AFaaS: Authorization framework as a service for Internet of Things based on interoperable OAuth

Se-Ra Oh and Young-Gab Kim

Abstract
Internet of Things has become a fundamental paradigm in our everyday lives. However, standards and technologies are often designed without considering interoperability, which is a critical issue for Internet of Things. Internet of Things environment requires interoperability to share resources (e.g. data and services) between heterogeneous Internet of Things domains. The open authorization (OAuth) 2.0 framework that is actively used in Internet of Things (as well as in conventional web environments) also did not focus on interoperability. In other words, the systems that implement the same OAuth 2.0 standard cannot interoperate without additional support. For this reason, we propose an authorization framework as a service. Authorization framework as a service provides an additional authorization layer to support standard authorization capabilities as an interoperable secure wrapper between different domains. Besides, authorization framework as a service supports the four extended authorization grant flow types to issue an interoperable access token, which has a global access scope across multiple heterogeneous domains. With the authorization framework as a service, interoperability can be supported for heterogeneous domains, and token management can also be simple because an interoperable access token can represent several existing access tokens that have local access scopes. Furthermore, this article presents a feasible interoperability scenario, implementation, and security considerations for authorization framework as a service, focusing on Internet of Things platforms.

Keywords
OAuth, authorization, interoperability, interoperable OAuth, Internet of Things

Date received: 30 April 2019; accepted: 16 January 2020

Handling Editor: Naveen Chilamkurti

Introduction
The open authorization (OAuth) 2.0 framework was established to delegate limited access to a third-party application. This standard has been widely used not only in the conventional HTTP-based web environment but also in the Internet of Things (IoT) environment, because it reduces the burden of the constrained device for authorization. The existing OAuth 2.0 framework, however, is not well adapted to IoT environments because it does not focus on interoperability.

IoT environment is fragmented, even though connectivity is a critical issue in IoT: interoperability between things is required to share resources (e.g. aggregated sensing data and various services) among heterogeneous IoT domains. The design principles of the OAuth 2.0 standard are flexibility and extensibility. The standard left several components partially or entirely undefined, to adapt OAuth 2.0 according to individual requirements. For this reason, authorization servers in
IoT that implement the same OAuth 2.0 standard may be incompatible: they may differ in authorization capabilities, endpoints (for token issuance and authorization), token formats, and so on. To solve this problem, the OAuth-based authorization framework can be virtualized as an interoperable authorization layer for heterogeneous IoT domains with a proper authentication scheme, as shown in Figure 1.

Therefore, we propose the concept of authorization framework as a service (AFaaS), which extends the existing OAuth 2.0 framework to provide an interoperable authorization layer that supports standard authorization capabilities. Authentication is also an important issue; however, this article focuses on the authorization. AFaaS can be adopted as a secure wrapper in front of the existing security components to provide interoperability, even if every system has its own security components. Moreover, AFaaS extends conventional authorization grant types to issue an interoperable access token (IAT), which has a global access scope across multiple domains. When using an IAT, resources can be shared with heterogeneous IoT domains, and token management (e.g., token refreshing and revocation) becomes easier, because the IAT can represent several conventional access tokens that have local access scopes. The IAT can also be issued when multiple resource owners or clients are involved in a single domain.

The remainder of this article is organized as follows. Section “Background” analyzes related work and the existing OAuth 2.0. Section “Interoperable OAuth 2.0 framework” describes the proposed AFaaS, including four extended authorization grant flows. Section “A feasible scenario and security considerations” presents a feasible scenario and security considerations for AFaaS, focusing on IoT platforms. Section “Discussion” shows some discussions and, finally, section “Conclusion and future work” presents the conclusion and future work.

**Background**

**Related work**

Although the proposed AFaaS can be applied to diverse OAuth-based systems, our primary target environment is IoT platforms that need high interoperability to provide connectivity between things. This section presents four related projects (BIG IoT (“Bridging the interoperability gap of the IoT”), INTER-IoT (“Interoperability for heterogeneous IoT platforms”), symbIoTe (“Symbiosis of smart objects across IoT environments”), and VICINITY), which are parts of the IoT European Platform Initiative (IoT-EPI) project and were proposed to achieve IoT platforms’ interoperability.

BIG IoT5,6 provided a generic and unified web application program interface (API) to enable access to applications and services from multiple IoT platforms. With the common web APIs, software to access different IoT platforms can easily be developed.

INTER-IoT was proposed to make things and services interoperate together. This project supports interoperability by adopting three main solutions: INTER-LAYER, INTER-FRAMEWORK, and INTER-METHODOLOGY.5,6

symbIoTe was designed to provide a flexible interoperability framework across four-layered domains: application domain, cloud domain, smart space domain, and smart device domain. In particular, it allows the platform to select a proper level of integration of symbIoTe-specific services in their platforms. For example, level 2 (L2) enables platforms to interact with each other using an interworking interface.7,8

Finally, VICINITY connects IoT environments with decentralized interoperability. In this solution, things are connected to a social network named “virtual neighborhood,” where the owners of things control their resources using the VICINITY neighborhood manager (VNM).9

The four solutions that aim to provide interoperability for IoT platforms have goals similar to our research. In addition to the four projects, Koo et al.10 also studied interoperability among heterogeneous IoT platforms based on device identification of the platforms. However, no research exists regarding the interoperability between IoT platforms based on OAuth 2.0. The OAuth 2.0 framework, which is the de facto standard for protecting web-based APIs, is becoming essential, because web-based APIs become important to support various functions in IoT environments. Therefore, this study extends the conventional OAuth 2.0 framework to be interoperable, in contrast with the existing research on IoT platforms’ interoperability.
Besides, related work includes research on OAuth-based security frameworks in diverse domains. Our study focuses on IoT, which is our primary target environment. A brief analysis of OAuth-related research for authentication and authorization in IoT environments is provided as follows.

“Authentication and authorization for constrained environments” (ACE) working group defined ACE-OAuth for authentication and authorization of IoT devices in constrained environments, such as IoT and machine-to-machine (M2M). The ACE-OAuth framework is mainly based on OAuth and constrained application protocol (CoAP), which is a lightweight web transfer protocol.

Cirani et al. proposed a flexible authorization framework, named OAuth-based authorization service (IoT-OAS), that can be easily integrated into an existing system. By delegating the authorization-related process to the IoT-OAS, IoT devices can be relieved from security burden in a way.

Kim et al. built an OAuth- and CoAP-based authentication and authorization system. Besides, they used a token cache to reduce the network traffic and authorization server load and conducted a performance evaluation of the framework.

Sciancalepore et al. designed and implemented a lightweight and flexible authentication and authorization framework based on open standards, such as CoAP, OAuth 2.0, and IETF (Internet Engineering Task Force) tokens (i.e. Bearer, JSON Web Token (JWT), and Proof of Possession (PoP)).

Jung and Jung proposed a personal OAuth authorization server run on a user’s mobile phone. The authorization server is generally placed on a trusted server, such as Amazon and Google; however, this study proposed users to control their authorization server.

Sucasas et al. modified the conventional authorization code flow of OAuth 2.0 to enhance user privacy using a pseudonym-based signature scheme for smart city mobile applications. This research focused on extending the existing OAuth 2.0 framework for user authentication because the original OAuth 2.0 framework mostly handles authorization.

Oktian et al. designed and implemented an authorization framework, OAuthkeeper, to protect REST API in a software-defined network (SDN) environment based on OAuth 2.0. This research addressed challenges specific to SDN environments.

Siris et al. proposed an authorization framework based on blockchain and OAuth 2.0 to provide delegation of authorization for constrained IoT nodes. In this work, the authors combined OAuth 2.0 with blockchain to allow authorizations to be linked to payments so that the OAuth 2.0 message flow for the payments remains on the blockchain.

Jonnada et al. also proposed an authorization and access control framework for an input-constrained collaborative appliance for remote-help (CARE) system using the OAuth 2.0 authorization framework. The framework is designed to allow workers to grant helpers access to remote-controlled resources. In addition, the authors protect malicious behaviors of the helpers by limiting the scope in access token.

Alonso et al. proposed a model to integrate FIWARE OAuth 2.0-based authorization services with electronic identification and trust services (eIDAS) and implemented the system to validate their model. The model can support interoperability by delegating authorization in third-party application; however, it requires eIDAS infrastructure to provide the interoperability. On the other hand, our proposed scheme does not need an additional infrastructure to support interoperability because we extend the OAuth 2.0 itself to ensure interoperability.

We have described related work on OAuth 2.0-based authorization in IoT environments. However, most existing OAuth-related researches only focused on a single domain, even though connectivity between different domains (to share resources) is becoming essential in IoT. We previously researched interoperable OAuth 2.0 for heterogeneous IoT platforms; however, the study focused on the access control framework. Therefore, this article focuses explicitly on AFaaS, which supports an interoperable OAuth 2.0 framework considering multiple different domains to provide interoperability among them.

Overview of the existing OAuth 2.0 framework and its limitations

OAuth 2.0 framework generally works as shown in Figure 2.

To request protected resources from a resource server, a client obtains an authorization grant from the resource owner, or its credentials (e.g. ID and password of the resource owner). The client that has the grant then forwards it to an authorization server, receives an access token, and finally can request the protected resource using the issued access token.

The roles and main terms of the OAuth 2.0 framework are defined as follows.

Resource owner. Resource owner is an entity that can grant access to protected resources to a third party. It can be a person or an application.

Client. Client is an application requesting an access token to an authorization server or requesting protected resources from a resource server with an access token. OAuth 2.0 standard defines two client types:
confidential client or public client. A confidential client can maintain the confidentiality of its credentials (e.g. client ID and password). A public client cannot maintain the confidentiality of its credentials.

**Resource server.** Resource server is a server that stores protected resources (e.g. sensing data and web APIs). This server handles a resource request with an access token and responds to the request based on an access decision of the authorization server.

**Authorization server.** Authorization server is a server that issues an access token based on an authorization grant of a resource owner. To obtain an authorization grant from the resource owner, the owner must be authenticated in advance.

**Access token.** Access tokens represent a delegated authorization grant of a resource owner. Access tokens have two important properties: scope and expiration time. The scope indicates resources that can be accessible with this access token. The expiration time indicates the time available for this access token.

**Refresh token.** Refresh tokens are used to issue new access tokens if the previous access tokens are expired or invalid. Refresh tokens are issued by the authorization server, as the access token, but have a longer expiration time compared to access tokens. In general, a refresh token can be represented as an authorization grant of a resource owner, because refresh tokens are used to mint access tokens.

OAuth 2.0 framework standard presents four authorization grant types to issue an access token: authorization code, implicit grant, resource owner password credentials, and client credentials. Authorization grant types are designed for a specific purposes depending on the requirements. The problem, however, is that the grant types are designed to be used in a single domain. If a client wants to access protected resources stored in other domains, separate access tokens must be obtained for each domain.

Moreover, several parts of the standard are undefined (such as the capabilities of an authorization server and the token formats). This makes the OAuth 2.0 framework flexible but raises the interoperability issue. Consequently, systems cannot interoperate even if they implement the same OAuth 2.0 standard because of several different proper ways to implement the OAuth 2.0 framework. Therefore, we should request protected resources in heterogeneous domains with multiple access tokens issued from each domain. In this case, we must also manage all the access tokens, so that they are not leaked and expired. In addition, the token owner must revoke the tokens when the owner no longer needs the tokens, or when an attacker compromises the tokens for security reasons.

To achieve a more seamless interoperability for the OAuth 2.0 framework, we need extended authorization grant types considering multiple resource owners and authorization servers in heterogeneous domains (e.g. IoT platforms). In addition, a virtualized authorization layer must be adopted to provide standard authorization capabilities in the diverse domains as an interoperable secure wrapper. Using the extended grant types, an IAT that has a global access scope (across the heterogeneous domains) can be issued. The IAT provides interoperability among IoT platforms based on its authorization grants acquired from multiple resource owners. Furthermore, it can make token management

---

**Figure 2.** OAuth 2.0 flow overview.
(e.g. token refreshing and revocation) easy, because it can represent numbers of the existing access tokens that have local access scopes.

**Interoperable OAuth 2.0 framework**

This section presents the target environment and proposed AFaaS. In particular, it describes the requirements and benefits of the virtualized authorization layer and four extended authorization grant flows.

**Target environment**

The proposed AFaaS can be used in different environments to provide interoperability between heterogeneous domains as well as IoT. This article, however, focuses explicitly on IoT platforms as a target environment for AFaaS.

IoT platforms provide common functions, such as messaging and security, for various things (e.g. IoT device and actuator). Platforms may need to provide essential functions for tens of billions of things expected to be created in the near future. However, authentication and authorization schemes in diverse IoT platforms have been implemented based on the specific security goals. In particular, OAuth-based access control frameworks that are actively used in IoT platforms have not been providing interoperability among IoT platforms, even though sharing resources is a critical issue for IoT environments. For example, to use resources in conventional OAuth-based IoT platforms, an individual token for each IoT platform must be issued and managed (i.e. it must be kept securely, refreshed when it expires, and revoked if it is compromised or no longer needed). In addition, translating the token format of a specific IoT platform to one of a target IoT platform, to provide interoperability between IoT platforms, could be another challenge if a lot of IoT platforms participate in an interworking.

To this end, AFaaS, which provides a standard authorization functionality, can be deployed in heterogeneous IoT platforms to support interoperability, as shown in Figure 3. Because the OAuth 2.0 framework was designed to use HTTP, IoT platforms must support HTTP.

In Figure 3, authorization layers are deployed in front of IoT platforms A and B using AFaaS. With the interoperable authorization layers, which support the interoperable OAuth 2.0 framework, the heterogeneous domains (i.e. IoT platforms A and B) can be interoperable. The requirements and benefits of the virtualized authorization layer are described in section "Requirements and benefits of the virtualized authorization layer." Furthermore, a feasible interoperability scenario based on Figure 3 is described in section "A feasible scenario and security considerations."

**Requirements and benefits of the virtualized authorization layer**

The OAuth 2.0 framework was designed without considering interoperability. Several parts such as authorization server capabilities and token format were undefined in the existing OAuth 2.0 framework. Even though heterogeneous systems implement the same OAuth 2.0 standard, they cannot interoperate with other systems because of their individual implementations. Therefore, a standard authorization layer, which
can be easily applied to heterogeneous systems and supports extended authorization grant types used to issue an IAT, is required to provide the same authorization capabilities using virtualization.

The main requirements and benefits regarding the virtualized authorization layer are as follows.

- **The authorization layer must provide standard authorization capabilities to every domain.** To this end, it can be virtualized as shown in Figure 3. Using this approach, the proposed authorization layer can be easily added in front of the existing domain-specific components to support secure interoperability with other domains. Moreover, because the structure for IAT is the same, any domain can understand the IAT forwarded from other domains. Therefore, based on the IAT, resources in different domains can be easily shared without additional processes (e.g., translation of heterogeneous token structure or request of access tokens for each domain) because of the global access scope of the IAT. With the IAT, token management (e.g., token issuing, refreshing, and revocation) is also simple, because an IAT can represent multiple access tokens with local access scopes.

- **Easy-to-use interface to define a platform-independent access control policy (e.g., extensible access control markup language (XACML))** is required for interoperability. Security manager for a specific domain should determine whether the domain will grant access for other domains or not, by defining a domain-level interoperability policy (DL-IP) with concrete conditions. Moreover, access from other domains might be undetermined according to only the DL-IP. For example, resource owners in a single domain can have different opinions on whether they want to grant access from other domains or not. In this case, a security manager can delegate the access decision for each resource to the resource owners. Resource owners would then determine whether resources can be accessed from other domains or not by defining resource-level interoperability policy (RL-IP). For a security reason, the default configuration of the DL-IP and RL-IP must be “Not granted” for all accesses from other domains.

- **Endpoints for authorization and token management (e.g., token refreshing and revocation) in the authorization layer should be fixed.** Based on the fixed endpoints, a client in other domains can request interoperable resources through the endpoints with minimal prior knowledge. For example, we can easily infer the token endpoint (e.g., http://domain_A/AFaaS/token) of domain A, which deploys the proposed authorization layer, if we know the IP address or domain name for domain A, because the token endpoint is always the same (i.e. http://*/AFaaS/token).

- **A domain that deploys the authorization layer must be able to conduct its required process (e.g., a specially designed security function for the domain) using domain-specific components.** Furthermore, if capabilities for interoperability with other heterogeneous domains are not required, a client should be able to use the domain-specific components directly to access the domain without the authorization layer.

### Extended authorization grant flows to issue an IAT

To issue an IAT, AFaaS uses four extended authorization grant flow types: authorization code for multiple domains (ACMD), implicit for multiple domains (IMD), multiple resource owners’ password credentials (MROPC), and multiple clients’ credentials (MCC). Conventional authorization grant flow types only considered a single resource owner, client, and domain. However, the proposed grant flow types (i.e., ACMD, IMD, MROPC, and MCC) can handle one or more resource owners, clients, and domains. By obtaining authorization grants from multiple resource owners in heterogeneous domains, a client can access protected resources across the domains with an IAT that has a global access scope. In addition, the IAT makes token management easy because it can represent a number of existing access tokens with a local access scope. Assume that there are three IoT platforms A, B, and C and resources 1, 2, and 3 on each IoT platform. In general, a client must request three access tokens to use each resource in the three IoT platforms and the client must manage all the access tokens. To solve these problems, an IAT can be used to simplify the token management and provide interoperability.

The original authorization code grant flow is generally used to exchange an authorization code that represents an authorization grant of a resource owner for an access token, which has only a local access scope (e.g., access scope: “A resource in domain A”). ACMD follows the original authorization code flow. To request an IAT, a client needs one or more authorization grants of resource owners in heterogeneous domains. The overview of ACMD is shown in Figure 4.

Figure 4 shows resource owners related to the access scope requested by a client, and authorization servers in multiple domains that deploy the proposed authorization layer, supporting the extended authorization grant flows. Resource owners and authorization servers are grouped together in Figure 4; however, they are actually distributed. The abstracted flow of ACMD is as follows:
1. A client initiates the ACMD grant flow.
2. The client obtains authorization grants and sends the grants to an authorization server.
3. The authorization server validates the grants and gives the client an authorization code.
4. The client sends the code.
5. The authorization server issues an IAT if the code validation is successful.

The basic principle of ACMD is very similar to the original authorization code flow. An IAT is issued based on resource owners’ authorization grants. A more detailed flow is shown as a sequence diagram in Figure 5.

1. A client initiates the ACMD flow.
2. User-agent requests an IAT from the token endpoint of authorization server 1 with the client’s desired access scope, which indicates one or more domains and its resources (e.g. “Access Scope”: {“domain A”: “resource A,” “domain B”: “resource B”}), and redirection URI. Note that the user-agent can also request an IAT from authorization server 2’s token endpoint, because all authorization servers share the same capabilities through the virtualized authorization layer. This may help mitigate the bottlenecks of a single authorization server.
3. Authorization server 1 redirects the user-agent to the login page.
4. The user-agent shows the login page to the user (i.e. resource owner (RO)). The login page contains input fields, as many as the number of resource owners. For example, if the client

---

**Figure 4.** Overview of ACMD.

**Figure 5.** Detailed flow of ACMD.
wants two resources of two owners, the login page will consist of two ID and password input fields for the two resource owners. In this example, we assume that the client requests a global access scope for “resource A” and “resource B” in “domain A” and “domain B,” respectively.

5. The user enters the required credentials of the resource owners related to the requested access scope (e.g. two IDs and passwords of the resource owners for resources A and B).

6. The user-agent forwards the credentials to authorization server 1.

7. Authorization server 1 validates the interoperability policies (i.e. DL-IP and RL-IP) and the credentials related to domain A’s resource owner. Note that ACMD will continue even if the interoperability policies of domain A for the requested access scope are not fulfilled to issue an IAT, because “domain B” that stores “resource B” could accept the request for an IAT.

8. Authorization server 1 then requests validation of the interoperability policies and the credentials related to domain B’s resource owner. In this step, authorization server 1 can send the validation request to authorization server 2, because the client specified the location of “domain B” in the access scope when it initiated the ACMD flow. In addition, authorization server 1 can infer authorization server 2’s validation endpoint for the credentials because of the same authorization capabilities. For example, if the validation endpoint to check a resource owner’s credentials of authorization server 1 is “http://domain_A/AFaaS/auth_cre,” authorization server 2’s credential validation endpoint will be “http://domain_B/AFaaS/auth_cre.”

9. Authorization server 2 checks the DL-IP and RL-IP for “domain B” and “resource B.” If access to “resource B” is allowed by the policies, authorization server 2 then validates the credentials of resource owner 2.

10. Authorization server 2 sends the validation result to authorization server 1. Note that, if the client requested more resources in other domains, C and D, authorization server 1 has to send more validation requests to domains C and D.

11. Authorization server 1 updates the access scope for an IAT based on the validation results of authorization servers 1 and 2. If the credentials that the client entered are correct, the server will not modify the access scope. However, if the validation result from authorization server 2 is “incorrect,” authorization server 1 will update the access scope to ensure that the client cannot access “resource B” in “domain B.”

12, 13. Authorization server 1 redirects the user-agent back to the client, using the redirection URI with the issued authorization code.

14. The client sends a request for an IAT to the token endpoint of authorization server 1 with the authorization code and redirection URI.

15. Authorization server 1 validates the authorization code and checks that the redirection URI received is the same as the URI in step 2. If the final check is successful, the authorization server issues an IAT that has a global scope across domains A and B.

After the client successfully receives an IAT, the client can access the resources stored in domains A and B in the scope of the IAT.

The IMD grant type is not much different from ACMD, because it is a simplified version of the ACMD. Figure 6 shows an overview of the IMD grant flow.

As shown in Figures 4 and 6, the flow of the IMD grant type is very similar to the ACMD flow. The only difference is that the authorization server directly sends an IAT to the client instead of an authorization code in step 3 of the IMD flow. Figure 7 shows an overview of the MROPC grant flow.
In MROPC, the confidential client exchanges the resource owners’ credentials for an IAT without any user-agent. A detailed flow is shown in Figure 8.

1. RO 1 provides its credentials (e.g. username and password) to the client.
2. RO 2 provides its credentials to the client. If the client knows the credentials in advance, steps 1 and 2 can be skipped.
3. The client initiates the MROPC for an IAT with the credentials related to the desired access scope. The requested access scope will consist of the domains and resources of ROs 1 and 2 (e.g. “Access Scope”: “domain A”: “resource 1,” “domain B”: “resource 2”).
4. The authorization server (AS) 1 in domain A validates the interoperability policies (i.e. DL-IP and RL-IP) and credentials of RO 1 and then stores the validation result.
5. AS 1 sends a validation request to AS 2 in domain B.
6. AS 2 validates the interoperability policies and credentials of RO 2.
7. AS 2 sends the validation result.
8. AS 1 updates the access scope requested from the client based on the validation results. For example, if interoperability policies of domain B are inactive, the scope will be modified to limit access to domain B’s resource. In addition, even if the policies are allowed to interoperate other domains (i.e. domain B), AS 1 will update the scope to ensure that the client cannot access resource 2 in domain B when the credentials validation result from server 2 is “invalid.”
9. AS 1 issues an IAT if all validation processes are successful.

Finally, Figure 9 shows an overview of the MCC authorization grant flow. The MCC authorization grant type is not much different from MROPC. A client can request an IAT using the client’s credentials instead of the resource owner’s credentials. Because the client’s credentials can represent an authorization grant for the client, a client that has the credentials can be granted access to the client-related domains. Therefore, clients (i.e. applications) can simply interoperate to provide cross-domain services with an IAT issued using MCC. Note that, as for other grant types, the access scope of the IAT will be modified by the authorization server if any credentials are invalid.
A feasible scenario and security considerations

Section “Interoperable OAuth 2.0 framework” described four extended authorization grant flows to issue an IAT. This section describes a feasible scenario, implementation, and related security considerations specifically based on MROPC and MCC. These authorization grant flows are more suitable for IoT, which is our primary target environment, than other flows (i.e. ACMD and IMD). MROPC and MCC are not only much more straightforward than other flows but also can work without human interaction, because they use user/client credentials that represent the resource owner’s authorization grant to obtain access permission.

Flowchart for the scenario

Figure 10 shows an overview of a feasible interoperability scenario based on IoT platforms. In the scenario, a client uses a resource in IoT platform B with an IAT issued from IoT platform A. To issue an IAT, IoT platforms must deploy authorization layers that support extended authorization grant flows. The scenario consists of two phases. The first phase consists of the request steps for an IAT (i.e. steps 1–6). The second phase consists of the request steps for a resource (i.e. steps 7–10). The detailed flow of Figure 10 is as follows.

1. A client requests an IAT with credentials for resources 1 and 2. In the case of MROPC, the client will use resource owner’s credentials (e.g. IDs and passwords); however, the client’s credentials (e.g. client IDs and secrets) will be used in the case of MCC. This scenario assumes that credentials are registered in advance. Even though a client sends the IAT request to IoT platform A in this scenario, the client also can send the request to IoT platform B, because IoT platforms A and B provide the same

*Creds.*: credentials, RL-IP: resource-level interoperability policy, DL-IP: domain-level interoperability policy.
authorization capabilities based on their authorization layer.

2. AS 1 checks the DL-IP and RL-IP, which are related to requested resource 1. If the policies allow access from the client domain, the AS then validates the credentials for IoT platform A.

3. AS 1 requests a validation of interoperability policies and credentials related to resource 2 from IoT platform B.

4. AS 2 checks the DL-IP and RL-IP for resource 2. AS 2 then validates the credentials forwarded from AS 1 when the policies allow the requested interoperability.

5. AS 2 sends the validation result to AS 1.

6. AS 1 checks the validation results of steps 2 and 4 and updates the access scope based on the validation results. AS 1 then creates an IAT that has the updated access scope and forwards the original IAT to the client. Because the original IAT can be used to request resources related to the access scope, it must be only transmitted to a legitimate client. In addition, a hashed IAT is shared with AS 2. With the hashed IAT, AS 2 can validate the original IAT sent by a client when it requests a resource.

7. The client requests a resource with the issued original IAT.

8. AS 2 validates the IAT with the hashed IAT. In addition, policies for the resource should be checked whether the request is allowed or not.

9. Domain-specific functions can be processed in this step. IoT platform B could perform additional functions, for example, attribute-based access control (ABAC) for in-depth security. After the domain-specific functions are completed, IoT platform B retrieves resource 2 and sends it to AS 2.

10. AS 2 forwards resource 2 to the client as a reverse proxy. When used as a reverse proxy, the authorization server can introspect every request from other domains and returns a resource on behalf of the back-end resource server. Furthermore, a client cannot reach to resource server because it can communicate only with the front-end server (i.e. authorization server).

The client also can access resource 1 stored in IoT platform A with the same IAT used to access resource 2 in the scenario because the IAT has a global access scope across heterogeneous domains (i.e. IoT platforms A and B).

Implementation

We implemented the feasible interoperability scenario using FIWARE and oneM2M (i.e. Mobius developed by Korea Electronics Technology Institute (KETI)), which are promising open-source IoT platforms. Figures 11 and 12 show the real environments that we used to implement the interoperability scenario. Note that we assumed the two IoT platforms have a trust relationship and DL-IP and RL-IP are already defined to allow interoperability between the IoT platforms.

In the case of the FIWARE, there is a FIWARE server (i.e. Orion context broker) on desktop PC and FIWARE client on Raspberry Pi. FIWARE client senses temperature and humidity values using DHT11 sensor and then sends the value to the FIWARE server, which stores the value as resources of FIWARE (i.e. /entities/FIWARE_sensor/attrs/temperature and /entities/FIWARE_sensor/attrs/humidity). On the other hand, oneM2M server (i.e. Mobius) and client are deployed on another Raspberry Pi. The oneM2M client senses the air pollution value and sends the data to the oneM2M server as a resource of oneM2M (i.e. /mobius-yt/OneM2M_sensor/AirQuality/latest_status).

In addition, we modified the OAuth 2.0 library, OAuth2orize, to implement MCC used to validate the feasible interoperability scenario. We added multiple_client_credentials as a new grant type in the library and two HTTP endpoints for credential validation and token sharing (i.e. /validate_creds and /share_token) using Node.js Express. The endpoint, /validate_creds,
takes care of validation request for client credentials and another endpoint, /share_token, receives a hashed IAT that contains access scope including its IoT platform. Authorization servers, which support interoperable authorization among IoT platforms based on the modified OAuth 2.0 library, are deployed on the FIWARE and oneM2M to issue an IAT and validate the IAT. Figure 13 shows the feasible interoperability scenario with specific deployment and configuration.

In Figure 13, a requester requests an IAT to FIWARE’s AS using two pairs of client credentials registered in FIWARE and oneM2M, respectively, and then requests oneM2M resource (i.e., latest air pollution value) with the IAT. The detailed flow for the scenario is as follows. Note that we assume that all provisioning steps (e.g., registering client) are finished in advance.

1. A requester requests an IAT to FIWARE’s AS with two pairs of client credentials. Figure 14 shows an HTTP request for issuing an IAT, which has a global access scope across FIWARE_sensor and oneM2M_sensor. In addition, the grant type in the request is MCC.
2. FIWARE AS received an IAT validates FIWARE’s client credentials. Then the AS requests validation for oneM2M’s client credentials to oneM2M’s credential validation endpoint (i.e., 192.168.0.24:5000/validate_creds). If oneM2M AS’s credential validation endpoint sends a successful response for the validation to FIWARE AS, the FIWARE AS makes an IAT. In this case, the access scope is not modified because the scopes are all related to the correct client credentials.
3. FIWARE AS shares a hashed IAT with oneM2M AS using its token sharing endpoint (i.e., 192.168.0.24:5000/share_token). The hashed IAT can only be used to validate IAT.
4. If an IAT is created and shared with all related IoT platforms, FIWARE AS forwards the IAT.

Figure 13. Interoperability scenario between FIWARE and oneM2M.
to the requester. Figure 15 shows the issued IAT from FIWARE AS.

5. The requester who has the IAT that has a global access scope across FIWARE_sensor and oneM2M_sensor can access the resources of the FIWARE_sensor and oneM2M_sensor. Figure 16 shows an HTTP request and response for oneM2M_sensor’s resource (i.e. air pollution value) using the IAT issued from FIWARE. If the IAT is invalid, an error message (i.e. Unauthorized) will be returned as shown in Figure 17.

We only requested oneM2M’s resource in this scenario and implementation; however, FIWARE’s resource (i.e. temperature and humidity values) can also be requested with the same IAT. In addition, not only FIWARE and oneM2M but other IoT platforms can be included as one of the IAT’s access scopes.

Security considerations for the scenario

This section presents security considerations for the interoperability scenario. In addition to these considerations, the existing security analysis documents1,24–27 must be referred to, because the extended OAuth 2.0 framework fundamentally follows the conventional OAuth 2.0.

Ensuring availability of the authorization server. A client that has a malicious purpose could request an IAT continuously. In this case, an authorization server has to validate the interoperability policies and credentials of multiple domains that the client requested. To mitigate these types of attacks, the authorization server could limit consecutive requests from the same client. In addition, valid requests for an IAT might be cached to improve the availability of the authorization server according to the security policy.

Security of the credentials and IAT. The security for the credentials and IAT strongly depends on the transport layer security (TLS) as the existing OAuth 2.0 framework. Lightweight TLS such as mbed TLS28 and wolfSSL29 may be used when resources of things (e.g. IoT devices, sensors, and actuators) are constrained. Furthermore, resources will be secured even if a hashed IAT is compromised, because it is only used to validate an original IAT. An attacker also cannot obtain the original IAT from the hashed IAT as long as the attacker cannot break the hash function used to create the hashed IAT.

Consideration for an impersonation attack when the MROPC or MCC is used. In Figure 10, a requester must send the credentials (i.e. multiple pairs of credentials such as IDs
and passwords of resource owners 1 and 2) related to the desired access scope to get a global access scope between the heterogeneous IoT platforms. In this case, the IoT platform that received the credentials can impersonate the resource owners because the credentials represent resource owners’ privilege even if the IoT platforms are trustworthy. Therefore, all credentials transmitted in other domains must be encrypted, so that only a proper IoT platform can check the credentials. For example, the credentials could be encrypted with the public key of an IoT platform to ensure that only the IoT platform, which is a legitimate server, uses the credentials.

Discussion
This section describes two discussions that have not been sufficiently mentioned in this article.

As a service model for the proposed authorization framework
This article implemented the authorization server that supports interoperable authorization among IoT platforms and manually deployed the AS in FIWARE and oneM2M. However, AFaaS including the AS should be provided as a service in the near future based on cloud computing. By embracing this concept, IoT platform vendors could quickly deploy the interoperable authorization layer to support interoperability among IoT platforms in their IoT platforms. For example, if an IoT platform vendor requests AFaaS for their IoT platform, a virtualized Docker container could be created in a cloud server and connected with the IoT platform to provide the AFaaS as shown in Figure 3. Users can then use the interoperability service as if the service were part of the IoT platform through the virtualized container in the cloud server. With this concept, there is no need to manually implement the AFaaS on every IoT platform and client.

Quantitative evaluation for AFaaS
This article did not conduct a quantitative evaluation for AFaaS since we have been implementing AFaaS. We are planning to evaluate AFaaS in terms of computational and network load, time consumed by the proposed grant types (i.e. ACMD, IMD, MROPC, and MCC), and so on when the AFaaS is fully implemented. However, even though we currently did not conduct the quantitative evaluation, we can approximatively predict the performance of AFaaS in a way. For example, the more the IoT platforms related to access scope, the longer it will take to issue an IAT because credentials must be validated by the IoT platforms and the IAT also must be shared among the IoT platforms. If four IoT platforms are included in the access scope to request an IAT, it will consume time twice as long to include two IoT platforms in the access scope for issuing an IAT. However, we are expecting consumed time to issue an IAT is faster than the conventional way (i.e. issuing tokens for every IoT platform) since an IAT is requested once from a requester and responded once to the requester. In addition, databases for access tokens can be managed more efficiently because an IAT can represent multiple conventional access tokens based on a global access scope.

Conclusion and future work
The OAuth 2.0 framework has been widely used to protect the IoT platforms as well as conventional web environments. However, heterogeneous systems that implement the OAuth 2.0 framework cannot interoperate, because the existing OAuth 2.0 framework does not consider interoperability. Therefore, we proposed AFaaS that provides interoperability for heterogeneous domains using a virtualized authorization layer and four extended authorization grant flows. The virtualized authorization layer was presented to share standard authorization capability among diverse domains; the layer is used as an interoperable secure wrapper in front of the domains. In particular, we proposed an IAT that has a global access scope to provide interoperability among heterogeneous domains based on the four extended authorization grant flows (i.e. ACMD, IMD, MROPC, and MCC). The IAT was used not only for interoperability, but also for simplifying token management (e.g. token creation, refreshing, and revocation). Finally, we presented a feasible interoperability scenario, implementation, and security considerations, focusing on IoT platform environments. In the near future, we will fully implement the AFaaS and conduct an in-depth security analysis.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by a grant of the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare, Republic of Korea (grant number: HI18C1140).
ORCID iDs
Se-Ra Oh https://orcid.org/0000-0002-4448-2748
Young-Gab Kim https://orcid.org/0000-0001-9585-8808

References
1. IETF RFC 6749. The OAuth 2.0 authorization framework. https://global.ihls.com/doc_detail.cfm?document_name=1EIT%20RFC%206749&item_s_key=00596668&item_key_date=870231&csf=ECIA&input_search_filter=ASME
2. Vermezan O. Advancing IoT platforms interoperability. Gistrup: River Publishers, 2018.
3. Jell T, Baumgartner C, Bröring A, et al. BIG IoT: interconnecting IoT platforms from different domains—first success story. In: Proceedings of the information technology—new generations: 15th international conference on information technology, 2018, pp.721–724. New York: Springer.
4. Bröring A, Schmid S, Schindhelm CK, et al. Enabling IoT ecosystems through platform interoperability. IEEE Softw 2017; 34: 54–61.
5. Gravina R, Palau C, Manso M, et al. Towards multi-layer interoperability of heterogeneous IoT platforms: the INTER-IoT approach. Cham: Springer, 2017.
6. Pace P, Gravina R, Aloi G, et al. IoT platforms interoperability for active and assisted living healthcare services support. In: Proceedings of 2017 global Internet of Things summit (GIoTS), Geneva, 6–9 June 2017, pp.1–6. New York: IEEE.
7. Soursos S, Zarko IP, Zwickl P, et al. Towards the cross-domain interoperability of IoT platforms. In: Proceedings of the European conference on networks and communications (EuCNC), Athens, 27–30 June 2016, pp.398–402. New York: IEEE.
8. Zarko IP, Soursos S, Gojmerac I, et al. Towards an IoT framework for semantic and organizational interoperability. In: Proceedings of 2017 global Internet of Things summit (GIoTS), Geneva, 6–9 June 2017, pp.1–6. New York: IEEE.
9. Guan Y, Quintero JCV, Guerrero JM, et al. An open virtual neighbourhood network to connect IoT infrastructures and smart objects—VICINITY: IoT enables interoperability as a service. In: Proceedings of 2017 global Internet of Things summit (GIoTS), Geneva, 6–9 June 2017, pp.1–6. New York: IEEE.
10. Koo J, Oh S-R and Kim Y-G. Device identification interoperability in heterogeneous IoT platforms. Sensors 2019; 19(6): 1433.
11. IETF. Authentication and authorization for constrained environments (ACE) using the OAuth 2.0 framework (ACE-OAuth), https://www.semanticscholar.org/paper/Authentication-and-Authorization-for-Constrained-Erdtman-Seitz/ff4e85d41f940e3dec0001d1b832ace19130eb392
12. Cirani S, Picone M, Gonizzi P, et al. IoT-OAS: an OAuth-based authorization service architecture for secure services in IoT scenarios. IEEE Sens J 2015; 15: 1224–1234.
13. Kim KH, Lim Hk, Heo JS, et al. Design and implementation of CoAP authorization framework based on OAuth 2.0. KIPS Trans Comput Commun Syst 2017; 6(8): 329–342.
14. Sciancalepore S, Piro G, Caldarola D, et al. OAuth-IoT: an access control framework for the Internet of Things based on open standards. In: Proceedings of the IEEE symposium on computers and communications (ISCC), Heraklion, 3–6 July 2017, pp.1–6. New York: IEEE.
15. Jung SW and Jung S. Personal OAuth authorization server and push OAuth for Internet of Things. Int J Distrib Sens Netw 2017; 13: 1–11.
16. Sucasas V, Mantas G, Althunibat S, et al. A privacy-enhanced OAuth 2.0 based protocol for smart city mobile applications. Comput Secur 2018; 74: 258–274.
17. Oktian YE, Lee SG and Lam JH. OAuthkeeper: an authorization framework for software defined network. J Netw Syst Manag 2018; 26: 147–168.
18. Siris VA, Dimopoulos D, Fiotiou N, et al. OAuth 2.0 meets blockchain for authorization in constrained IoT environments. In: Proceedings of the IEEE 5th world forum on Internet of Things (WF-IoT), Limerick, 15–18 April 2019, pp.364–367. New York: IEEE.
19. Jonnada S, Dantu R, Shrestha P, et al. An OAuth-based authorization framework for access control in remote collaboration systems. In: Proceedings of the national cyber summit (NCS), Huntsville, AL, 5–7 June 2018, pp.38–44. New York: IEEE.
20. Alonso A, Pozo A, Choque J, et al. An identity framework for providing access to FIWARE OAuth 2.0-based services according to the eIDAS European regulation. IEEE Access 2019; 7: 88435–88449.
21. Oh S-R, Kim Y-G and Cho S. An interoperable access control framework for diverse IoT platforms based on OAuth and Role. Sensors 2019; 19(8): 1884.
22. OASIS. eXtensible Access Control Markup Language, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
23. OAuth2orize. https://github.com/jaredhanson/oauth2orize
24. IETF RFC 6750. The OAuth 2.0 authorization framework: bearer token usage, https://tools.ietf.org/html/rfc6750
25. IETF RFC 7662. OAuth 2.0 token introspection, https://tools.ietf.org/html/rfc7662
26. IETF RFC 7009. OAuth 2.0 token revocation, https://tools.ietf.org/html/rfc7009
27. IETF RFC 6819. OAuth 2.0 threat model and security considerations, https://tools.ietf.org/html/rfc6819
28. mbed TLS. https://tls.mbed.org/
29. wolfSSL. https://www.wolfssl.com/