Journal of Network and Computer Applications 229 (2024) 103922

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Research paper

The universal federator: A third-party authentication solution to federated cloud, edge, and fog

Asad Ali [a,*], Ying-Dar Lin [b], Jian Liu [c], Chin-Tser Huang [c]

[a] National Institute of Cyber Security, Ministry of Digital Affairs, No. 143, Yanping S. Road, Taipei, 100057, Taiwan
[b] National Yang Ming Chiao Tung University, No. 1001, Daxue Rd., East District, Hsinchu City, 300093, Taiwan
[c] University of South Carolina, 550 Assembly Street, Columbia, 29201, SC, USA

ARTICLE INFO

Keywords:
Authentication
Federation
Cloud computing
Edge computing
Fog computing
OIDC
EPS-AKA
802.1x

ABSTRACT

Cloud, Edge, and Fog computing provide computational services to different end users. A federation among these computing paradigms is beneficial, as it enhances the capability, capacity, coverage, and services of cloud, edge, and fog. An authentication method is needed to realize such a federation among cloud, edge, and fog so that a user belonging to one of these computing paradigms can use the services offered by other computing paradigms in the federation without creating a new account. This paper proposes a standard-compliant universal federator that transparently provides third-party authentication among different protocols, used by cloud, edge, and fog, such as 3GPP EPS-AKA, OpenID Connect (OIDC), and 802.1x. The federator provides transparency by using a controller and modules that act as virtual counterparts of the authentication entities in EPS-AKA, OIDC, and 802.1x. These virtual counterparts play multiple roles, depending upon the involved protocols. We deployed a testbed, published our implementation on GitHub, and tested third-party authentication for 16 scenarios across EPS-AKA, OIDC, and 802.1x. The results show that our federator successfully provides third-party authentication while taking 4.07–51.8% of the total authentication time, which ranges between 1.193–3.825 s for 16 scenarios. Some scenarios involving 802.1x take considerably longer due to the bottleneck caused by the 802.1x switch. We also conducted a security analysis to show that our proposed federator fulfills multiple security requirements.

1. Introduction

Cloud computing (Mell et al., 2011) has been one of the most widely used computing paradigms over the years because of its scalability, flexibility, quick deployment, and cost savings advantages. However, the explosion of IoT devices and the introduction of latency-sensitive applications like augmented reality, multimedia streaming, multiplayer gaming, and self-driving vehicles have exposed the fundamental limitation of the cloud computing paradigm. Being far away from end users, the cloud introduces more latency, which is unsuitable for delay-sensitive applications. Edge computing and fog computing (Bonomi et al., 2012) solve the latency problem by bringing cloud services closer to end users.

Edge computing incorporates the capabilities of cloud computing into mobile network operators, as proposed by the European Telecommunication Standards Institute (ETSI). Edge computing is closer to the end users and provides the same services as cloud computing, but its latency, capacity, and computing power are lower than the cloud. Fog computing is like edge computing but closer to the end user. Fog can be considered a small cloud closer to the user that provides services such as storage and computation (OpenFog Consortium et al., 2017), but mobile network operators do not provide it. Although different terms in the literature are used for computing paradigms similar to fog computing, such as edge, mobile, and mobile cloud computing, we will conform to the term fog computing, defined by the Industrial Internet Consortium (OpenFog Consortium et al., 2017).

Each of these computing paradigms has its own advantages over other paradigms. Cloud computing provides more computing power and storage while having latency constraints. Fog and Edge computing relieves the latency issue but cannot provide computing and storage power at a cloud scale. There are different heterogeneous IoT devices with different requirements for different applications. Hence, more than one computing paradigm is required to fulfill such requirements.

A federation among these computing paradigms is necessary so that users can avail the advantages of all computing paradigms. The federation allows different service providers to come together to enhance their capabilities via leasing capacity and other services from each

* Corresponding author.
E-mail addresses: asad.ali@nics.nat.gov.tw (A. Ali), ydlin@cs.nctu.edu.tw (Y.-D. Lin), jianl@email.sc.edu (J. Liu), huangct@cse.sc.edu (C.-T. Huang).

https://doi.org/10.1016/j.jnca.2024.103922
Received 14 March 2024; Received in revised form 19 May 2024; Accepted 17 June 2024
Available online 22 June 2024

other. Thus, such a federation can be considered a comprehensive and full-scale computing platform that can satisfy the demands of all sorts of devices. Depending upon the applications, users may access any of these computing paradigms simultaneously or at different times as per their application requirements, such as powerful computation or low latency.

Moreover, the benefits of federation are two-fold as it allows the subscribers to have one account-service everywhere, which means that they can access services provided by different sorts of computing paradigms without creating multiple accounts. Federation is also advantageous for the service providers, as they can rely on each other's strengths and reduce their weaknesses via leasing capacity from each other. A well-managed federation among bigger cloud service providers with smaller fog and edge computing provides more mobility and freedom.

However, the federation among multiple computing paradigms creates particular challenges, which include *user authentication*, *secure interfacing*, and *application mobility*, to name a few. A complete federation among multiple paradigms can only be realized once a proper authentication mechanism is put in place. Different computing paradigms use different authentication protocols, and the translation process among these protocols is complicated and expensive. This makes the authentication of a user the most critical challenge because if a user has to create an account on every different service provider, it will introduce the problem of handling multiple accounts and re-subscriptions, which will incur latency. A federation will allow a user to access different providers' services without creating a separate account on those computing platforms. Third-party authentication is the solution that can allow service providers to authenticate users without creating new accounts. A service provider with the user credentials will be referred to as the home service provider, and the one who wants to authenticate the user without the credentials will be referred to as the foreign service provider.

Another problem that arises here is the existence of different authentication protocols for multiple service providers like OpenID Connect (OIDC), 802.1x, and EPS-AKA. Each of these protocols has different authentication mechanisms; hence, the third-party authentication mechanism would need to translate between these protocols with the help of an entity. This will also raise security concerns, as the newly introduced entity needs to be secure enough so that service providers can trust that entity. In this research, we aim to answer the following vital questions: Can a foreign service provider authenticate a user without an account? If yes, how to translate among multiple authentication protocols used by multiple service providers, and how to keep this process transparent? How to keep the process secure so that service providers can trust it, and how much latency is introduced while doing that?

There are a few studies in the literature that propose federation among clouds (Villegas et al., 2012), or federation among cloud and edge, edge and fog (Ali et al., 2021a), fog and fog (Ali et al., 2021b), or provide novel authentication mechanisms for security. However, no study solves the third-party authentication problem in federated cloud, edge, and fog environments as a whole. We propose a universal federator (proxy) to provide third-party authentication among cloud, edge, and fog using different authentication protocols to solve the authentication, latency, and security issues. Our designed federator is standard-compliant and transparent. The federator does not focus on a couple of paradigms like cloud–edge, edge–cloud, or cloud–fog. However, it provides a universal translation among cloud, edge, and fog authentication protocols in all possible scenarios. Our designed federator comprises a federation controller and modules that serve as virtual counterparts to the authentication entities within the protocols involved. Our designed federator provides transparency by using these modules. The federator we have developed offers a means to simplify the costly and complex translation process while also achieving cost reduction. We also provide a backup in case the federator fails, as it is a single point of failure.

The results of our experiments conducted on a testbed, whose implementation details are published on GitHub (https://github.com/ericliujian/3rd-Party-Authenticaion), show that federator successfully provides third-party authentication among cloud, edge, and fog for OIDC, EPS-AKA and 802.1x while taking 4.07–51.8% of the authentication time that varies between 1.193–3.825s for 16 scenarios. The federator also provides multiple authentication options while providing translation, transparency, and security. The novel contributions of this work are summarized as follows:

- We propose a universal federator that provides federation among cloud, edge, and fog service providers using multiple protocols.
- Our designed federator enables users to conveniently access multiple service providers using a single account across various vendors.
- Our designed federator is standard-compliant, transparent, and provides multiple login options to the user so that it can select accordingly.
- We conduct a thorough security analysis of our proposed federator to demonstrate its robustness and ensure its security.

The rest of this paper is organized as follows. Section 2 introduces OIDC, EPS-AKA, and 802.1x authentication protocols in cloud, edge, and fog computing paradigms, along with related work. We provide our problem statement in Section 3. Section 4 introduces the universal federator along with its design and architecture. The implementation of the universal federator prototype, federator modules, and the testbed is presented in Section 5. Section 6 gives the results and evaluation, and Section 7 provides a discussion section. In Section 8, we conclude the paper and give some insight for future work.

2. Background

This section explores the cloud, edge, and fog computing background explaining the three protocols that we will use in this work, along with the related work and the threat model.

2.1. Computing paradigms applications

2.1.1. Cloud computing

Cloud computing provides the end users with computational, storage, and networking services. A traditional or IoT device can purchase a subscription for the cloud and use it for data processing, storage, or analyzing the data. The benefits of the cloud include cost-effectiveness, scalability, productivity, speed, and reliability. Cloud computing applications are widespread, such as data storage, testing, social networks, and backup, as this is the oldest computing paradigm in place. Many new startups also use cloud computing services for their storage and computation.

2.1.2. Edge computing

Edge computing stems from the European Telecommunication Standards Institute's (ETSI) proposal of integrating virtualization capabilities into the Mobile network operators (MNOs) within the RAN (Radio Access Network). Edge computing provides services such as computation and storage within the RAN. These services are similar to the cloud, but Edge provides these services with lesser computing power, lesser storage capacity, and reduced latency compared to the cloud. Edge computing is suitable for cellular network subscribers, as they can use computational services without subscribing to cloud or fog service providers. This eliminates the need for a new creation of subscriptions, which is one motivation for our federator.

2.1.3. Fog computing

Fog computing provides computational services like cloud computing, but it is decentralized as opposed to the cloud and closer to the end users. Fog can be considered a small cloud that provides cloud services closer to the users. Fog computing already exists in smart homes and smart city environments (Songhorabadi et al., 2023) and it differs from edge computing as it is not provided by MNOs. Fog computing is decentralized, and the data, computing, and storage are located between the devices that generate data and the cloud. Many applications use fog computing, such as augmented reality, multiplayer gaming, virtual reality, and multimedia streaming, as fog provides low latency, faster data transfer, and less bandwidth cost. Although each of these paradigms has pros and cons, a federation among them is highly beneficial for the upcoming applications requiring high storage and low latency. For example, data generated by different IoT devices need lots of storage space (as provided by the cloud) and low latency (as provided by edge and fog). Hence, there is a dire need for federation among these computing paradigms.

2.2. Authentication protocols in cloud, edge, and fog

In modern computing architectures such as cloud, edge, and fog layers, various authentication protocols are selectively employed by different vendors to enhance security and efficiency. OpenID Connect (OIDC) is commonly utilized in cloud and fog environments due to its robust identity management and federation capability. For edge computing, which often involves mobile networks, EPS-AKA is preferred for its strong integration with mobile network authentication systems, ensuring secure user verification. Additionally, 802.1x is applied in fog computing for its effective network access control, which is crucial for decentralized fog environments.

2.2.1. OIDC

OpenID Connect (OIDC) provides third-party authentication on top of OAuth 2.0. The OIDC allows the users to authenticate themselves with an application via another application. There are three key entities that are involved in the OIDC authentication. The first is a user, the second is a Relying Party (RP) and the third is an OpenID Provider (IdP or OP). When a user attempts to access an application (RP), the RP redirects the user to the IdP for authentication. The IdP then prompts the user to authenticate themselves, typically through a username and password or other authentication mechanisms. Once the authentication is successful, the IdP generates an identity token containing information about the user. The IdP then sends this token to the RP, which uses it to authorize the user and grants access to the requested resources. This exchange of authentication messages between the user, RP, and IdP enables users to access diverse applications with a single set of credentials. Data types exchanged between these entities include identity tokens containing user information such as user ID and scope. Many applications use OIDC, and the advantage of using OIDC is that the user does not need to create a new account every time they use a new application. There are options like login with Apple, Google, or Facebook. Therefore, the user can access an application via their existing account on another application, provided both applications are federated. This protocol is mainly used in cloud and fog.

2.2.2. EPS-AKA

Evolved Packet System Authentication and Key Agreement (EPS-AKA) is an authentication protocol that is used in cellular networks for the authentication of a user. The entities that collaborate seamlessly in the EPS-AKA authentication are a User Equipment (UE), an eNodeB (eNB), a Mobility Management Entity (MME), and a Home Subscriber Server (HSS). The authentication process is initiated when the UE sends a request message to the MME via eNB. The MME contacts the HSS located in the core network, which shares an authentication vector with the MME. The MME then forwards the necessary information to the UE for calculating the response and authenticates the UE once it receives the response from the UE. Data types exchanged between these entities include Authentication Vectors (AVs) that are generated by the HSS which contain information for the MME and the UE. This authentication protocol plays a pivotal role not only in ensuring the integrity of cellular network communications but also in supporting the deployment of edge computing services, crucial for the evolving landscape of cellular networks.

2.2.3. 802.1x

Another protocol that vendors can use is the 802.1x protocol. This is an important protocol that provides authentication to the devices that need to connect to other devices over a LAN or a WLAN. 802.1x protocol orchestrates interactions among three essential entities: a supplicant (end-user), an authenticator (access point that works as a bridge between the user and the LAN), and a RADIUS server, known as an authentication server. Each entity serves a distinct role in the authentication process. The supplicant presents its credentials to the authenticator to initiate the authentication process. The authenticator mediates access to the network resources based on the supplicant's credentials. However, the critical task of authenticating the supplicant lies with the RADIUS server, which verifies the supplicant's credentials against a centralized database. The data types exchanged include authentication requests and responses, containing user credentials and challenge tokens. The communication between these three entities is enabled by the Extensible Authentication Protocol (EAP) is used. By leveraging EAP, the 802.1x protocol ensures robust and standardized authentication procedures across diverse network environments. The deployment of this protocol is particularly prevalent in fog computing environments.

2.3. Related work

A few studies in the literature provide authentication and federation among different computing paradigms. We compare them with our work regarding whether providing third-party authentication, transparency, and multiple protocols support. It can be seen from Table 1 that several works have been done for cloud–cloud: Baran (2018), Megouache et al. (2020), Aruna et al. (2022) with the objective of federated authentication, security and data mitigation. Other works (Targali et al., 2013; Choyi and Brusilovsky, 2016; Han et al., 2019; Li et al., 2020; Edris et al., 2020) focus on edge–edge scenario. They tried to solve seamless, third-party, handover, anonymous and federated authentication problems. These studies propose a modification in the existing protocols or propose a solution approach that is not transparent, and they also do not consider multiple federation scenarios.

Several studies have explored extending different methods like OAuth 2.0 (Gibbons et al., 2014), unary-token (Deebak et al., 2020), blockchain (Zhang et al., 2021) and pre-authentication/post-authentication (Vinoth et al., 2022) to enhance federation between cloud–edge environments. These protocols form the foundation for solutions to improve security, seamless connectivity, privacy and reduce time. Others like Kahvazadeh et al. (2017), Tao et al. (2017), Zhang et al. (2023) are for cloud–fog scenarios, but all these studies do not consider the transparency of the proposed solution while only focusing on one or two federation scenarios.

Other studies in the literature have proposed federation among edge–fog with the objectives of WLAN-3GPP integration (Roeland and Rommer, 2014), medical emergency (Nakkar et al., 2020), enhancing security (Amanlou et al., 2021). For fog–fog scenario, (Santos et al., 2019; Tuli et al., 2019) use REST HTTP web services for application migration. Alharbi et al. (2017) focus on security and Ogundoyin and Kamil (2021) on peer-to-peer authentication. The studies in literature either propose a novel protocol or propose modifications to the existing protocols. None of these studies provide transparency and multiple protocol support while considering multiple federation scenarios among

Table 1
Related work.

Category	Name	Method	Objective	Third-party Authentication	Transparency	Multiple Protocols Support
Cloud–Cloud	Baran (2018)	Cloud Proxy	Federated Authentication	✓	X	X
	Megouache et al. (2020)	Asymmetric Encryption	Improve Security	X	X	X
	Aruna et al. (2022)	Self-sovereign Identity	Data Mitigation	X	X	X
Edge–Edge	Targali et al. (2013)	Generic Federated ID system	Seamless Authentication	X	X	X
	Choyi and Brusilovsky (2016)	Authentication Proxy	Third-party Authentication	✓	X	X
	Han et al. (2019)	Modified EAP-AKA	Handover Authentication	X	✓	X
	Li et al. (2020)	Identity based AKE	Anonymous Authentication	X	X	X
	Edris et al. (2020)	OAuth2.0	Federated Authentication	✓	X	X
Cloud–Edge	Gibbons et al. (2014)	OAuth2.0	Security	X	X	X
	Deebak et al. (2020)	Unary-token	Seamless Connectivity	X	X	X
	Zhang et al. (2021)	Blockchain	Privacy	X	X	X
	Vinoth et al. (2022)	Pre-authentication	Reduce Time and Storage	X	X	X
Cloud–Fog	Kahvazadeh et al. (2017)	SDN Master/Slave Control	Security	X	X	X
	Tao et al. (2017)	Cross-layer design	V2G service provision	X	X	X
	Zhang et al. (2023)	Role-based Trust Evaluation	User Authentication	✓	X	X
Edge–Fog	Roeland and Rommer (2014)	GTP Tunneling	WLAN-3GPP integration	✓	X	X
	Nakkar et al. (2020)	Broadcast Protocol	Medical Emergency	X	X	X
	Amanlou et al. (2021)	Semi-centralized Key	Enhance Security	X	X	X
Fog–Fog	Santos et al. (2019) Tuli et al. (2019)	REST HTTP web service	Application Migration	X	X	X
	Alharbi et al. (2017)	Challenge–response Auth.	Security	X	X	X
	Ogundoyin and Kamil (2021)	AKA Protocol	Peer-to-peer Authentication	X	X	X
Cloud–Edge–Fog (Ours)	Federated Authentication	Proxy Based Approach	Third-party Authentication	✓	✓	✓

cloud, edge, and fog. Our proposed universal federator is original because although it does not propose a novel protocol or try to modify existing protocols, it glues the existing protocols together through a novel federator. Also, the proposed federator is not limited to a 1–1 mapping of providers or protocols, but rather provides multiple options and considers 16 scenarios among cloud, edge, and fog while providing multiple protocols support and transparency. We did not find any similar solution in related works, which makes our design novel.

2.4. Threat model

In order to simplify the problem at hand, we make certain assumptions. Specifically, we assume that the cloud, edge, and fog service providers involved in the system are diligently maintained and secured by their respective service providers and provisioning of their security is out of the scope of this work. These assumptions allow us to focus our analysis on other aspects of our designed federation system without considering potential vulnerabilities within those service providers. Therefore, the proposed federator is the vulnerable point in all 16 scenarios. The communication between the service providers and the federator is also considered to be vulnerable, and the federator mapping table can be read by an attacker. The attackers can also play replay attacks, MITM attacks, and data leakage attacks on the federator.

2.5. Security requirements

In light of the threats introduced in the previous subsection, some security requirements must be met to ensure system security. The basic security requirements are confidentiality and integrity for the communication between involved entities, i.e., user, home service provider, federator, and foreign service provider. We also incorporate security requirements, such as replay resistance, data manipulation and data leakage resistance, impersonation resistance, and Man in the Middle (MITM) attack resistance.

2.5.1. Confidentiality

Confidentiality is the security property that limits access to information. In our case, the flow of information between the user and the home and foreign service providers, as well as the user and the federator, must be confidential. Also, the confidentiality of information between different service providers must be ensured.

2.5.2. Integrity

This security property ensures that the information is accurate and trustworthy. In our case, the integrity of the information flowing between the involved parties must be ensured.

2.5.3. Replay resistance

This security property ensures that if an attacker captures a packet, it should not be able to forward it later to threaten security. In our proposed scheme, we must ensure that the home and foreign service providers detect and discard replayed messages.

2.5.4. MITM resistance

Man in the Middle attacks are usually carried out via active eavesdropping, where a malicious user can monitor the communication between legal users. Our proposed schemes must ensure that the malicious users cannot modify the exchanged messages and send them to the federator later.

2.5.5. Data manipulation resistance

Data manipulation is an attack where a malicious user can alter or modify critical and essential data, which might be the authentication token in our case. Such data manipulation is harmful to the service providers. Therefore, our system should be able to resist such attacks.

2.5.6. Data leakage resistance

Data leakage is unauthorized data transmission from an entity to outside a trusted domain. In our case, the data stored inside the federator can be leaked to the attackers. The security of that data must be ensured.

2.5.7. Impersonation resistance

An impersonation attack is an attack in which the attackers pose as a known or trusted person or entity. In our case, the attackers can pose as the trusted federator and can cause harm to the service providers.

Table 2
All possible Scenarios.

Home \ Foreign	Cloud	Edge	Fog	Fog-802.1x
Cloud	Cloud–Cloud	Cloud–Edge	Cloud–Fog	Cloud–Fog 802.1x
Edge	Edge–Cloud	Edge–Edge	Edge–Fog	Edge–Fog 802.1x
Fog	Fog–Cloud	Fog–Edge	Fog–Fog	Fog–Fog 802.1x
Fog-802.1x	Fog-802.1x-Cloud	Fog-802.1x-Edge	Fog-802.1x-Fog	Fog-802.1x-Fog-802.1x

3. Problem statement

We assume there are different computing paradigms and a user has an account on one of these, which can be a cloud, edge, or fog. The home service provider is the one which already provides services to the user and the user has an account there. We assume that the user moves from place to place, gets out of the range of the home service provider, and needs to connect to another service provider. We call this the foreign service provider, which has to provide a service to the user. In the case of a user that keeps its place, we assume that the user has different applications with different requirements and needs services from different service providers. The foreign service provider must be able to provide services to the user without creating a new account. Hence, there is a need for communication between the home and foreign service providers. These service providers may or may not be using the same protocols, as they may or may not belong to the same computing paradigm. Therefore, there is a need for a mechanism that provides third-party authentication for a user, among foreign and home service providers, with low latency while keeping the process transparent, standard compliant, and secure. The problem statement is as follows:

Given: Multiple computing paradigms using multiple protocols such as EPS-AKA, OIDC, and 802.1x.

Objective: To provide third-party authentication to the users of different computing paradigms.

Constraints: Transparency, Protocol translation and Security.

Assumptions: The need for one account service everywhere is a must, and service providers will come together to increase their capability and capacity.

The home and foreign service providers may be cloud, edge, and fog using different protocols. OIDC is one of the most established protocols in the cloud computing industry (Naik and Jenkins, 2016) that enables users to authenticate and authorize themselves to access APIs, protected data, and other resources on different websites. It provides identity federation and single sign-on (SSO) capabilities, which are ideal for cloud services that need to manage numerous user identities across various platforms. Therefore, we assume that the cloud will use OIDC protocol. We assume that the edge will use EPS-AKA as it is used primarily in cellular LTE networks (Choudhruy, 2012), leveraging SIM card-based credentials, which makes it well-suited for edge devices that often include mobile or cellular technologies. Due to the widespread adoption of OIDC in fog computing (Navas and Beltrán, 2019) and usage of 802.1x in fog computing for port-based Media Access Control (OpenFog Consortium et al., 2017), we assume that the fog will use OIDC and 802.1x.

Therefore, 16 scenarios are possible as we consider three different computing paradigms with different protocols, which can either be on the home or the foreign side, as shown in Table 2. For instance, one out of 16 possible scenarios is the Edge–Fog scenario, where an edge is the home service provider using EPS-AKA, and fog is the foreign service provider using OIDC. The fog service provider, using the OIDC authentication protocol, contains the service to be accessed by the user. The edge service provider, using EPS-AKA, is the home service provider with the user's credentials. The objective here is to allow a user, a subscriber of the edge, to access foreign fog's services. In the bigger picture, a cloud can be a home service provider, with four possible combinations cloud, edge, fog, and fog using 802.1x at the foreign sides. Similarly, edge, fog, and fog-802.1x can also be the home service providers. Hence, 16 scenarios are possible in this federation.

Certain issues need to be solved to realize these 16 scenarios: (1) Third-party authentication for a user that needs to access the services of different providers using its home credentials and (2) Intermediate communication: an entity that resides between the services providers and provides a way for communication between them. The intermediary must also consider the authentication protocols used in cloud, edge, and fog, like OIDC, 802.1x, and EPS-AKA. The intermediary must be designed in accordance with the authentication components, such as the user, the relying party (RP), the OpenID provider (IdP or OP), the supplicant, the authenticator, the RADIUS server, the UE, the MME, and the HSS. The message flows of EPS-AKA, OIDC, and 802.1x are different from each other, which causes a mismatch for third-party authentication. Therefore, there is a need for an appropriate entity that provides mapping of messages between these protocols.

4. Proposed universal federator

To solve the problem identified and described in the previous section, we propose a universal federator as the communication intermediary. The federator supports multiple authentication protocols used by cloud, edge, and fog. The proposed federator sits between the cloud, edge, and fog service providers and can be implemented in the real network infrastructure in the following ways:

- Federation as a Service (FaaS): The federator can be deployed by one of the major cloud providers that are involved in the federation. They can offer it as a service, such as "Federation as a Service" (FaaS).
- Cellular Network VNF: The federator can also be deployed in one of the cellular networks involved in the federation. The federation unit can be implemented as a virtual network function (VNF) that will facilitate scalability.
- Third-party Vendor: The proposed federation unit can also be deployed by a third-party vendor, which can act as a neutral intermediary among service providers.

The primary design idea of the federator is transparency. Transparency means no change in the existing infrastructure or protocols. There could be three possible ways to address the problem identified and described in the previous section: (1) propose a novel protocol, (2) modify existing protocols. (3) build a solution on existing protocols. We chose the third option as it is the most suitable for the service providers and maintains transparency. The federator's transparency is necessary to avoid any modifications in the existing infrastructures and authentication protocols used by the cloud, edge, and fog. The federator's transparency is achieved with the help of a controller and several modules that play the virtual counterparts' roles in communicating with the entities involved in the authentication protocols. The primary design ideas of the federator are:

- The universal federator is kept transparent so that there is no modification required in the protocols and infrastructure of the service providers.
- The federator has virtual counterparts, according to the involved authentication protocols, so that it could play different roles, depending upon the involved entities in the authentication protocols.
- There is pairwise statefulness in the federator so that it can handle multiple third-party authentications.

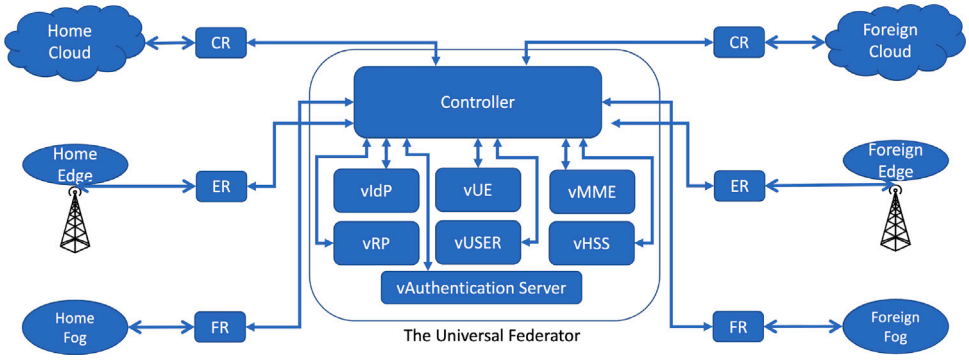


Fig. 1. The universal federator architecture.

Table 3
Federator roles against corresponding entities.

Proxy Role	Involved Protocol	Corresponding Entity	Paradigm
vHSS	EPS-AKA	vMME or foreign MME	Edge
vMME	EPS-AKA	vHSS or home HSS	Edge
Identity Provider	OIDC	Relying Party in Cloud or Fog	Cloud, Fog
Relying Party	OIDC	Identity Provider in Cloud or Fog	Cloud, Fog
vUser (Client)	OIDC	User Info endpoint in Cloud or Fog	Cloud, Fog
vUE	EPS-AKA	MME of home Edge	Edge
vAuthentication Server	802.1x	Foreign Fog (Authenticator)	Fog

Table 4
Detailed roles for each scenario.

Home \ Foreign	Cloud	Edge	Fog	Fog-802.1x
Cloud	vIdP / vRP	vHSS / vUser	vIdP / vRP	vAS / vUser
Edge	vIdP / vUE	vHSS / vMME	vIdP / vUE	vAS / vUE
Fog	vIdP / vUser, vRP	vHSS / vUser	vIdP / vUser, vRP	vAS / vUser
Fog-802.1x	vIdP / vUser	vHSS / vUser	vIdP / vUser	vAS / vUser

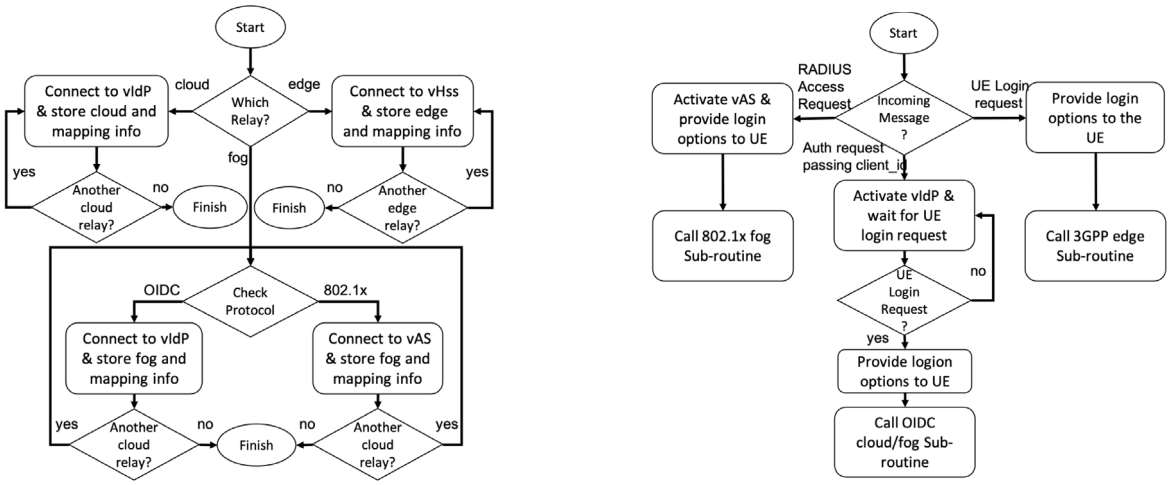
4.1. Architecture

Our universal federator serves as a central intermediary for cloud, edge, and fog layers, streamlining communications between these platforms. It employs different virtual modules that replicate functions of key authentication protocols like OIDC, EPS-AKA, and 802.1x and manage interactions between home and foreign providers. Our federator supports various user credentials depending on the authentication protocol used. For OIDC, it accepts ID Tokens, User IDs, and Passwords, facilitating secure and federated identity management. In the context of 802.1x, it uses User IDs and Passwords to ensure network access control. The federator handles ID Tokens and SIM Credentials for EPS-AKA, leveraging mobile network authentication methods to secure edge devices. This multi-protocol support enables the federator to provide flexible and robust authentication across cloud, edge, and fog environments.

In OIDC, user-related claims are communicated between the OpenID Provider (OP) and the Relying Party (RP). In 3GPP EPS-AKA, claims involve user identity and authentication details like the IMSI, authentication vectors, RAND, and AUTN to secure communications. In 802.1x, the supplicant and RADIUS protocol facilitate claim translation. Our authentication proxy is adept at managing and translating claims for these protocols, ensuring robust and seamless authentication across different systems.

The universal federator sits between the cloud, edge, and fog service providers, as shown in Fig. 1. The federator is designed while considering the entities involved in OIDC, EPS-AKA, and 802.1x. The modules inside the federator are selected to communicate transparently with the service providers on home and foreign sides using different authentication protocols. Hence, the federator contains the virtual user (vUser), the virtual relying party (vRP), the virtual User Equipment (vUE), and the virtual Mobility Management Entity (vMME) to communicate with the home service providers. The virtual OpenID provider (vIdP), the virtual authentication server (vAS), and the virtual Home Subscriber Server (vHSS) to communicate with the foreign service providers. The federator also consists of a controller that controls the modules and provides a way for communication between these modules and service providers. The federator is connected to cloud, edge, and fog via cloud relay, edge relay, and fog relay.

The modules deployed inside the federator are the virtual entities of the OIDC, 802.1x, and EPS-AKA authentication protocols. Depending upon the authentication protocols at the home and foreign side, the federator activates different components, as shown in Table 3. The controller selects and coordinates between these virtual components. A detailed breakdown of the roles played by the federator in all 16 scenarios is shown in Table 4. The proposed federator not only needs to provide transparent third-party authentication but also needs to be secure. Therefore, the federator makes use of PKI for key management, generates unique session keys for key freshness and secrecy, uses nonce for replay provision stores the information in an encrypted manner. The details of security provisions are provided in Section 7.3. In the following subsection, we will describe the logic flow for the federator in detail.



(a) The initialization phase (b) The operational phase

Fig. 2. Federator initialization and operation.

4.2. Logic flow

4.2.1. Initialization phase

As shown in Fig. 2(a), the initialization phase connects the service providers with the universal federator. The initialization phase starts by receiving the connection requests coming from the relays. The federator needs to check which relay it is coming from. The next step is to check the incoming message to find the source. The source information is matched with the stored information about the relays, and the output is either cloud, edge, or fog relay. If the output is the cloud relay, the federator needs to perform specific actions which include; (i) storing the cloud information, (ii) providing cloud connection to the vIdP module, and (iii) storing the related information for mapping purposes.

Once the information is stored, the federator waits for other connection requests. If the federator receives the request as input, it checks if it is from a cloud relay and if it is from a cloud relay, the federator repeats the process; otherwise, the process is finished. If the output is an edge relay, the federator needs to perform specific actions; (i) store the edge information, (ii) provide cloud connection to the vHSS and vMME module, and (iii) store the related information for mapping purposes. Once the information is stored, the federator waits for other connection requests. If the federator receives the request as input, it checks if it is from an edge relay and if it is from an edge relay, the federator repeats the process; otherwise, the process is finished.

The federator needs to check the protocol to see if the output is a fog relay. The incoming request is matched with the existing supported protocols. If it matches with the OIDC protocol, the federator needs to perform certain actions; (i) store the fog information, (ii) provide fog connection to the vIdP module, and (iii) store the related information for mapping purposes. Once the information is stored, the federator waits for other connection requests. If the federator receives the request as input, it checks if it is from a fog relay, and if yes, it repeats the process. Otherwise, the process is finished. If the output is 802.1x protocol, the federator needs to perform certain actions; (i) store the fog information, (ii) provide fog connection to the vAS module, and (iii) store the related information for mapping purposes. The initialization phase creates a lookup table, which does not need to be updated frequently, as it will only be updated when a new service provider joins the federation.

4.2.2. Operational phase

Once the service providers have connected with the federator through the initialization phase, the operational phase starts, as shown in Fig. 2(b). The federator receives an incoming message from one of the relays. The message is fed as input to the federator, and the federator needs to respond differently, depending upon different messages. The federator checks the incoming message from the relay and matches it with the existing entries for possible matches. If the incoming message matches the RADIUS Access request, the federator needs to perform certain actions; (i) activate the vAS to receive the request and (ii) provide login options to the UE. The UE response is fed to the subroutine "User Auth for 802.1x".

On the other hand, if the incoming message matches the UE login request, the federator provides login options to the UE, and the response is fed to the subroutine: "User Auth for 3GPP Edge". If the incoming message matches with the Auth Request passing client_id, the federator activates the vIdP to receive the message and puts it into the wait state. The federator waits for the UE login request, and once the UE sends the login request, the federator provides login options to the UE and feeds the UE response to the subroutine: "User Auth for Cloud/Fog OIDC". If no response is received while the federator is in this step, the process is halted. The mentioned subroutines are how users are authenticated with foreign service providers using home credentials.

4.3. Forwarding table

The federator builds the forwarding Table 5 based on the federator logic flow, as it needs to decide which components need to be activated and what mappings must be done for a particular scenario. The federator logic flow starts by examining the traffic coming from the relays. The federator needs to find out which is the home relay and which is the foreign relay. If the home relay is fog, the controller needs to find out whether the fog protocol is OIDC. If the home relay is fog and the fog protocol is not OIDC, vUser is activated on the home side of the federator. If the home relay is cloud, vUser, and vRP are activated on the home side of the federator, depending on the foreign relay.

If the home relay is edge, vMME, and vUE are activated on the home side of the federator, depending upon the foreign relay. The federator controller also needs to find out the foreign relay and if the foreign relay is an edge relay, vHSS is activated on the foreign side of the federator. If the foreign relay is a cloud relay, vIdP is activated on the foreign side. If the foreign relay is a fog relay, the federator controller needs to find out whether OIDC is used in the fog. If OIDC is used, vIdP is activated, and if not, vAS is activated on the foreign side of the federator. In the case of the edge–cloud scenario, the universal federator will select the vUE as the module on the home side and vIdP as the module on the foreign side. The federator will need to map the 3GPP EPS-AKA and OIDC protocols by translating the "Login with

Table 5
The Forwarding Table.

Home	Destination	Home Module	Foreign Module	Mapping from	Mapping to
Cloud	Cloud	vRP	vIdP	Auth Request passing client_id from SP	Auth request passing client_id as RP
	Edge	vUser	vHSS	Auth Request (IMSI)	Login with IMSI
	Fog-OIDC	vRP	vIdP	Auth Request passing client_id from SP	Auth request passing client_id as RP
	Fog-802.1x	vUser	vAS	RADIUS Access Request	Login Request
Edge	Cloud	vUE	vIdP	Login with IMSI	Auth Request (IMSI)
	Edge	vMME	vHSS	Auth Request (IMSI)	Auth Request (IMSI) as MME
	Fog-OIDC	vUE	vIdP	Login with IMSI	Auth Request (IMSI)
	Fog-802.1x	vUE	vAS	Login with IMSI	Auth Request (IMSI)
Fog-OIDC	Cloud	vUser+RP	vIdP	Auth Request passing client_id from SP	Auth request passing client_id as RP
	Edge	vUser	vHSS	Auth Request (IMSI)	Login with IMSI
	Fog-OIDC	vUser+RP	vIdP	Auth Request passing client_id from SP	Login + Auth Request passing client ID as RP
	Fog-802.1x	vUser	vAS	RADIUS Access Request	Login Request
Fog-802.1x	Cloud	vUser	vIdP	Auth Request passing client_id from SP	EAPOL start request
	Edge	vUser	vHSS	Auth Request (IMSI)	EAPOL start request
	Fog-OIDC	vUser	vIdP	Auth Request passing client_id from SP	EAPOL start request
	Fog-802.1x	vUser	vAS	RADIUS Access Request	EAPOL start request

IMSI" message to "Auth Request (IMSI)" to communicate with the edge. The same procedure will be followed according to Table 5 for the rest of the scenarios.

4.4. Message flow diagrams

This section explains the message flow for the most probable scenario out of 16 scenarios. We choose one of the 16 scenarios to explain the message flow. Depending upon the scenario, the other 15 message flows follow the same pattern but with different protocols and different federator components.

4.4.1. Edge–cloud scenario

Fig. 3 illustrates the universal federator message flow for the scenario where the home provider is an edge and the foreign provider is a cloud (Edge–Cloud scenario). The message flow for the Edge–Cloud scenario consists of three stages: request stage, response stage, and confirmation stage.

Stage 1—Request Stage: The UE initially sends the service request to the service provider (foreign cloud). The service provider redirects the UE to the federator controller via the cloud relay, as the UE is not the cloud's subscriber. The service provider also initiates the OIDC message "Auth Request passing client_id" to the virtual IdP component of the federator via the federator controller and the cloud relay. The federator controller provides identity service to the UE with multiple options. Here, the UE chooses the authentication service provided by a mobile network provider. The UE's IMSI stored in the USIM is sent as a user ID to the federator controller. The federator controller, after receiving the "Auth Request passing client_id" from the foreign cloud and "Login via Edge + IMSI" from the UE, activates the virtual IdP and the virtual UE components of the federator. The federator controller does the necessary mapping according to Table 5 and passes the mapped message to the virtual UE component, which forwards the IMSI to the home MME via the federator controller and the edge relay. The MME processes the message and sends the processed message to the HSS.

Stage 2—Response Stage: The HSS receives the authentication request, calculates the authentication response, and sends it back to the MME, which keeps the XRES and forwards the challenge to the virtual UE component in the universal federator via the edge relay and the federator controller. The federator controller then forwards the challenge to the original UE via the cloud relay. The UE uses the key, calculates the RES, and sends it to the virtual UE via the cloud relay and the federator controller. Consequently, the virtual UE forwards the RES, via the federator controller and the edge relay, to the MME to authenticate the UE.

Stage 3—Confirmation Stage: Once the MME checks that the RES equals the XRES, it sends a 200 OK message to the vUE via edge relay and the federator controller. The federator controller then instructs the vIdP to authenticate the end user and sends ID_token to the foreign cloud (RP). Then the cloud service provider authenticates the user by validating the authentication token. The cloud service provider permits the user to access its services upon token validation.

5. Implementation

In order to provide a comprehensive solution to the authentication problem in multiple scenarios discussed previously, we proposed to design a universal federator that can federate different service providers and support multiple protocols. Our proposed federator is transparent and accommodates the existing protocol structures with the help of other virtual counterparts inside it. These virtual counterparts demonstrate different roles while communicating with the service providers, which include: IdP, UE, HSS, RP, user, AAA, and supplicant. Our federator also provides pairing between the virtual counterparts to make the federator fit for both the home and foreign sides. In this section, we talk about the implementation details of our proposed federator.

5.1. Prototype architecture

Our federator's design aligns with the architectural principles of OIDC, EPS-AKA, and 802.1x authentication protocols, encompassing a central controller and several modular components. These modules comprise a virtual Identity Provider (vIdP), virtual User Equipment (vUE), virtual Home Subscriber Server (vHSS), virtual Mobility Management Entity (vMME), virtual Relying Party (vRP), virtual User (vUser or vSupplicant) and virtual Authentication Server (vAAA or vAS). This selection ensures that our federator integrates seamlessly with established authentication protocols, facilitating transparent communication with the necessary entities. The federator dynamically orchestrates these modules, selecting the appropriate components for each authentication scenario. For instance, when fog-OIDC acts on the home side, it anticipates authentication requests from an RP and communication with the user for authentication purposes. In such cases, the federator will employ a vUser and vRP to engage with the home vIdP. In foreign Edge scenarios, the vHSS communicates with the vUser/vMME. The vIdP facilitates authentication in cloud or fog-OIDC environments abroad. For foreign Fog-802.1x scenarios, the federator integrates with a RADIUS server (vAAA). Additionally, for edge computing scenarios, we utilize the Open Air Interface (OAI) to implement 4G network elements, including User Equipment (UE) and Edge components (HSS, MME, and eNB), ensuring comprehensive coverage and versatility across diverse authentication contexts.

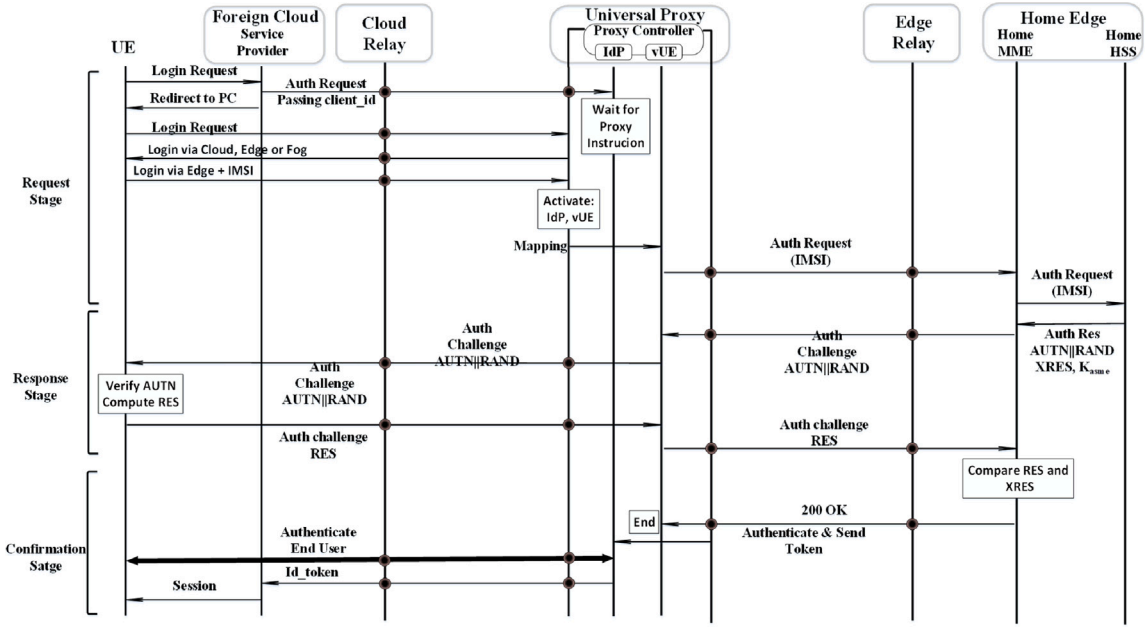


Fig. 3. Message flow for edge-to-cloud.

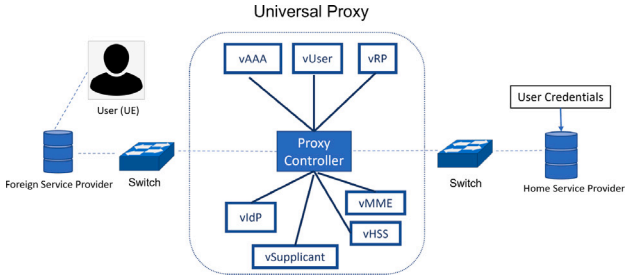


Fig. 4. Experiment testbed.

5.2. Experiment testbed

Our testing environment was set up using four personal computers (PCs) and two Cisco 2960X switches, as depicted in the referenced in Fig. 4. One switch facilitated operations on the home side, while the other supported activities on the foreign side. The PCs were equipped with Intel Core i7-10700F CPUs at 2.90 GHz and 32 GB of RAM and ran Ubuntu 16.04 OS. The configuration was as follows:

- The first PC functioned as the foreign service provider, hosting the foreign 802.1x authentication server (AAA) and the foreign OpenID Connect Relying Party (OIDC RP). This machine also supported the 3GPP network components for edge computing through the Open Air Interface (OAI).
- The second PC was the home service provider, incorporating the home 802.1x AAA and the home OIDC Identity Provider (IdP).
- The third PC was dedicated to the 802.1x supplicant and the OIDC user, simulating client-side operations.
- The fourth PC was designated for the universal federator, central to our 16 federation scenarios. It functioned as an intermediary, orchestrating communication and message transfer between the home and foreign entities.

This setup allowed us to comprehensively evaluate the federator's performance across diverse federation scenarios, ensuring robust testing of its capabilities in a controlled environment.

The open-source authentication packages we used in our federator are as follows: OIDC was deployed by using two different open-source OIDC implementations. We used AuthLib and Django OIDC Provider to deploy the OIDC components (RP, IdP, and User). The virtual supplicant and 802.1x supplicant were implemented with the WPA supplicant daemon so that we could deploy the users for 802.1x. The Authentication servers for 802.1x were deployed with FreeRADIUS, which is the most popular and widely deployed RADIUS server. Edge components were deployed by using the OpenAirInterface (OAI) cellular network. The user equipment(UE) in edge scenarios was implemented in Python and C. We have also uploaded our testbed implementation on GitHub (https://github.com/ericliujian/3rd-Party-Authenticaion).

6. Results and evaluation

Initially, our evaluation focuses on assessing authentication latency, wherein we document the time required for third-party authentication across 16 distinct scenarios. This involves comparing the authentication durations to discern patterns and underlying causes for the observed variances. Subsequently, we dissect the latency contributions for each scenario, categorizing them by the user, service provider, and the federator itself. Further, we conduct a comprehensive bottleneck analysis for these scenarios to determine whether the federator introduces any performance constraints. This step includes examining how authentication times differ when using federator-based authentication versus direct authentication methods, providing insights into the federator's efficiency. Lastly, we assess the federator's scalability by incrementally increasing network traffic. This experiment is designed to understand how heightened demand affects authentication latency, offering a clear picture of the federator's performance under varying loads. This multi-faceted approach allows us to thoroughly evaluate the federator's capabilities and identify areas for optimization.

6.1. Authentication latency analysis

6.1.1. Authentication time for all scenarios

Fig. 5 presents the authentication duration across 16 different scenarios, revealing that the Fog 802.1x-Fog 802.1x setup incurs the longest authentication time at (AT_{max} = 3.83 s) seconds, while the Cloud–Cloud configuration is the quickest at (AT_{min} = 1.19 s) seconds.

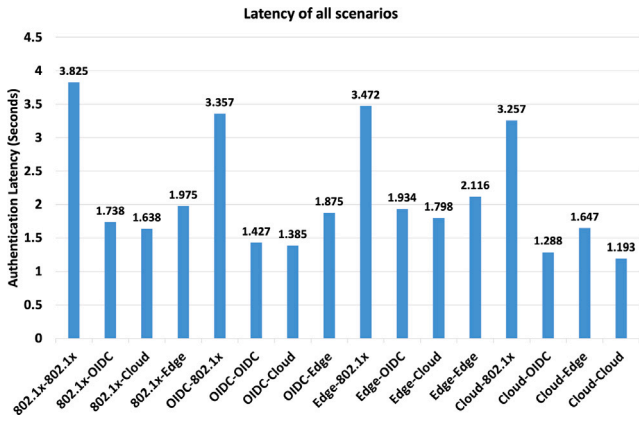


Fig. 5. Latency for all scenarios.

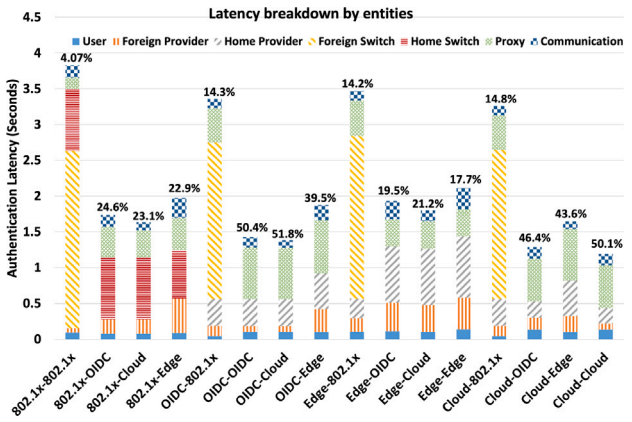


Fig. 6. Latency breakdown by entities.

On average, the authentication time across all scenarios is 2.12 s (AT_{avg}). The extended duration in the Fog 802.1x-Fog 802.1x scenario is attributed to the necessity of multiple Extensible Authentication Protocol (EAP) authentications. Specifically, when 802.1x authentication occurs on the home side, the switch must activate the connection port and execute a series of pre-configured setups, including the application of Access Control Lists (ACLs). On the foreign side, the process involves two stages of 802.1x authentication—initially granting restricted access and directing to a captive portal, followed by a second authentication that provides full network access. This results in three authentication steps for the Fog 802.1x-Fog 802.1x scenario, explaining its longer duration than others. Conversely, the Cloud–Cloud scenario achieves the shortest time by merely initiating the virtual identity provider and relying party, which exchange client_ID and ID_token for activation.

6.1.2. Latency breakdown by entities

In our analysis, we deconstructed the authentication process facilitated by the federator into its constituent entities to identify which component accounts for the most significant fraction of the overall authentication time. Illustrated in Fig. 6, the data reveals that the foreign switch is responsible for most of the authentication duration, represented by the yellow stacked column in the graph. Our calculations found that the federator's share of the total authentication time varies from 4.07% to 51.8%, equating to a time range of 0.156 s to 0.74 s. Notably, the federator's impact on the authentication timeline is more pronounced in scenarios involving the OpenID Connect (OIDC) protocol, constituting 13.5% to 51.8% of the authentication time. This increase is attributed to the OIDC protocol's inherent need for multiple redirections between the virtual Relying Party (vRP) and virtual User (vUser), which inherently extends the authentication process.

6.1.3. Latency breakdown by protocols

In our subsequent analysis, we segmented the authentication duration according to the specific protocols engaged. Fig. 7 shows that the 802.1x protocol accounted for a significantly longer portion of the authentication time compared to other protocols. This extended duration is primarily due to the 802.1x protocol's requirement for an additional RADIUS server, which serves as a security gatekeeper for the network. Upon a user's attempt to connect to the network, the RADIUS server authenticates the user and grants the necessary authorizations. Additionally, when edge computing elements are incorporated, the delay in edge scenarios can often be traced back to the User Equipment (UE). The UE must establish a connection with Mobile Edge Computing (MEC) components to gain internet access, which can introduce a bottleneck, particularly regarding network access and data transmission efficiency.

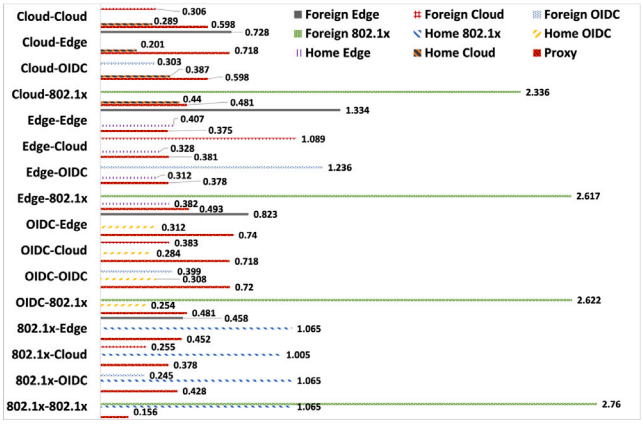


Fig. 7. Latency breakdown by protocols.

6.1.4. Bottleneck analysis

Fig. 6 reveals that in scenarios involving 802.1x, the federator contributes to only 4.07–24.6% of the total authentication time. The primary bottleneck in these cases is identified as the Cisco switch on both the home and foreign sides. The delay is attributed to the switch's implementation of the Spanning Tree Protocol (STP), which is responsible for establishing a new network path for the newly connected device. This process occurs before issuing the final Extensible Authentication Protocol (EAP) success message. Such findings underscore that the federator is not the bottleneck's source. Instead, our analysis points to the network equipment, specifically the switch's handling of new connections through STP, as the cause of the delay and this delay is inevitable.

6.1.5. Authentication with vs. without proposed federator

Fig. 8 compares authentication times, highlighting the differences between utilizing our proposed federator and a scenario without it. Without the federator in place, users are required to maintain multiple accounts across various service providers to achieve authentication, necessitating separate accounts for both home and foreign services. Consequently, this approach compels users to manage dual accounts and undergo two authentication processes. While it is true that introducing our universal federator increases the overall time required for the authentication process – attributable to the time the federator controller takes to initiate modules and facilitate communication among them – the incremental time delay is deemed reasonable. The primary benefit of employing the federator, which is eliminating the need for a secondary account at the foreign service provider, significantly outweighs the slightly increasing authentication time, offering a more streamlined and efficient user experience.

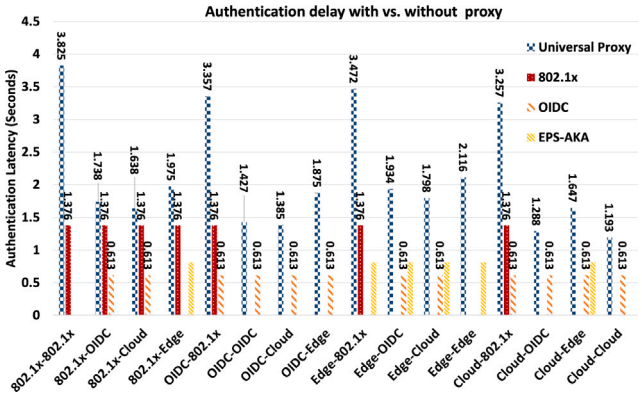


Fig. 8. Authentication time with vs. without our federator.

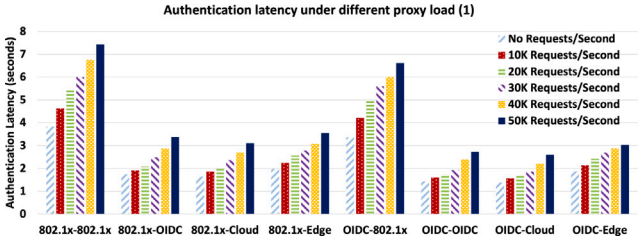


Fig. 9. Authentication time under different federator load (1).

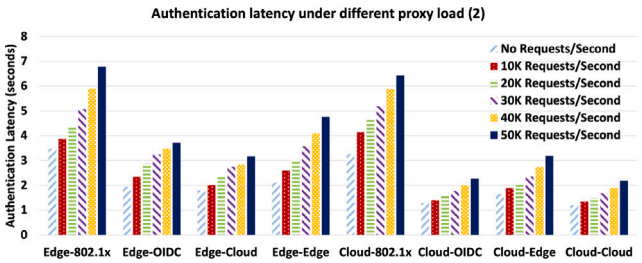


Fig. 10. Authentication time under different federator load (2).

6.2. Scalability testing

Beyond analyzing authentication latency, we delved into assessing the scalability of our proposed federator, precisely its capacity to efficiently handle an increasing volume of user requests. This scalability evaluation involved modifying the traffic load directed towards the federator, characterized by numerous connection requests per second. By incrementally raising the traffic load, we aimed to observe and document any corresponding changes in authentication latency. This approach allowed us to gauge the federator's robustness and adaptability under varied operational demands, providing insights into its performance under scenarios of heightened user activity.

To assess how our federator copes with varying traffic volumes, we developed a script designed to simulate a range of request loads, from 10,000 to 50,000 requests per second. It is important to clarify that this traffic pertains to the control plane, not the data plane. Hence, we quantify the load in terms of authentication requests per second. This metric reflects the federator's capacity to accommodate a rising number of new foreign user requests each second. We escalated the load from zero to 50,000 requests per second and, as depicted in our findings in Figs. 9 and 10, observed that the authentication time increased by 61%–120% in response to a 500% increase in traffic. This outcome convincingly demonstrates the federator's robustness, showing its ability to manage significantly higher levels of network traffic with a relatively modest increase in latency.

7. Discussion

7.1. Federator backup analysis

The federator serves as a central authentication system, providing third-party authentication across cloud, edge, and fog layers. Despite its robust design that ensures transparency and security, the federator poses a potential risk as a single point of failure within the federated network. If the federator fails, it could disrupt access for subscribers attempting to use services across different computing domains. In this subsection, we would like to introduce monitoring tools and high availability (HA) to solve a particular single failure risk.

Our federator system enhances its operational reliability by utilizing continuous monitoring, which tracks the system's performance in real-time through tools like Prometheus or DataDog. This ongoing surveillance allows the system to immediately identify and diagnose any operational glitches or inefficiencies. By detecting these issues early, the system can proactively address them before they escalate into more significant problems. This preemptive approach not only ensures smoother operation but also minimizes downtime, thereby enhancing overall system reliability and user satisfaction.

Apart from that, High availability (HA) can solve the problem of a single point of failure. Common HA strategies, such as redundancy, fail-over, fail-back, heartbeat, master–slave (Singh et al., 2012), can be applied to the federator to solve the problem of the single point of failure. In order to support HA, we could use redundancy and propose a backup federator, which acts as the secondary federator and sits-and-listens to the message exchange between the original federator and service providers. The messages between the original federator and the service providers are of two types; incoming and outgoing. The three types of messages incoming from the foreign service providers into the original federator are the "Auth Request passing client_ID", "UE Login Request/Auth Request", and "RADIUS Access request". The three types of messages outgoing from the original federator in response to the incoming messages, are "ID_token", "Auth Response", and "RADIUS Access request". The backup federator works in a promiscuous mode and listens to the incoming and outgoing messages exchanged between the original federator and service providers. Suppose the backup federator receives the incoming message and does not receive the corresponding outgoing message within $2 * AT_{avg}$ seconds. In that case, it assumes that the original federator has failed, takes the role of the original federator, and handles all the incoming requests. Once the original federator is fixed, it sends an "active_status" message to the backup federator. The backup federator sends an "ACK" to the original federator, handles the requests at hand, and goes back to the promiscuous mode. After receiving the "ACK", the original federator starts to handle all the incoming requests.

7.2. Cost analysis

Implementing our proposed third-party federator indeed introduces an initial increase in implementation costs for service providers. However, adopting such a federator can be a strategic investment for companies focused on maximizing profitability in the long run. The essence of a profitable service solution lies in its ability to cater to a broader user base while minimizing operational costs. The Internet of Things (IoT) is an excellent example of how this can benefit stakeholders across cloud, fog, and edge computing domains due to the vast amounts of data generated (Mejtoft, 2011). The raw data collected by IoT devices does not need to be immediately transferred to cloud environments for long-term storage or intensive processing (Bittencourt et al., 2017). By leveraging edge and fog computing services, service providers can significantly reduce their reliance on cloud resources,

leading to cost savings and more efficient use of resources. This strategy not only optimizes the handling of IoT data but also aligns with the broader objective of delivering high-quality services to as many users as possible at a reduced cost, ultimately enhancing service profitability in the long term.

According to a report by Leishman (2021), about 40% of all help desk requests are for password resets, a service that Forrester estimates costs organizations an average of $70 per call (Simic, 2019). Furthermore, with 68% of employees frequently toggling between as many as ten apps every hour (Bennett, 2024), the efficiency losses and administrative costs associated with multiple logins become even more pronounced. By implementing our proposed one-account-for-all universal federator, organizations can significantly reduce the frequency of password reset requests and streamline application access. Thus, our solution presents a cost-saving value for service providers in the long term. In today's highly competitive market, the abundance of alternatives available to consumers means that any friction or complexity encountered with the product could lead consumers to explore other options (McKeown, 2023). Our proposed implementation can significantly enhance user retention by simplifying the sign-up and login-in process, thereby increasing the likelihood that users will consistently choose the product.

7.3. Security analysis

In this section, we examine how the proposed federator provides guarantees for the previously identified security requirements, and explain how it defends against possible attacks.

7.3.1. Confidentiality and integrity

Confidentiality and *integrity* are two of the most important security requirements. Confidentiality and Integrity must be ensured across all stages of communication. Firstly, we divide the message flow into four possible stages; communication between the UE and the foreign service provider (S1), communication between the foreign service provider and federator (S2), federator (S3), and communication between the federator and home service provider (S4). The proposed solution must support *confidentiality* and *integrity* across these four stages.

S1. As we tested our proposed federator for 3GPP cellular networks, cloud, and fog, the security features provided by the cloud, fog, and 3GPP cellular apply to our solution as well. The communication between the UE and the foreign service provider is made secure by the use of existing protocols for cloud, edge, and fog.

S2. Stage S2 involves the communication between the foreign service provider and the federator. This communication must ensure confidentiality and integrity. Confidentiality in this stage is provided through TLS.

S3. Stage S3 involves the information that travels through and is saved inside the federator. All the information is transferred to the federator in an encrypted manner, and the information is also stored inside the federator in an encrypted manner. Therefore, if an adversary gets hold of the user's credentials, it will not be able to read the stored information.

S4. Stage S4 involves the communication between the federator and the home service provider. This communication must ensure confidentiality and integrity. The confidentiality of the user's credentials information is via TLS protocol, as OIDC can use the TLS.

7.3.2. Forward secrecy

The proposed federator uses PKI for key management, and both the user and servers can verify certificates, and initially, they can exchange the session key via PKI. The session key only lasts for one particular session, and unique session keys are freshly generated for each session to ensure the forward secrecy. This ensures that if the hacker obtains the most recent key, it is not able to guess the previous keys and is not able to decrypt the other sessions.

7.3.3. Resistance to replay

In order to resist replay, we use a nonce. Nonce is a string value that is used once for a specific use. If a user tries to replay a token provided by one of the modules in the federator, it can be easily detected via nonce. This ensures that the token is used once and cannot be replayed.

7.3.4. Resistance to MITM and data manipulation attacks

We have considered in our threat model that a user can be malicious, or a third party can get hold of a user and perform malicious activities through the legitimate user's device. This can lead to data manipulation attacks and man-in-the-middle attacks. All the other entities in the system are trusted. The authentication protocols (OIDC, EPS-AKA, and 802.1x) used in this work support the certificate-based authentication and also make use of digital signatures (via TLS). Therefore, MITM attacks and data manipulation attacks are mitigated both on the home side and the foreign side.

7.3.5. Resistance to data leakage

There is a possibility of data leakage attacks and the authentication protocols used in this work support certificate-based authentication, make use of digital signatures (via TLS), and the information saved inside the federator is kept in an encrypted fashion, which makes it impossible for the attacker to get hold of the useful information inside the federator.

7.3.6. Resistance to impersonation

In a typical TLS execution, the server provides the client with the certificate during handshake. Thus, there is no need for the clients to have access to the certificate beforehand. Once the client receives the certificate, it can verify it via the certificate authority's public key. If the certificate is not valid, the client ends the communication. If the certificate is verified, the client continues the communication with the server. The user can easily access these public keys via the browser. The use of certificates makes impersonation attacks impossible.

7.3.7. Compromised federator

In case the federator is compromised by an attacker, the end-to-end trust between the user and multiple service providers will be broken. The service providers will stop communicating with the federator in such a case. The information inside the federator is stored in an encrypted manner and hence, the attacker will not be able to get valuable information, and the threat will be mitigated at the federator and the attacker will not be allowed to infiltrate the cloud, edge, or fog networks via using information obtained from the federator.

8. Conclusions and future research directions

8.1. Conclusions

Establishing a federator among cloud, edge, and fog computing paradigms offers substantial capacity, capability, coverage, and cost-efficiency benefits for end users and service providers. Our proposed federator simplifies the authentication process significantly by eliminating the need for users to create and manage separate accounts for each network. For example, should users misplace their credentials for one network, they can effortlessly authenticate using their credentials from another. This breakthrough improves user experience and stands as a pivotal innovation, streamlining access across the diverse landscape of distributed computing. From the service provider's perspective, our federator offers the potential to optimize resource utilization by offloading specific tasks to edge or fog layers, thereby yielding benefits in terms of resource efficiency. Our experimental findings affirm the federator's effectiveness, demonstrating acceptable authentication times and robust scalability to manage a high volume of concurrent requests. Moreover, our security assessment confirms that the federator meets stringent security standards, ensuring a secure federation

across cloud, edge, and fog computing domains. These achievements underscore the federator's role as a critical enabler in the evolution of distributed computing, promising a more interconnected, efficient, and user-friendly computing environment.

8.2. Future research directions

In the future, our proposed federator can be extended to encompass additional aspects such as authorization, resource allocation, Service Level Agreements (SLAs), traffic offloading, and application mobility, thereby achieving a more comprehensive federation ecosystem. Moreover, the scalability of the federator presents another research direction for its deployment across multiple instances. While our current setup involves a singular federator within the network, the complexity of future network architectures may necessitate the introduction of multiple federators. Another research direction could be the safety of these federators in a multi-federator environment, where a Public Key Infrastructure (PKI) could be employed for enhanced security measures. This would involve each federator possessing unique public and private keys, with the public keys being exchanged securely through PKI certificates issued by a trusted certificate authority. The federator can also be extended to involve more authentication protocols.

CRediT authorship contribution statement

Asad Ali: Writing – original draft, Methodology, Conceptualization. **Ying-Dar Lin:** Supervision, Conceptualization. **Jian Liu:** Visualization, Validation, Investigation. **Chin-Tser Huang:** Validation, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Asad Ali, Ying-Dar Lin has patent #Communication system and method for performing third-party authentication between home service end and foreign service end (US11502987B2) issued to National Yang Ming Chiao Tung University (NYCU). If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Alharbi, S., Rodriguez, P., Maharaja, R., Iyer, P., Subaschandrabose, N., Ye, Z., 2017. Secure the internet of things with challenge response authentication in fog computing. In: 2017 IEEE 36th International Performance Computing and Communications Conference. IPCCC, IEEE, pp. 1–2.

Ali, A., Mallick, T., Sakib, S., Hossain, M., Lin, Y.-D., et al., 2021a. Provisioning fog services to 3GPP subscribers: Authentication and application mobility. arXiv preprint arXiv:2112.02476.

Ali, A., Şahin, A.U., Özkasap, Ö., Lin, Y.-D., 2021b. The universal fog proxy: A third-party authentication solution for federated fog systems with multiple protocols. IEEE Netw. 35 (6), 285–291.

Amanlou, S., Hasan, M.K., Bakar, K.A.A., 2021. Lightweight and secure authentication scheme for IoT network based on publish–subscribe fog computing model. Comput. Netw. 199, 108465. http://dx.doi.org/10.1016/j.comnet.2021.108465.

Aruna, M.G., Hasan, M.K., Islam, S., Mohan, K.G., Sharan, P., Hassan, R., 2022. Cloud to cloud data migration using self sovereign identity for 5G and beyond. Cluster Comput. 25 (4), 2317–2331. http://dx.doi.org/10.1007/s10586-021-03461-7.

Baran, D., 2018. Federated authentication support for OpenNebula.

Bennett, S., 2024. Single sign-on (SSO)statistics 2024. WebinarCare URL: https://webinarcare.com/best-single-sign-on-software/single-sign-on-statistics/.

Bittencourt, L.F., Diaz-Montes, J., Buyya, R., Rana, O.F., Parashar, M., 2017. Mobility-aware application scheduling in fog computing. IEEE Cloud Comput. 4, 26–35.

Bonomi, F., Milito, R., Zhu, J., Addepalli, S., 2012. Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. pp. 13–16.

Choudhruy, H., 2012. A new trust model for improved identity privacy in cellular networks. Int. J. Comput. Appl. 975, 8887.

Choyi, V.K., Brusilovsky, A., 2016. Seamless authentication across multiple entities. Google Patents, US Patent App. 14/779, 584.

Deebak, B., Al-Turjman, F., Mostarda, L., 2020. Seamless secure anonymous authentication for cloud-based mobile edge computing. Comput. Electr. Eng. 87, 106782. http://dx.doi.org/10.1016/j.compeleceng.2020.106782.

Edris, E.K.K., Aiash, M., Loo, J.K.-K., 2020. Network service federated identity (NSFId) protocol for service authorization in 5G network. In: 2020 Fifth International Conference on Fog and Mobile Edge Computing. FMEC, IEEE, pp. 128–135.

Gibbons, K., Raw, J.O., Curran, K., 2014. Security evaluation of the OAuth 2.0 framework. Inf. Manage. Comput. Secur. 22 (3).

Han, K., Ma, M., Li, X., Feng, Z., Hao, J., 2019. An efficient handover authentication mechanism for 5G wireless network. In: 2019 IEEE Wireless Communications and Networking Conference. WCNC, IEEE, pp. 1–8.

Kahvazadeh, S., Souza, V.B., Masip-Bruin, X., Marn-Tordera, E., Garcia, J., Diaz, R., 2017. Securing combined fog-to-cloud system through SDN approach. In: Proceedings of the 4th Workshop on CrossCloud Infrastructures & Platforms. pp. 1–6.

Leishman, G., 2021. Resetting password and saving time and money at the IT help desk. Cisco Duo URL: https://duo.com/blog/resetting-passwords-and-saving-time-and-money-at-the-it-help-desk.

Li, Y., Cheng, Q., Liu, X., Li, X., 2020. A secure anonymous identity-based scheme in new authentication architecture for mobile edge computing. IEEE Syst. J. 15 (1), 935–946.

McKeown, E., 2023. Top Benefits of single sign-on. URL: https://www.pingidentity.com/en/resources/blog/post/top-benefits-sso.html.

Megouache, L., Zitouni, A., Djoudi, M., 2020. Ensuring user authentication and data integrity in multi-cloud environment. Human-centric Comput. Inf. Sci. 10 (1), 15. http://dx.doi.org/10.1186/s13673-020-00224-y.

Mejtoft, T., 2011. Internet of things and co-creation of value. In: 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing. pp. 672–677.

Mell, P., Grance, T., et al., 2011. The NIST Definition of Cloud Computing. Computer Security Division, Information Technology Laboratory.

Naik, N., Jenkins, P., 2016. An analysis of open standard identity protocols in cloud computing security paradigm. In: 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress. DASC/PiCom/DataCom/CyberSciTech, IEEE, pp. 428–431.

Nakkar, M., Altawy, R., Youssef, A., 2020. Lightweight broadcast authentication protocol for edge-based applications. IEEE Internet Things J. 7 (12), 11766–11777. http://dx.doi.org/10.1109/JIOT.2020.3002221.

Navas, J., Beltrán, M., 2019. Understanding and mitigating OpenID connect threats. Comput. Secur. 84, 1–16.

Ogundoyin, S.O., Kamil, I.A., 2021. A lightweight authentication and key agreement protocol for secure fog-to-fog collaboration. In: 2021 IEEE International Mediterranean Conference on Communications and Networking. MeditCom, pp. 348–353.

OpenFog Consortium, et al., 2017. OpenFog Reference Architecture for Fog Computing. Architecture Working Group, pp. 1–162.

Roeland, D., Rommer, S., 2014. Advanced WLAN integration with the 3GPP evolved packet core. IEEE Commun. Mag. 52 (12), 22–27.

Santos, J., Wauters, T., Volckaert, B., De Turck, F., 2019. Towards network-aware resource provisioning in Kubernetes for fog computing applications. In: 2019 IEEE Conference on Network Softwarization. NetSoft, IEEE, pp. 351–359.

Simic, B., 2019. How password-less security benefits helpdesks. Help Net Secur. URL: https://www.helpnetsecurity.com/2019/04/12/password-less-security-benefits-helpdesks/.

Singh, D., Singh, J., Chhabra, A., 2012. High availability of clouds: Failover strategies for cloud computing using integrated checkpointing algorithms. In: 2012 International Conference on Communication Systems and Network Technologies. IEEE, pp. 698–703.

Songhorabadi, M., Rahimi, M., MoghadamFarid, A., Kashani, M.H., 2023. Fog computing approaches in IoT-enabled smart cities. J. Netw. Comput. Appl. 211, 103557.

Tao, M., Ota, K., Dong, M., 2017. Foud: Integrating fog and cloud for 5G-enabled V2G networks. IEEE Netw. 31 (2), 8–13.

Targali, Y., Choyi, V., Shah, Y., 2013. Seamless authentication and mobility across heterogeneous networks using federated identity systems. In: 2013 IEEE International Conference on Communications Workshops. ICC, IEEE, pp. 1232–1237.

Tuli, S., Mahmud, R., Tuli, S., Buyya, R., 2019. Fogbus: A blockchain-based lightweight framework for edge and fog computing. J. Syst. Softw. 154, 22–36.

Villegas, D., Bobroff, N., Rodero, I., Delgado, J., Liu, Y., Devarakonda, A., Fong, L., Sadjadi, S.M., Parashar, M., 2012. Cloud federation in a layered service model. J. Comput. System Sci. 78 (5), 1330–1344.

Vinoth, R., Deborah, L.J., Vijayakumar, P., Gupta, B.B., 2022. An anonymous pre-authentication and post-authentication scheme assisted by cloud for medical IoT environments. IEEE Trans. Netw. Sci. Eng. 9 (5), 3633–3642. http://dx.doi.org/10.1109/TNSE.2022.3176407.

Zhang, Y., Li, B., Liu, B., Hu, Y., Zheng, H., 2021. A privacy-aware PUFs-based multiserver authentication protocol in cloud-edge IoT systems using blockchain. IEEE Internet Things J. 8 (18), 13958–13974. http://dx.doi.org/10.1109/JIOT.2021.3068410.

Zhang, J., Li, T., Ying, Z., Ma, J., 2023. Trust-based secure multi-cloud collaboration framework in cloud-fog-assisted IoT. IEEE Trans. Cloud Comput. 11 (2), 1546–1561.

Asad Ali is currently working as a researcher at the National Institute of Cyber Security (NICS), Taiwan. He received his Ph.D. degree from National Yang Ming Chiao Tung University (NYCU), Taiwan in 2022. He received his master's degree in electrical engineering from National University of Science & Technology (NUST), Pakistan. His research interests are threat intelligence, network security, wireless communications, network design and optimization.

Ying-Dar Lin is a Chair Professor of computer science at National Yang Ming Chiao Tung University (NYCU), Taiwan. He received his Ph.D. in computer science from the University of California at Los Angeles (UCLA) in 1993. His research interests include network softwarization, cybersecurity, and wireless communications. His work on multi-hop cellular was the first along this line, and has been cited over 1000 times. He is an IEEE Fellow and IEEE Distinguished Lecturer. He has served or is serving on the editorial boards of several IEEE journals and magazines, and was the Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST) during 2016–2020.

Jian Liu received his B.S. degree in Applied Chemistry from Tianjin University, China. He received one M.S. degree in Statistics and one M.S. degree in Computer Science from West Virginia University, USA in 2018. Jian is now a Ph.D. student in the Department of Computer Science and Engineering, University of South Carolina. His research focuses on 5G network transmission and security. He is also interested in cloud and edge computing and blockchain.

Chin-Tser Huang is a Professor in the Department of Computer Science and Engineering at University of South Carolina at Columbia. He received the B.S. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 1993, and the M.S. and Ph.D. degrees in Computer Sciences from The University of Texas at Austin in 1998 and 2003, respectively. His research interests include network security, network protocol design and verification, wireless communication systems, cloud and edge computing, blockchain, and game theoretic modeling. He is the director of the Secure Protocol Implementation and Development (SPID) Laboratory at the University of South Carolina. He is the author (along with Mohamed Gouda) of the book "Hop Integrity in the Internet," published by Springer in 2005. His research has been funded by DARPA, AFOSR, AFRL, NSF, and NEH. He is an NRC Research Associate in 2020, and a recipient of the USAF Summer Faculty Fellowship Award and of the AFRL Visiting Faculty Research Program Award in 2008–2020. He is a Senior Member of IEEE and ACM.