authentication sequence or device, in other words, when the Hub Proxy has received a new authentication request with an unknown realm. Note that this table may have duplicate base realms, e.g., example4.jp, when there are multiple devices trying authentication with the same realm. The tracking key field is for tracking the authentication sequence. This table is used to capture the current forwarding address and to retain it until the Hub Proxy receives an Access-Accept packet. The proxy feature of the Hub Proxy uses the entries in both the proxy table in Table 1 and the authentication tracker table in Table 3.

Suppose two user devices are running authentication sequences for the same realm concurrently. If the authentication tracker table did not exist and if the realm tracker table were used for forwarding, the problem might occur that one of the devices changes the forwarding address just after the other receives an Access-Accept packet and before it registers the original address to the proxy table. Our early system [12] had this problem. To avoid an incorrect realm–RC pair being registered, we need mutual exclusion and serialization mechanisms which make concurrent processing difficult. These problems can be automatically eliminated by utilizing the authentication tracker table as long as the RADIUS server software can track the authentication sequences correctly.

When the Hub Proxy has received an Access-Accept packet, the corresponding realm–RC pair found in the authentication tracker table is added to the proxy table.

2.5 Details of the Algorithms

2.5.1 RADIUS Packets

The Hub Proxy receives four types of RADIUS packets, Access-Request, Access-Accept, Access-Reject, and Access-Challenge during the authentication sequence. The proposed system uses them as the triggers for various processes.

It might seem natural to switch the destination RC during RC discovery when the Hub Proxy detects an authentication failure. However, using the Access-Reject packet does not always work because some devices do not explicitly send Access-Reject packets as described in Section 2.1. If Access-Reject were used, those devices would continue trying the current RC when server authentication fails and lose a chance to try other RCs. This problem was found in our first prototype system [12]. To avoid this problem, our new system proceeds to the next RC when the first Access-Request packet is received. Therefore, no additional processing exists for Access-Reject packets, and we consider three situations only. The following sub-sections describe the algorithms.

2.5.2 Access-Request

Figure 4 shows a flowchart for the algorithm initiated by an Access-Request packet. The blocks in the dashed boxes represent Denial-of-Service (DOS) suppression which will be explained in

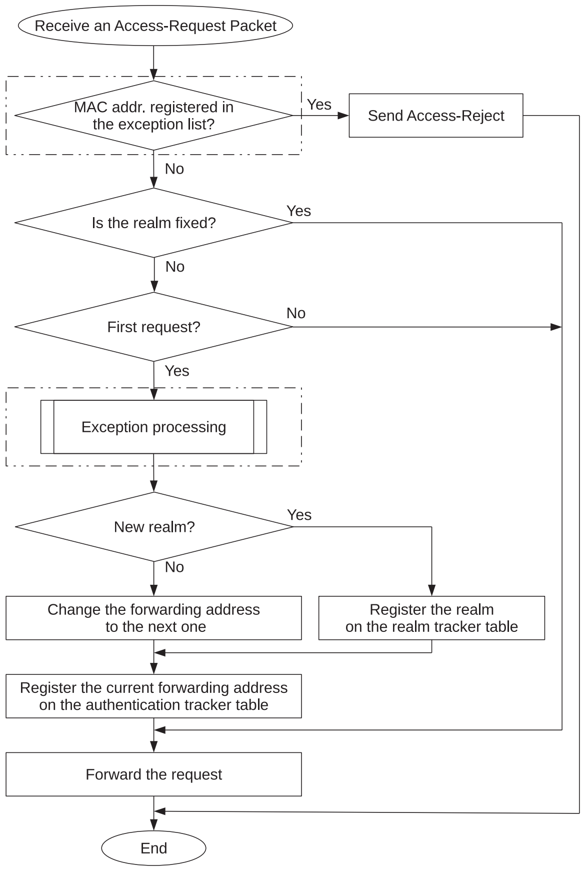

Receive an Access-Request Packet
MAC addr. registered in the exception list?
Yes
Send Access-Reject
No
Is the realm fixed?
Yes
No
First request?
No
Yes
Exception processing
New realm?
Yes
No
Change the forwarding address to the next one
Register the realm on the realm tracker table
Register the current forwarding address on the authentication tracker table
Forward the request
End

Fig. 4 The flowchart for Access-Request packets.

detail later. When the Hub Proxy receives an Access-Request packet, it checks the device's Media Access Control (MAC) address found in the Calling-Station-Id attribute in the RADIUS protocol. If the MAC address is found in the exception list which retains the addresses of potentially malicious devices, the Hub Proxy responds immediately with an Access-Reject packet without relaying the Access-Request packet for the current RC.

If the request is not the first one, then the proxy takes no particular action and forwards it to the RC with the matching realm by using the proxy table (Table 1) and the authentication tracker table (Table 3). If the request is the first one of a new authentication sequence, then the Hub Proxy tries to find whether the proxy table has an entry matching with the realm in the request. If there is a match for the realm, then the request is forwarded immediately.

If no matching entry is found, then the Hub Proxy looks up the realm tracker table (Table 2). Next, sub-realms are stripped off in order to obtain the institution's base realm. Some *a priori* knowledge about the regional base names is used here to find the level of the institution name which is on the second or third level. In our examples we only consider the .go.jp and .jp realms. If the realm is new, a new entry is inserted into the realm tracker table. The same realm is registered in the authentication tracker table.

If the Hub Proxy successfully finds a matching realm in the realm tracker table, the proxy increments the current RC ID in a round-robin manner to change the forwarding address. Then, the authentication tracker table is updated. A new entry is created if the authentication request originated from a new device.

Users may sometimes type in wrong user IDs and/or wrong

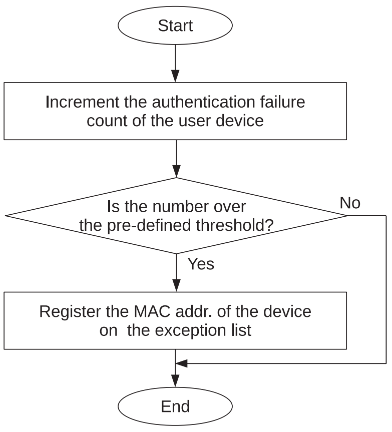

Start
Increment the authentication failure count of the user device
Is the number over the pre-defined threshold?
No
Yes
Register the MAC addr. of the device on the exception list
End

Fig. 5 DoS suppression processing.

realms during WLAN configuration on their devices. The Hub Proxy may see many authentication failures, and the authentication tracker table would thus become large. The First-In First-Out (FIFO) algorithm is used to clean up expired entries and to limit the table size. If a device repeats failed authentication frequently, probably due to a bad implementation, it can be captured by the DoS suppression processing explained later.

2.5.3 Access-Challenge

Access-Challenge packets are sent from the authentication server back to the user's device. When the Hub Proxy receives an Access-Challenge packet, the proxy updates the tracking key in the authentication tracker table and forwards the packet.

2.5.4 Access-Accept

When the Hub Proxy receives an Access-Accept packet, it checks whether the realm in the packet has already been registered. If the realm is found in the proxy table, the server does not have to carry out any processing and simply terminates the authentication sequence. Otherwise, the proxy picks up the forwarding address in the matching entry on the authentication tracker table. The address is registered in the proxy table together with the corresponding realm name. Note that we do not use the realm tracker table because the current RC may be modified by another user device just before the Access-Accept packet arrives.

2.5.5 DoS Suppression

A possible attack on the proposed system would be a DoS attack. It has been reported that some devices failed in authentication repeatedly, causing some congestion at the RADIUS proxies. Another scenario would be that a malicious person tries filling up the realm tracker table and the authentication tracker table by crafting and sending many fake authentication requests with non-existent realms. The legitimate users would suffer from temporary authentication failures if such an attack were to occur.

To mitigate the impact of this kind of attack, we have introduced a mechanism for DoS suppression as a precaution. Since the time parameters and the count threshold in this process will need some fine tuning in a real system in the future, this paper only demonstrates that such functionality is possible. Note that this feature is for protecting the system from simple and casual attacks rather than from any elaborate attacks.

Figure 5 shows a flowchart of the algorithm. The Hub Proxy performs the processing only if it receives first Access-Request packets with unknown realms. When the proxy sees a new device identified by its MAC address, it registers the device in the authentication counter table (**Table 4**) and sets the counter to 1. The proxy increments the counter every time it receives a similar packet from the same device. If the counter value exceeds a predefined threshold value within a short period of time, the device MAC address is registered in the device exception list shown in **Table 5**. After the device is registered in the device exception table, the Hub Proxy always replies to the device with an Access-Reject packet, skipping RC discovery.

Table 4 Authentication counter table.

Name of column	Content
timestamp	time stamp of the last rejection
mac_addr	MAC address of the device
auth_count	number of authentication failures

Table 5 Device exception list.

Name of column	Content
timestamp	time stamp of the registration
mac_addr	MAC address of the device

The device is removed from the authentication counter table after a pre-defined time has passed or the table becomes full. In the same way, the device is removed from the device exception list after a pre-defined time has passed or the list becomes full. The LRU algorithm is used for both table and list.

A limitation of this DoS suppression is that it cannot deal with attacks with MAC address changes. The MAC address may be easily changed on recent devices. This type of attack fools the RADIUS proxies and may apply a harmful load to the identity federation network as well. Therefore, such an attack should be handled on the AP system or on a device which is as close to the user device as possible.

2.6 Security Considerations

In the proposed method, the Hub Proxy may inevitably transfer an Access-Request packet with an unknown realm to some incorrect RCs owing to the nature of the algorithm. Since the Access-Request packet may carry sensitive information, such as passwords, disclosure of such information to an incorrect server would be a problem. The IEEE 802.1X standard provides some mechanisms for avoiding such information leakage as follows.

As already explained in Section 2.1, an end-to-end TLS tunnel between the user device and the authentication server is used in the EAP to protect sensitive information from eavesdropping. The user device can check the authenticity of the authentication server in a secure manner before transmitting the sensitive information if the CA certificate for the correct server is installed in advance. Therefore, in the proposed system, it is necessary to install the correct CA certificate in advance on the user device. Note that this requirement is not specific to our proposed method since enabling server authentication has become popular in many WLAN systems including eduroam for security reasons.

Even if the server authentication was not enabled for some reason, other mechanisms would still protect the user credentials. EAP-TLS is a recommended authentication method because it is based on digital certificates instead of a password. If conventional user/password authentication is preferred for operational reasons, and if MS-CHAPv2 is adopted, the security is limited to some extent since a vulnerability has been reported [14].

From the viewpoints of RCs, receiving authentication requests from users in foreign RCs would be a little bit noisier. However, this would not be a significant problem since we already see many stray authentication attempts, including those involving foreign realms, in the eduroam infrastructure as well as in some actual roaming systems today.

Operations that the proposed method can perform are limited to monitoring the server/user authentication status at the Hub Proxy, switching the destination RC, and retaining the routes to the correct RCs. Since no proxy is capable of intervening the traffic inside the TLS tunnel, we cannot find any security threat added by our system. If any were to be found, it would result in a security breach of the EAP method being used.

3. Implementation and Evaluation

3.1 Environment and Setting

We have developed a proof-of-concept inter-federation roaming system to evaluate the proposed method. FreeRADIUS is used as the RADIUS software since it is quite popular in eduroam and govroam. To implement the required functions, we use Python and Perl modules both of which work in FreeRADIUS.

The system consists of some virtual servers constructed by Docker on the Ubuntu operating system, and an access point is connected to one of the Docker containers acting as the SP. Details of these environments are shown in **Tables 6** and **7**.

Figure 6 shows the network layout of the virtual servers. Three RCs are assumed to exist and their top-level servers are connected to the Hub Proxy. We consider the number of RCs to be sufficient for evaluating the RC discovery method. Even if the number is more than three, we can expect that only the number of retries increases and that the function itself is not affected. At the beginning of this work, we confirmed that the federation network worked well by sending some Access-Request packets from the Hub Proxy to each RC by manual operation. Five tables explained in Sections 2.4 and 2.5 are created in the PostgreSQL database in addition to the normal schema for the RADIUS user database.

Table 6 Environment of the host server.

OS	Ubuntu 16.04
RADIUS Server	FreeRADIUS 3.0.15
Virtualization software	Docker 18.06.1-ce
Certificate management software	OpenSSL 1.0.2g

Table 7 Environment of the virtual servers.

OS on all servers	Ubuntu 16.04
RADIUS Server	FreeRADIUS 3.0.15
module 1	Python 2.7.12
module 2	Perl v5.22.1
SQL database	PostgreSQL 9.5.12

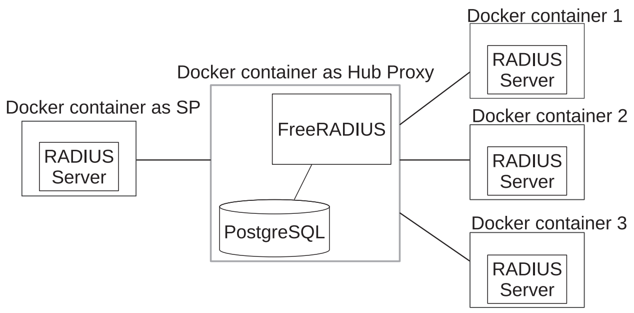


Fig. 6 Overview of the virtual network.

We cannot use the proxy.conf file in FreeRADIUS to dynamically change the destinations of the authentication requests since the file is read only once at start up. The destinations can be changed dynamically using a PostgreSQL database attached to FreeRADIUS. We register the addresses of the proxies of all RCs in the proxy.conf file in advance since they are static. Each RC is identified by a numbered name such as "proxy1," and the Hub Proxy switches the forwarding address by specifying the name instead of the actual IP address.

As shown in **Table 8**, we assigned some realms to each virtual server acting as an RC. Some regex entries are registered for only the fixed realms. For example, "`~^(.+\.)?ac\.jp$`" is used for grouping .ac.jp realms.

We used three types of operating systems as shown in **Table 9**. All the devices were configured with the server authentication feature enabled in advance.

3.2 Evaluation of RC Discovery and Learning

We must ensure that there is no problem in the system even in the worst case scenarios. In order to produce such cases intentionally, we have changed the RC ID initialization process in the following evaluations. The current RC ID in the realm tracker table is initialized so that the correct RC is the farthest.

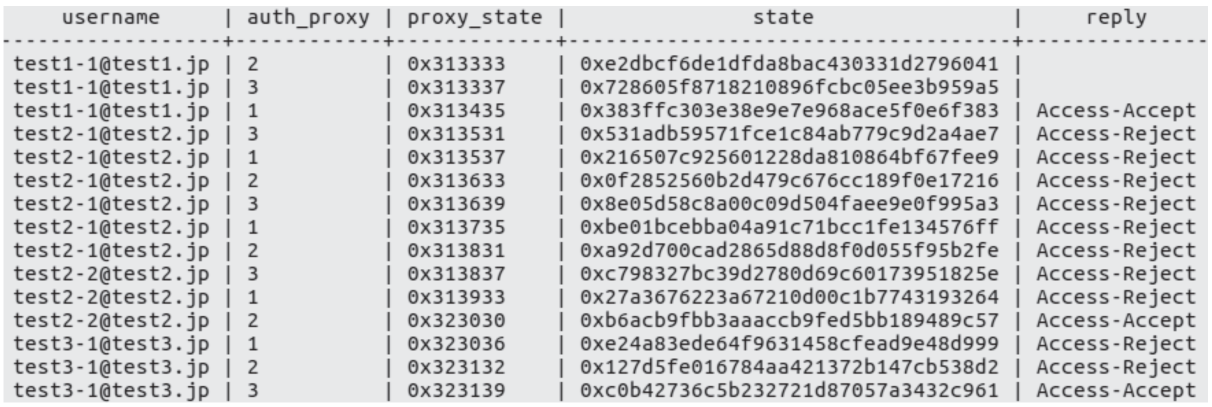
The devices send Access-Request packets to the Hub Proxy. **Figure 7** shows a portion of the authentication log in the SQL log file. The first three lines are for an iPhone 8 (iOS device) with the username test1-1@test1.jp. The Hub Proxy forwarded the first authentication request to proxy2, and then the second one to proxy3. The Hub Proxy did not receive any Access-Reject packet because the server authentications failed but the device did not send an Access-Reject packet. When the third request was sent to proxy1, which contains the realm test1.jp, both the server authentication and the user authentication were successful and the Hub Proxy received an Access-Accept packet. When that happened, the realm test1.jp was registered in the proxy table. No further

Table 8 Registered realms.

Roaming Consortium	institution base realms
Container 1	test1.ac.jp test1.eduroam.jp test1.v.eduroam.jp test1.f.eduroam.jp test1.jp
Container 2	test2.go.jp test2.jp
Container 3	test3.jp

Table 9 Specification of user devices.

Name of the device	OS version	Registered user ID
iPhone 8	iOS 12.1.4	test1-1@test1.jp
Nexus 7	Android 6.0.1	test2-1@test2.jp test2-2@test2.jp
Moto G6 (XT1925-7)	Android 8.0.0	test3-1@test3.jp



```
     username      | auth_proxy | proxy_state |               state                |     reply
-------------------+------------+-------------+------------------------------------+---------------
 test1-1@test1.jp | 2          | 0x313333    | 0xe2dbcf6de1dfda8bac430331d2796041 |
 test1-1@test1.jp | 3          | 0x313337    | 0x728605f8718210896fcbc05ee3b959a5 |
 test1-1@test1.jp | 1          | 0x313435    | 0x383ffc303e38e9e7e968ace5f0e6f383 | Access-Accept
 test2-1@test2.jp | 3          | 0x313531    | 0x531adb59571fce1c84ab779c9d2a4ae7 | Access-Reject
 test2-1@test2.jp | 1          | 0x313537    | 0x216507c925601228da810864bf67fee9 | Access-Reject
 test2-1@test2.jp | 2          | 0x313633    | 0x0f2852560b2d479c676cc189f0e17216 | Access-Reject
 test2-1@test2.jp | 3          | 0x313639    | 0x8e05d58c8a00c09d504faee9e0f995a3 | Access-Reject
 test2-1@test2.jp | 1          | 0x313735    | 0xbe01bcebba04a91c71bcc1fe134576ff | Access-Reject
 test2-1@test2.jp | 2          | 0x313831    | 0xa92d700cad2865d88d8f0d055f95b2fe | Access-Reject
 test2-2@test2.jp | 3          | 0x313837    | 0xc798327bc39d2780d69c60173951825e | Access-Reject
 test2-2@test2.jp | 1          | 0x313933    | 0x27a3676223a67210d00c1b7743193264 | Access-Reject
 test2-2@test2.jp | 2          | 0x323030    | 0xb6acb9fbb3aaaccb9fed5bb189489c57 | Access-Accept
 test3-1@test3.jp | 1          | 0x323036    | 0xe24a83ede64f9631458cfead9e48d999 | Access-Reject
 test3-1@test3.jp | 2          | 0x323132    | 0x127d5fe016784aa421372b147cb538d2 | Access-Reject
 test3-1@test3.jp | 3          | 0x323139    | 0xc0b42736c5b232721d87057a3432c961 | Access-Accept
```

Fig. 7 Examples of RC discovery.

Table 10 Authentication retry parameters of various devices (averaged over 10 retries).

device	number of continuous retries	first interval (sec)	second interval (sec)	total time (sec)	1 auth time (sec)
iPhone 8	3	11.41	0.60	12.41	0.40
Nexus 7	more than 10	11.59	11.81	23.75	0.35
Moto G6	5	13.74	13.14	27.26	0.37

RC discovery takes place for this base realm.

Lines 4–9 of the SQL log file are for an Android device with the username test2-1@test2.jp. The device was intentionally configured with an incorrect password. The Hub Proxy received an Access-Reject packet even for the correct destination (proxy2), and RC discovery continued. Then, the device was reconfigured with another username test2-2@test2.jp in the same realm as the previous one. As shown in line 12, the Hub Proxy received an Access-Accept packet, and the realm was registered. Android devices seem to return Access-Reject packets explicitly when server authentication fails as we explained earlier.

The last three lines are for an Android device with the username test3-1@test3.jp. The Hub Proxy received an Access-Accept packet from the correct destination (proxy3).

As a result, we have confirmed that the proposed RC discovery system works as designed. All Access-Request packets with correct realms that were not registered on the Hub Proxy ultimately reached their correct authentication servers. In addition, we confirmed that the Hub Proxy stores the forwarding address correctly when the authentication has been successful once, and that subsequent authentication requests go smoothly without RC discovery.

The functionality is good since all possible combinations of the realm–RC mappings have been tested. In order to check the practicality of the method, we carried out the following additional testing. We created 3,000 accounts with random realms (institution names) per RC, i.e., 9,000 in total, and fed them to the Hub Proxy. We confirmed that all the realms were mapped to the correct RCs without any failure cases. Since the number of eduroam member operators is at most around 1,000 per country as of this writing and since we have not seen any RC larger than this so far, the proposed system is considered to have enough capacity with respect to the number of operators.

Table 10 shows the measured parameters of the authentication retry functions for each device used. We counted the number of continuous retries and measured the average time per authentication sequence over 10 trials. The first interval in Table 10 shows the time between the first and second trials. The second interval shows the time between the second and third trials. The total time is the amount of time required for the Hub Proxy to finish the three trials and to succeed in authentication. The "1 auth time" is the amount of time required for the Hub Proxy to finish one authentication sequence.

As shown in Table 10, the number of continuous retries on iPhone 8 was fairly low. This limitation may result in a worse user experience when the number of RCs becomes larger. However, we do not expect this to be a serious problem since RC discovery only takes place at the first connection of a new RC. In general, some Operating Systems (OSs) have a longer pause after continuous authentication failures and resume the authentication attempts later. Thus, there is still a chance for the device to reach the correct authentication server. In addition, each RC is expected to have many users and their multiple devices perform authentications using the same realm. The effective number of authentication requests is much larger than the number of RCs in such a situation.

Since there have been only a few large RCs in the world so far, we assumed that the number of RCs would not be so large in this work. Our recommendation is that new RCs be designed to utilize the realms embedded with RC-specific keywords so that the RCs can be easily identifiable.

3.3 DoS Suppression

To simulate a DoS attack on the Hub Proxy, we created a large number of Access-Request packets including those with non-existent realms, and sent them from a device to the Hub Proxy. As a result, the Hub Proxy successfully blocked the authentication requests after the specified number of authentication retries.

As explained in Section 2.5.5, in an actual roaming system, the AP system at each SP should handle suppression of DoS attacks locally and should try to minimize congestion on the identity federation network.

4. Conclusions

In this paper we address the realm-based routing problem that arises in a roaming system connecting some large RCs, each of which has many realms. We developed a method for automatic RC discovery and routing, which is used by the regional Hub Proxies in our inter-federation roaming architecture developed earlier. Our main objective is to automate the routing table creation removing the additional labor at the RC side in particular. Our final goal however is to achieve an inter-federation roaming system. The new method works with conventional 802.1X-based systems and maintains backward compatibility even in Passpoint/NGH environments. The effectiveness of the method was confirmed using a virtual inter-federation roaming network. The most important advantage of our solution compared with existing ones is that no IdP or SP is affected in the presented framework. Since there is no additional system or task on the RC side, the system is expected to contribute to faster development and adoption of large-scale WLAN roaming systems.

Our future work includes further testing of the system in a real large-scale roaming environment to analyze the performance of RC discovery in detail. We have been operating an NGH-based roaming system in Japan since 2017 [15], and its proxy will be a good candidate for prototyping and testing after other large RCs have been connected in the future. The automatic discovery mechanism we propose in this paper would also be useful for finding active paths, and we will seek for more applications that improve the service availability of WLAN roaming systems.

References

[1] Wireless Broadband Alliance: NEXT GENERATION WI-FI, available from ⟨https://www.wballiance.com/what-we-do/next-generation-wi-fi/⟩ (accessed 2019-12-17).

[2] Wi-Fi Alliance: Wi-Fi CERTIFIED Passpoint^{TM} (Release 2) Deployment Guidelines Rev 1.1 (2016).

[3] IEEE Std 802.1X-2010, Port-Based Network Access Control (2010).

[4] Wireless Broadband Alliance: City Wi-Fi Roaming, available from ⟨http://worldwifiday.com/city-wi-fi-roaming/⟩ (accessed 2019-12-17).

[5] Florio, L. and Wierenga, K.: Eduroam, providing mobility for roaming users, *Proc. EUNIS2005* (2005).

[6] govroam, available from ⟨https://govroam.be/⟩ (accessed 2020-04-13).

[7] Wierenga, K., Winter, S. and Wolniewicz, T.: The eduroam Architecture for Network Roaming, IETF RFC7593 (2015).

[8] Goto, H.: Inter-roaming architecture for eduroam/govroam/public WLAN integrated services, Mobility Day, TNC17 (2017), available from ⟨https://wiki.geant.org/display/TFMNM/Mobility+Working+Group+Meeting+at+TNC17⟩ (accessed 2019-12-17).

[9] Rigney, C., Willens, S., Rubens, A. and Simpson, W.: Remote Authentication Dial In User Service (RADIUS), IETF RFC2865 (2000).

[10] Yamaki, H., Yamada, Y., Kato, Y., Kobayashi, E., Saotome, Y. and Matsumoto, D.: Integration of Wifi Services Based on the IEEE802.11u Standard, *Proc. CCATS 2015*, pp.132–137 (2015).

[11] Wi-Fi Alliance: Hotspot 2.0 Specification Version 3.0 (2019).

[12] Irie, K. and Goto, H.: Automatic Roaming Consortium Discovery and Routing for Large-Scale Wireless LAN Roaming Systems, *Proc. IEEE CAMAD 2018*, pp.374–379 (2018).

[13] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J. and Levkowetz, H.: Extensible Authentication Protocol (EAP), IETF RFC3748 (2004).

[14] Microsoft Security Advisory, Unencapsulated MS-CHAP v2 Authentication Could Allow Information Disclosure, available from ⟨https://docs.microsoft.com/en-us/security-updates/SecurityAdvisories/2012/2743314⟩ (accessed 2019-12-17).

[15] Goto, H.: Cityroam, Providing Secure Public Wireless LAN Services with International Roaming, *Proc. RTUWO'18*, pp.204–208 (2018).

Kazunari Irie graduated from the School of Engineering, Tohoku University in 2018, and received his M.Sc. degree in information science from the Graduate School of Information Sciences, Tohoku University in 2020. His research interests include information security, identity federation, and authentication systems for wireless networks such as eduroam.

Hideaki Goto received his M.Sc. degree in physics in 1992, and his Ph.D. degree in engineering in 1995 from Tohoku University in Japan. He has been an Associate Professor of Tohoku University since 2001. His research interests include pattern recognition, document image analysis and recognition, identity federation, wireless network, and information security. He is a member of IPSJ, IEICE, and IEEE Computer Society. He has been a member of the Global eduroam Governance Committee (GeGC) at GÉANT for four terms.