Lyu et al., Int J Comput Softw Eng 2022, 7: 176
https://doi.org/10.15344/2456-4451/2022/176

International Journal of
Computer & Software Engineering

Original Article Open Access
Special Issue: Computational Health Informatics

Identity Authentication and Authorization for MEC-based Smart Healthcare System

Jia Lyu, Jianquan Ji, Jianhua Wang and Xiaolin Chang*
Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, P. R. China

Abstract

The rapid development of smart healthcare (s-health) has facilitated information interaction between entities within healthcare systems. In order to enhance the quality of the services provided in healthcare, and satisfy the demand for computing and storage of massive data generated in the healthcare system, the combination of smart healthcare and Multi-access Edge Computing (MEC) technology has become a new trend. However, due to the openness of the MEC, the user's health data in the MEC-based smart healthcare system is faced with security risks and privacy leakage. In this paper, we propose an identity authentication and authorization scheme for the MEC-based smart healthcare system by extending the OAuth2.0 authorization protocol. The proposed scheme allows end-users to access the private medical data stored in MEC and carries out unified identity authentication for users to realize Single Sign-On. Formal analysis (namely, Burrows Abadi-Needham logic) and informal analysis are conducted to analyze the capability of the proposed scheme in resisting replay attacks, man-in-the-middle attacks, and cross-site request forgery attacks. And the results of the simulation indicate that the proposed scheme is feasible.

Publication History:

Received: April 21, 2022
Accepted: May 07, 2022
Published: May 09, 2022

Keywords:

Multi-access edge computing, Smart healthcare, Authorization; Single Sign-On

1. Introduction

The rapid development of the Internet of Medical Things (IoMT) boosts the emergence of various smart healthcare (s-health) systems to monitor the health of patients in real-time. The s-health systems generate a lot of electronic medical records, e.g., diagnostic reports, and physiological parameters. Multi-access Edge Computing (MEC) has the advantages of fast real-time calculation and scalability applied to s-health systems [1]. As illustrated in Figure 1, many people, such as numerous patients, doctors, family members, and other groups, have convenient access to a wide range of medical applications through the MEC-based s-health system. However, these systems open the doors for attackers to launch different types of attacks, such as replay attacks, man-in-the-middle attacks, and cross-site request forgery (CSRF) attacks [2]. These attacks are likely to cause the resources loss of medical scene-oriented information systems and the risk of privacy leakage and data tampering. Therefore, it is of utmost essential that ensure the identity authentication and authorization of s-health systems.

But conventional password-based authentication is painful for users to register each application in the multi-server environments and maintain all pairs of identities and passwords [3]. Single Sign-On (SSO) supports users in accessing series credit applications for authenticating only once [4]. For this reason, implementing SSO is critical to avoid repetitive identity authentication and improve the user experience. Furthermore, SSO is vulnerable to the single point of failure [3]. Existing SSO protocols mainly involve Kerberos [5], Security Assertion Markup Language (SAML) [6], OpenID Connect [7] and OAuth 2.0 [8], etc. The Kerberos protocol is built on the symmetric cryptosystem that works well on small and medium-sized networks, but it cannot detect replay attacks. Because of its diversified functionalities, SAML is well suited for multi-trust domains, but its complexity limits expansion. Ren et al. [9] proposed a unified authentication SSO cross-cloud scheme based on Kerberos and SAML. OpenID Connect and OAuth2.0 pose privacy risks, according to Li et al. [10], because the identity provider can easily intercept access information. Although OAuth has several security flaws in the token transmission and key storage processes, it is suitable for MEC-based s-health systems due to its high flexibility.

To alleviate the above security and privacy issues and it easy for users to access various s-health applications based on MEC, we design a mutual authentication and authorization scheme. Specifically, our contributions to the proposed scheme can be described as follows:

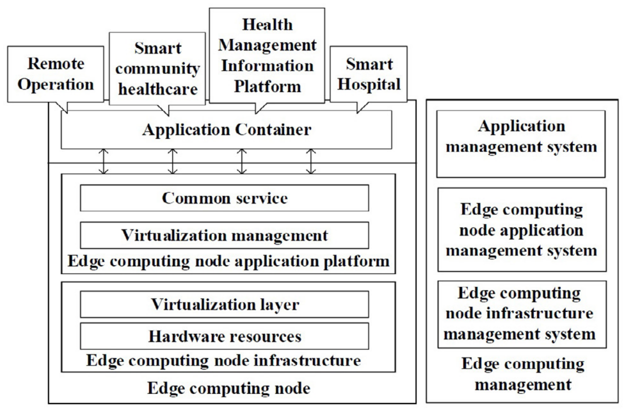

Remote Operation
Smart community healthcare
Health Management Information Platform
Smart Hospital
Application Container
Common service
Virtualization management
Edge computing node application platform
Virtualization layer
Hardware resources
Edge computing node infrastructure
Edge computing node
Application management system
Edge computing node application management system
Edge computing node infrastructure management system
Edge computing management

Figure 1: The MEC-based smart healthcare system.

- To the best of our knowledge, we firstly propose a mutual authentication and authorization scheme for the MEC-based s-health system by extending the OAuth2.0.

***Corresponding Author:** Dr. Xiaolin Chang, Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, P. R. China; E-mail: xlchang@bjtu.edu.cn

Citation: Lyu J, Ji J, Wang J, Chang X (2022) Identity Authentication and Authorization for MEC-based Smart Healthcare System. Int J Comput Softw Eng 7: 176. doi: https://doi.org/10.15344/2456-4451/2022/176

- Our proposed scheme realizes SSO by carrying out unified identity authentication for users. Users only need to log in once but can access all the trusted applications.
- We utilize Burrows Abadi-Needham (BAN) logic to formally analyze the authorization part of the scheme. Considering the limitation of BAN logic, we also supplement the informal analysis.
- The simulation experiments of the proposed scheme are implemented by the OAuth2.0 framework and the OMNET ++ framework. The results indicate the scheme has a low processing latency and transmission latency.

The rest of this paper is organized as follows. Section 2 presents related work. The architecture of MEC and security attacks are presented in Section 3. Section 4 gives the process of our proposed scheme implementation. Section 5 and Section 6 analyze the security and performance of the scheme, respectively. Section 7 concludes the paper.

2. Related Work

In MEC, data processing is no longer transmitted to the remote cloud but only needs to be solved on the edge side. This method not only obtains low latency and high efficiency but also faces the problem of data privacy protection. To ensure data privacy in cloud computing, the authentication and authorization scheme of logging in to cloud platforms accessing protected resources has been a wide concern by academic circles, and a series of researches on unified authentication and SSO technology for users have been carried out. SSO protocols mainly involve Security Assertion Markup Language (SAML), OpenID Connect, and OAuth2.0. This section mainly concentrates on SSO technology based on OAuth2.0 in the MEC environment.

Meniya et al. [11] discussed the concept of SSO. Resource data is used in the open cloud by sharing. All cloud service providers only accept the same SSO, which provides convenience for users by increasing the interoperability between cloud applications. Ren et al. [9] proposed a unified authentication SSO cross-cloud scheme based on Kerberos and SAML. Sven et al. [7] studied the SSO protocol based on OpenID Connect. In order to prevent identity providers from knowing the relying party of users' login, the OpenID Connect protocol was extended to solve the privacy problem and avoid collision between identity providers and relying on parties to track users. Li et al. [10] systematically analyzed the security vulnerabilities of OAuth2.0 and OpenID Connect protocols in SSO protocol in terms of user privacy protection, described the simplicity of identity provider tracking user access, and proposed methods to alleviate these privacy issues.

Khan et al. [8] proposed an authentication scheme based on OAuth2.0 for IoT security. By comparing the user information and access token in the security manager's local database, only authorized or trusted users can access the IoT network. Liu et al. [12] discussed OAuth, one of the most popular SSO protocols, and summarized the security problems in the implementation of OAuth2.0 under the Android environment, including the storage of client keys, access tokens, improper handling of redirection in applications, etc., and analyzed the vulnerabilities, and then discussed the direction to reduce these security problems. It is similar to our work, but our design is oriented to the MEC environment. By analyzing the vulnerability of the extended OAuth2.0 protocol, we propose corresponding solutions to improve the overall security of the SSO protocol.

3. Preliminary

In this section, we describe the MEC components involved in the follow-up of the paper. The detailed structure of MEC can be seen in [13]. After that, we analyze the possible security attacks of building SSO protocol based on OAuth 2.0.

Architecture of MEC

From Figure 2, we can see that MEC architecture is divided into two parts: mobile edge host level and mobile edge system level. Among them, MEP is responsible for receiving traffic forwarding rules from the platform manager, MEC apps, or MEC services, and then issuing instructions to the plane based on the forwarding rules. MEC app is a virtual machine instance running on the virtualization infrastructure, which can be a third-party application developed by third-party developers. The virtualization infrastructure provides computing, storage, and network resources for mobile edge applications. MEPM realizes the creation and termination of MEC apps, the authentication of applications, and the management of traffic rules. MEO is the core function provided by MEC, which controls the resources and capacity of the MEC network. OSS is used to receive requests for instantiation or termination of MEC applications from the CFS portal and user terminals. It can check the integrity and authorization information of application requests, and then forward the requests to MEO for further processing. Table 1 defines all acronyms involved in MEC architecture.

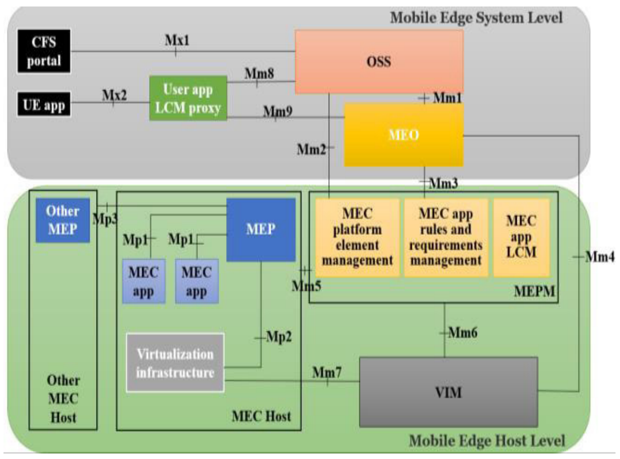


Figure 2: MEC architecture.

Acronym	Definition
CFS portal	Customer-Facing Service Portal
UE app	User Equipment application
User app LCM proxy	User App Life-Cycle Management Proxy
OSS	Operations Support System
MEO	Mobile Edge Orchestrator
MEP	Mobile Edge Platform
MEPM	Mobile Edge Platform Manager
VIM	Virtualization Infrastructure Manager

Table 1: Definiation of acronyms

Security attacks

This section presents the attacks which our proposed scheme can defend against.

Replay attack

In the authorization and authentication process based on OAuth2.0, attackers usually replay the authorization code [14]. After the user agrees to authorize, the authentication server will send an authorization code to respond. Malicious personnel will intercept this process, and then manually issue the authorization code to the redirection address of the application. This will trigger the authorization code exchange mechanism of the client. The application mistakenly considers the attacker as the authentication server-side to respond.

Man-in-the-middle attack

At present, it is very serious for the attacker to implement man-in-the-middle attacks in OAuth2.0 protocol by maliciously tampering with the redirection address of the authorization code **Error! Reference source not found.** . Li et al. [16] studied the SSO website based on OAuth2.0, which supports Google, Facebook, and other accounts to log in. This paper presents a redirection address manipulation attack against OAuth2.0. The attacker can obtain the victim's authorization code without the user's knowledge. Li et al. [17] also discussed that many application relying parties do not use encrypted channels to protect Google login data transmission. Moreover, the service provider often ignores to check whether the redirection address has been tampered with, which leads to the leakage of the authorization code.

Cross-site request forgery (CSRF) attack

In OAuth2.0, when the attacker implements the CSRF attack, he first uses his account to login into the application and obtains the authorization code. When a legitimate user requests resource access through authorization, the attacker forges a successful authorization request to the user, replaces the authorization code with his previously received authorization code, and sends it to the user. The user is induced to initiate an access token exchange request, which enables the user to obtain the attacker's access token. After the attack is completed, the user's identity account will be bound with the attacker's access token, and the attacker's account will also be bound with the user. Li et al. [18] discussed that an attacker can use the CSRF attack to control the application account of the victim user without knowing the user's username and password.

4. The Proposed Scheme

This section details the proposed authentication and authorization scheme for the MEC-based smart healthcare system and introduces the scheme according to two situations: first login to the MEC host and second login. Table II defines the parameters to be used in the rest of the paper.

First login

As shown in Figure 3, when the user accesses the application for the first time to request the resource service of the MEC-based healthcare system, MEO will authenticate and authorize the user. After MEC App obtains the access token returned by MEPM, the application can access the protected resources in the MEC-based healthcare system according to the access rights of the token. The specific process is as follows:

Parameter	Definition
App_id	The identity of the application
App_secret	The key to the application
Redirect_uri	The URI address of the redirect
Response_type	Represents the authorization type, whose value is fixed as a "token"
code	The authorization code is used to request an access token
state	Any random value
Access_token	Used to access protected resources in MEC
Refresh_token	Update the token to be used to get the next access token
scope	The scope of the access permissions
Grant_type	Represents the authorization model, this value is fixed as "authorization_code"
Expires_in	The expiration time of the access token
Token_type	The type of the access token

Table 2: Definiation of parameters.

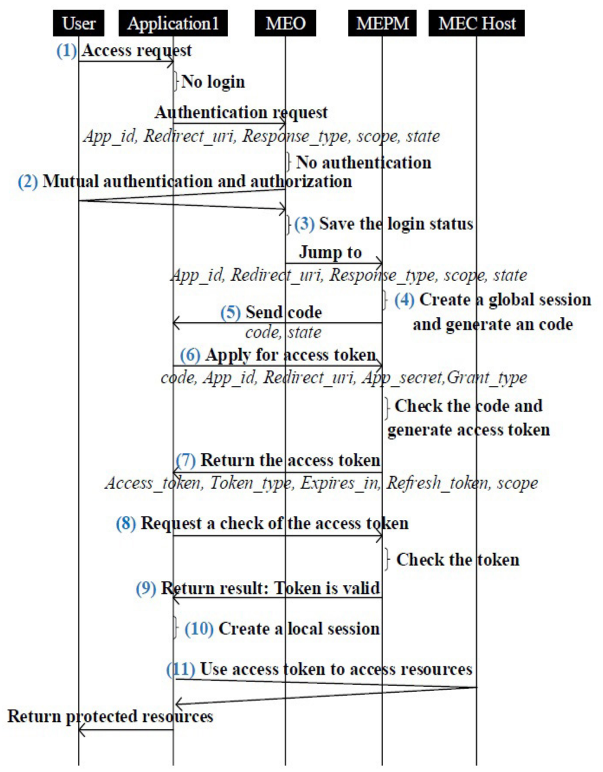

User
Application1
MEO
MEPM
MEC Host
(1) Access request
No login
Authentication request
App_id, Redirect_uri, Response_type, scope, state
No authentication
(2) Mutual authentication and authorization
(3) Save the login status
Jump to
App_id, Redirect_uri, Response_type, scope, state
(4) Create a global session and generate an code
(5) Send code
code, state
(6) Apply for access token
code, App_id, Redirect_uri, App_secret, Grant_type
Check the code and generate access token
(7) Return the access token
Access_token, Token_type, Expires_in, Refresh_token, scope
(8) Request a check of the access token
Check the token
(9) Return result: Token is valid
(10) Create a local session
(11) Use access token to access resources
Return protected resources

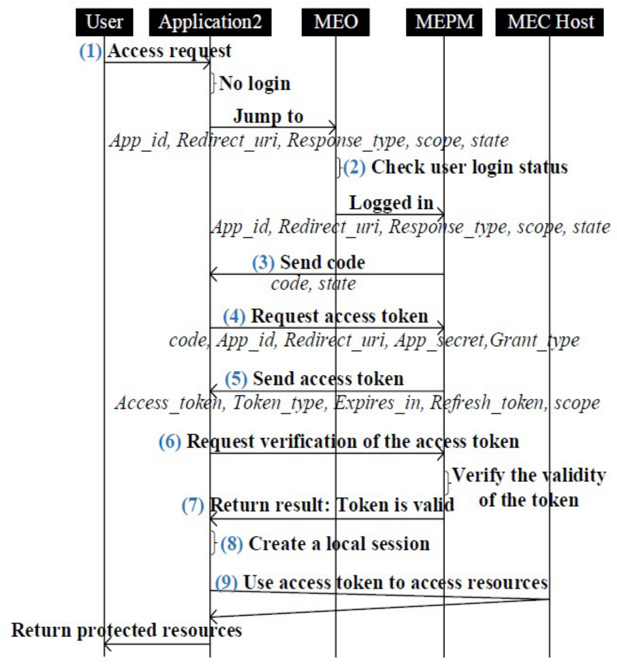
Figure 3: The authentication process of the first login.

1. When a user makes a resource access request to the MEC-based healthcare system through the MEC application for the first time, the application will send the user identity authentication request to MEO.
2. After checking the database, MEO finds that the user is not authenticated to log in, and then uses the redirection address sent by the application client to return to the authentication and authorization page of the MEC-based healthcare system for authentication.

3. After authentication, MEO saves the user login status, and then sends the received parameters to MEPM.
4. MEPM checks the client by comparing whether the *App_id* and *Redirect_uri* parameters are consistent with the application in the registration and then generates the authorization code.
5. MEPM returns the redirection address specified by the application in advance and carries the state random parameter to prevent replay attacks.
6. The application uses the received authorization code to apply to MEPM for access tokens and carries the key *App_secret* to help MEPM authenticate itself.
7. MEPM judges the validity of application identity by verifying the application key and verifies the authorization code. If the verification is successful, MEPM generates the access token,s and then returns the token with the parameters such as *Token_type, Expires_in, scope*, etc.
8. In order to prevent the access token from being attacked, the application will request to verify the validity of the token.
9. After the token is verified, MEPM returns the verification result
10. The application will create a local session with the user after receiving the token validation message.
11. When users request access to resources, the application uses access tokens to access the protected resources in the MEC host and complete the application service.

Different from the user who first logins to the application based on the MEC-based healthcare system, during the second login process showed in Figure 4, MEO checks the login status of the user in the database to determine whether re-authentication is needed. If the login status of the user is confirmed to be valid, MEPM will directly return the authorization code. In the process of returning the access token later, MEPM can query the database directly and return the authorization token associated with the current user to the user without regenerating it. The application will apply to MEPM to check whether the access token is valid. The application can access the protected resources only after it is confirmed to be valid. Otherwise, the token will be updated. The specific process is as follows:

1. When users send resource access requests to other applications in the MEC-based healthcare system, if they are not authenticated, the application client will also jump to MEO for authentication.
2. By checking the user's login status, MEO finds that the user has logged in, then authenticates the validity of the login status. After confirming that the user login status is valid, jump to MEPM.
3. MEPM verifies the application identity according to the received parameters, and then generates the authorization code and sends it to the client.
4. The application obtains access tokens by sending the authorization code to MEPM and carries parameters with its information to facilitate MEPM authenticating the application.
5. After verification, MEPM finds the access token associated with the current user from the database and returns it.
6. After receiving the token, the application will request the MEPM to verify the validity of the token.
7. MEPM returns the final verification result.
8. After receiving the valid information of the token, the application creates a local session with the user.
9. With the access token, the application can directly obtain the protected resources in the MEC host and return them.


User
Application2
MEO
MEPM
MEC Host
(1) Access request
No login
Jump to
App_id, Redirect_uri, Response_type, scope, state
(2) Check user login status
Logged in
App_id, Redirect_uri, Response_type, scope, state
(3) Send code
code, state
(4) Request access token
code, App_id, Redirect_uri, App_secret,Grant_type
(5) Send access token
Access_token, Token_type, Expires_in, Refresh_token, scope
(6) Request verification of the access token
Verify the validity of the token
(7) Return result: Token is valid
(8) Create a local session
(9) Use access token to access resources
Return protected resources
Figure 4: The authentication process of the second login.

5. Security Analysis

In this section, we use Burrows Abadi-Needham (BAN) logic to evaluate the security of the authorization part based on OAuth2.0. Then we carry out an informal analysis of the parts that are not considered by BAN logic and finally achieve the purpose of verifying the security of the scheme.

Formal analysis of BAN logic

In this part, we use BAN logic to formally analyze the authentication and authorization part of the scheme.

1. Basic introduction and proof rules

BAN logic contains three kinds of processing objects: subject, key, and formula. P and Q denote the principal variable, K represents the shared key variable, X and Y represent the formula variable, and N represents the specific random value. Table 3 lists the common BAN logic symbols.

$P \Delta X$	P received a message containing X
$P \mid\sim X$	P sent a message containing X
$P \mid\equiv X$	P believes that X is true
$P \mid\Rightarrow X$	P has control over X
$\#(X)$	X is fresh
$P \overset{k}{\leftrightarrow} Q$	The shared key K is used to communicate between P and Q
$P \overset{X}{\rightleftharpoons} Q$	X is the shared secret of P and Q
$\{X\}_k$	Encrypt formula X with key K

Table 3: Defination of BAN logic symbol.

The following is the BAN logic rules used in this paper. The formula on the horizontal line represents the premise, and the formula below the horizontal line represents the conclusion based on the premise.

$$\text{R1}: \frac{P|\equiv p\overset{k}{\leftrightarrow}Q,\ P \lhd \{X\}_k}{P|\equiv Q|\sim X}$$

$$\text{R2}: \frac{P|\equiv \#(X),\ P|\equiv Q|\sim X}{P|\equiv Q|\equiv X}$$

$$\text{R3}: \frac{P|\equiv Q|\Rightarrow X,\ P|\equiv Q|\equiv X}{P|\equiv X}$$

$$\text{R4}: \frac{P|\equiv p\overset{k}{\leftrightarrow}Q,\ P \lhd \{X\}_k}{P \lhd X}$$

$$\text{R5}: \frac{P|\equiv \#(X)}{P|\equiv \#(X,Y)}$$

Protocol description

To facilitate the BAN logic analysis of the protocol, we regard the authorization code and access token as secrets, and the *state* parameter is the random number N. And this scheme uses the SSL encryption channel for communication, so the shared key Kam and Kah are set to encrypt the transmission data.

The specific construction process will be introduced in the following part of the paper.

a) $A \rightarrow M$: $\text{Message}_0 = \{N, T_A\}K_{am}$

b) $M \rightarrow M$: $\text{Message}_1 = \{M \overset{Y}{\rightleftharpoons} A, N, T_M\}\ K_{am}$

c) $A \rightarrow M$: $\text{Message}_2 = \{M \overset{Y}{\rightleftharpoons} A, K_a, T_A\}\ K_{am}$

d) $M \rightarrow A$: $\text{Message}_3 = \{H \overset{X}{\rightleftharpoons} A, ,T_M\}\ K_{am}$

e) $A \rightarrow H$: $\text{Message}_4 = \{H \overset{X}{\rightleftharpoons} A, ,T_A\}\ K_{ah}$

M	MEPM
H	MEC Host
A	Application
K_{am}	The shared key between MEPM and Application
K_{ah}	The shared key between MEC Host and Application
N	Random number "*state*"
K_a	"*App_secret*" Key for the application
X	Access token
Y	Authorization code
T	The timestamp

Table 4: Defination of symbols

Basic assumptions and verification objectives

Table 5 lists all the safety assumptions, and the final verification objectives are as follows:

$$A|\equiv H \overset{X}{\rightleftharpoons} A$$

$$H|\equiv H \overset{X}{\rightleftharpoons} A$$

Basic assumptions	
$A|\equiv A\overset{kam}{\leftrightarrow}M$	$H|\equiv A\overset{kah}{\leftrightarrow}H$
$M|\equiv A\overset{kam}{\leftrightarrow}M$	$A|\equiv A\overset{kah}{\leftrightarrow}H$
$\#(N)$	$\#(T_A)$
$\#(T_M)$	$A|\equiv \#(T_M)$
$M|\equiv \#(T_A)$	$M|\equiv \# N$
$H|\equiv \#(T_A)$	$H|\equiv M$
$M|\Rightarrow H \overset{X}{\rightleftharpoons} A$	$A|\equiv M|\Rightarrow H \overset{X}{\rightleftharpoons} A$

Table 5: Basic assumptions

Process of proof

a) Message_0 and Message_1 show the following

- $M \lhd N$ (According to R4)
- $A \lhd (M \overset{Y}{\rightleftharpoons} A, N, T_M)$ (According to R4)
- $A|\equiv M|\sim (M \overset{Y}{\rightleftharpoons} A, N, T_M)$ (According to R1)
- $A|\equiv M|\equiv M \overset{Y}{\rightleftharpoons} A$ (According to R5 & R2)

b) Message_2 show the following

- $M \lhd (M \overset{Y}{\rightleftharpoons} A, K_a, T_A)$ (According to R4)
- $M|\equiv A|\sim (M \overset{Y}{\rightleftharpoons} A, K_a, T_A)$ (According to R1)
- $M|\equiv A|\equiv M \overset{Y}{\rightleftharpoons} A$ (According to R5 & R2)

Here we can prove that the authorization code is verified successfully.

c) Message_3 show the following

- $A \lhd (H \overset{X}{\rightleftharpoons} A, T_A)$ (According to R4)
- $A|\equiv M|\sim (H \overset{X}{\rightleftharpoons} A, T_M)$ (According to R1)
- $A|\equiv M|\equiv H \overset{X}{\rightleftharpoons} A$ (According to R5 & R2)
- $A|\equiv H \overset{X}{\rightleftharpoons} A$ (According to R3)

The application successfully exchanged access token.

d) Message_4 show the following

- $H \lhd (H \overset{X}{\rightleftharpoons} A, T_A)$ (According to R4)
- $H|\equiv A|\equiv H \overset{X}{\rightleftharpoons} A$ (According to R1, R5, R2)

Since MEPM and MEC host are mutually trusted entities in the MEC-based healthcare system, and according to $M|\Rightarrow H \overset{X}{\rightleftharpoons} A$, we obtain $H|\equiv H \overset{X}{\rightleftharpoons} A$ Through BAN logic analysis and verification, it is proved that the scheme proposed in this paper can achieve the expected security goal.

Informal analysis

Because BAN logic has some limitations in terms of semantic definition and initial assumptions, it cannot analyze some attacks. In this part, we make an informal analysis based on the following attacks, which BAN logic does not provide.

Man in the middle attack protection

The authorization code generated by MEPM is a callback through the redirection address, which is easy to intercept [19]. Therefore, to prevent the privacy information from being stolen by the attacker in the transmission process, the implementation of SSO ensures the integrity of the token by constructing an SSL encryption channel between the application and the MEC authentication server. The specific implementation process is shown in Figure 5.

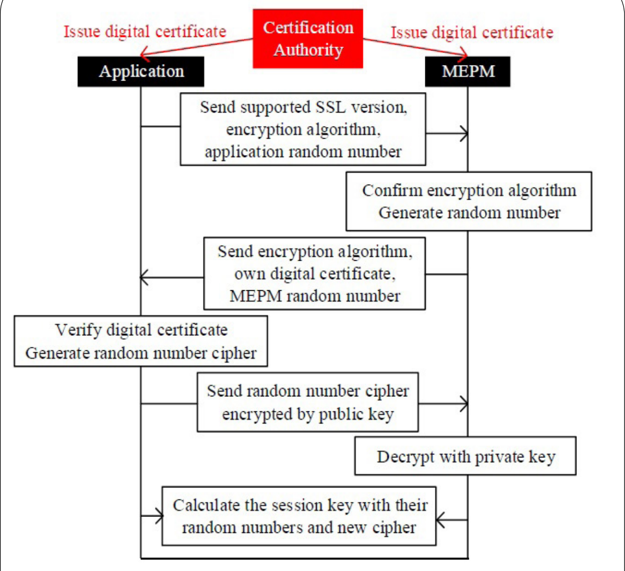

Issue digital certificate
Certification Authority
Issue digital certificate
Application
MEPM
Send supported SSL version, encryption algorithm, application random number
Confirm encryption algorithm Generate random number
Send encryption algorithm, own digital certificate, MEPM random number
Verify digital certificate Generate random number cipher
Send random number cipher encrypted by public key
Decrypt with private key
Calculate the session key with their random numbers and new cipher

Figure 5: Construction model of SSL encryption channel.

CSRF attack protection

To mitigate this attack, *code* is set as one-time encoding. The authorization code will be invalid immediately after being used once, and its validity period is only 30s. If both the attacker and MEC application request the access token from MEPM by using the obtained authorization code, MEPM will invalidate the access token sent to them because this authorization code is used twice. In addition, MEPM will verify the binding relationship between *code* and *App_id* when verifying the authorization code.

In addition, the *state* random number is used as the parameter that the application must carry when making the authorization request. Yang et al. [20] tested more than 400 websites that used OAuth2.0 and ranked at the top. They found that more than half of the websites were not or inappropriate to use *state* parameters, thus being attacked by CSRF. Therefore, in this scheme, the *state* parameter has the following characteristics:

- Unpredictability: the *state* is a random parameter that is difficult for attackers to predict.
- Relevance: The value of *state* is associated with the current session.
- Uniqueness: The *state* generated by each user or each request is unique.
- Timeliness: Once the *state* parameter is used, it will be invalid immediately.

When a user makes a resource access request, the application redirects the user to MEPM. At this point, MEPM generates random numbers based on user information to be added to the session as *state* parameters. After that, the application carries the *state* parameter when requesting the authorization code, and MEPM judges the validity of the request by verifying the *state*. The authorization code and *state* are returned together, and the application verifies whether the two *state* parameters are consistent. If they are inconsistent, the application will reject directly.

Access token theft protection

For the application accessing to MEC-based healthcare system, the authentication is completed through registration, and MEPM will issue application authentication information including *App_id* and *App_secret*. *App_id* is used to identify the identity of the application, and *App_secret* is used as the key of the application. Only the application and MEPM can know it. When an application requests an access token, it will carry an *App_secret*. MEPM verifies the identity of the application by verifying the *App_secret*, to prevent the attacker from obtaining the access token illegally.

6. Performance Analysis

This section evaluates the performance of the scheme in the aspects of transmission delay and processing delay.

Experimental model

To evaluate the performance of the identity authentication and authorization scheme proposed herein, the main functional operations of the scheme are measured. The evaluation of main function operations is carried out under the environment of operating system 64-bit Window 10 and the processor of Intel (R) Core (TM) i7-10510U CPU @ 1.80GHz. The processing delay of the identity authentication and authorization scheme proposed in the IV section is d by OAuth 2.0 framework.

The measurement of transmission overhead is accomplished by an OMNET ++ 5.6.2 [21] framework which is built in an operating system 64-bit Window 11 and Intel (R) Core (TM) I5-8250U CPU @ 1.60GHz processor. Simulation of the two login processes is implemented by defining network scenarios, node types, and module connections. The specific model is shown in Figure 6.

Analysis of measurement results

The main operations are divided into seven parts, such as user authorization, token encryption, token creation, token refresh, token authentication, BASE64, and timestamp service. The processing latency of main operations is shown in Table 6.

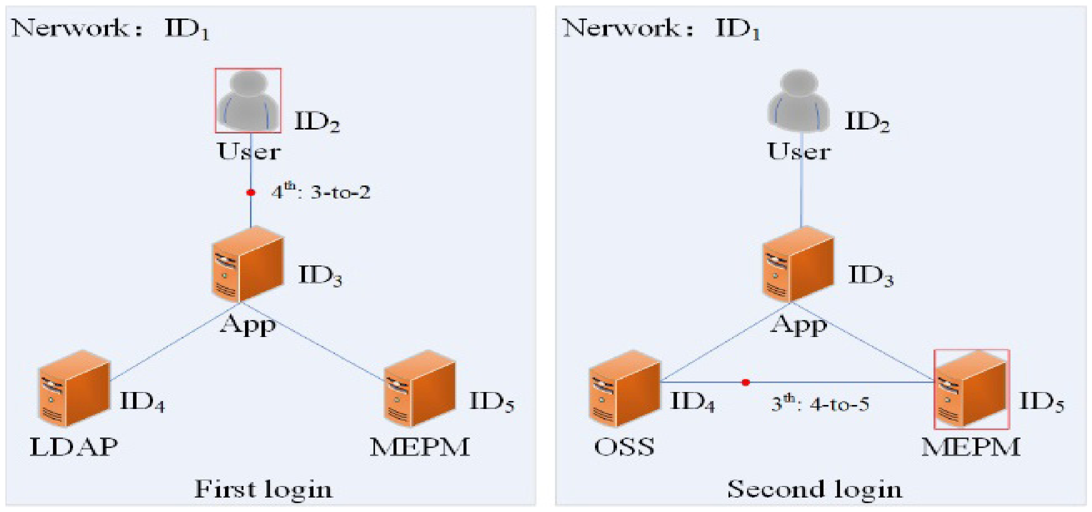


Figure 6: The model of the simulation system.

Par	Main operation	Processing latency /s
UA	User Authorization	0.164
TE	Token Encryption (HMAC-SHA I)	0.248
TC	Token Creation	0.192
TR	Token Refresh	0.19
TR	Token Authentication	0.272
B	BASE64	0.028
TS	Timestamp Service	0.015

Table 6: Processing latency of the main operation.

Further, the processing delay is measured in three different cases. In the first case, that is, users should complete a series of operations such as identity authentication, token distribution, encryption, and verification in the first login process. In the second case, users issue access requests to the credit application, namely the second login. Then the user can enjoy the service directly while carrying the access token and verified as a valid state. However, due to the token being invalid in the third case, the service is available only after re-authorization and token refresh operations.

Figure 7 compares the processing time in the above three cases. The reductions of the processing latency between the first login and the two situations of the second login (valid token and invalid token) are 68.2% and 27.7% respectively. Besides, traditional simple logins are required to repeat the registration process when each application first login, which will lead to a bad user experience. In our scheme, users only need to register once and complete the login of all credit applications. Therefore, the proposed scheme greatly reduces the time overhead of single sign-on.

In addition, OMNET ++ simulates three login processes, measuring the overall transmission delay required to complete the protocol. From Table 7, the transmission delay of the second login process is reduced by 51.78% and 18.49% respectively compared with the first login.

Finally, Figure 8 shows the result which is combined with the processing delay with the transmission delay. Unlike the first login,

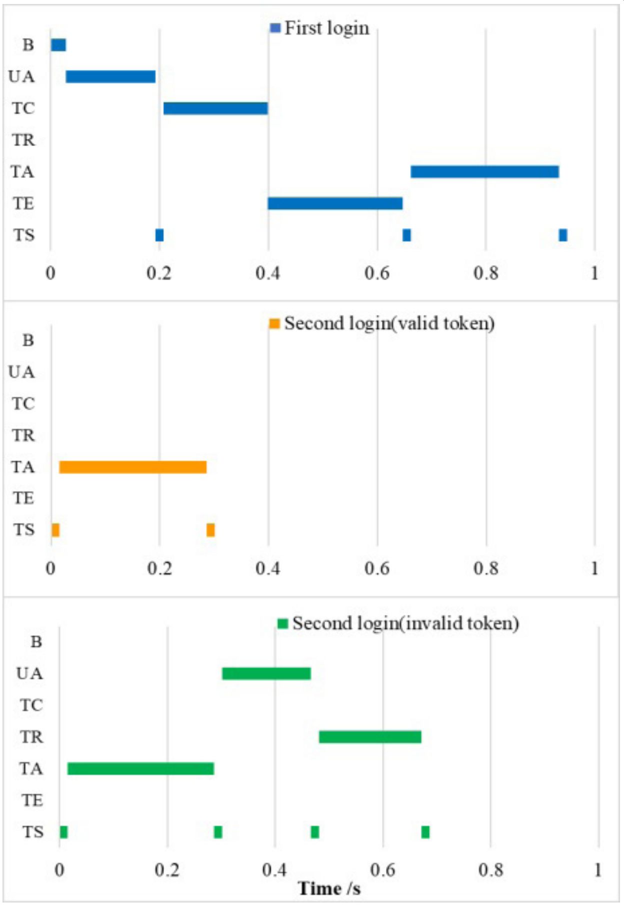


Figure 7: Processing latency comparison.

Phase	Average transmission delay /s	Total transmission delay /s
First login	0.075	0.898
Second login (valid token)	0.072	0.433
Second login (invalid token)	0.073	0.732

Table 7: Transmission overhead.

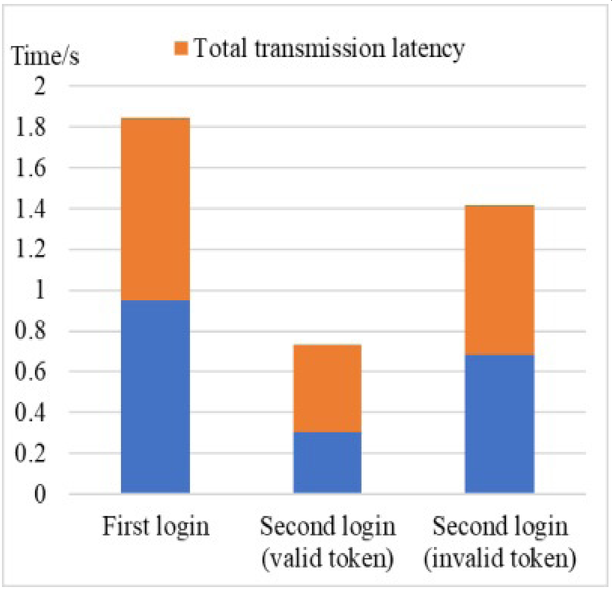


Figure 8: Total latency comparison.

the second login process has greatly improved regardless of whether the token is valid. The second login process has improved in two aspects of transmission latency and processing latency and ultimately reduces total delay by 60.20% and 23.22%. Compared with the traditional login, which needs repeated registration when facing multiple application systems, this scheme does not need a redundant registration process. Therefore, the scheme also has the advantages of low processing delay and transmission delay at the first login. At the same time, the authorization process of the second login greatly reduces the overhead of the total scheme. In summary, the proposed scheme is feasible and practical.

Conclusions

In this paper, we propose an identity authentication and authorization scheme for the MEC-based smart healthcare system. The scheme is based on the OAuth2.0 protocol and compatible with the MEC-based smart healthcare system. At the same time, the scheme realizes SSO to facilitate users to access all the protected resources in the system through once unified authentication. Moreover, BAN logic is adopted to formally analyze the scheme. In addition, considering the limitations of BAN logic, we supplement the informal analysis, and finally validate the capability of the scheme in resisting replay attacks, man-in-the-middle attacks, and CSRF attacks. The performance evaluation is completed by simulation experiments, which proves that the proposed scheme is feasible and effective.

Competing Interests

The authors declare that they have no competing interests.

References

1. Bhuiyan MN, Rahman MM, Billah MM, Saha D (2021) Internet of Things (IoT): A Review of Its Enabling Technologies in Healthcare Applications, Standards Protocols, Security, and Market Opportunities. IEEE Internet Things J 8: 13.
2. Hathaliya JJ, Tanwar S (2020) An exhaustive survey on security and privacy issues in Healthcare 4.0. Comput Commun 153: 311–335.
3. de Almeida MG, Canedo ED (2022) Authentication and Authorization in Microservices Architecture: A Systematic Literature Review. Appl Sci 12: 6.
4. Sciarretta G, Carbone R, Ranise S, Viganò L (2020) Formal Analysis of Mobile Multi-Factor Authentication with Single Sign-On Login. ACM Trans Priv Secur 23: 1-37.
5. Al-Dubai YF, Khamitkar SD (2022) Kerberos: Secure Single Sign-On Authentication Protocol Framework for Cloud Access Control. Glob J Comput Sci Technol.
6. Monzillo R, Kaler C, Nadalin A, Hallam-Baker P, Milono C (2006) Web Services Security: SAML Token Profile.
7. Hammann S, Sasse R, Basin D (2020) Privacy-Preserving OpenID Connect," in Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, New York, NY, USA, pp. 277-289.
8. Khan J, Li JP, Ali I, Parveen S, Khan G, et al. (2018) An Authentication Technique Based on Oauth 2.0 Protocol for Internet of Things (IoT) Network, in 2018 15$_{th}$ International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), pp. 160–165.
9. Ren X, Shi L (2014) The cloud single sign-on authentication based on Kerberos and SAML. WIT Trans Inf Commun Technol 52: 1311–1320.
10. Li W, Mitchell CJ (2020) User Access Privacy in OAuth 2.0 and OpenID Connect," in 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), pp 664–6732.
11. Meniya AD, Jethva HB (2012) Single-Sign-On (SSO) across open cloud computing federation. Int J Eng Res Appl 2: 891–895.
12. Liu X, Liu J, Wang W, Zhu S () Android single sign-on security: Issues, taxonomy and directions. Future Gener Comput Syst 89: 402–420.
13. Ranaweera P, Jurcut AD, Liyanage M (2019) Realizing Multi-Access Edge Computing Feasibility: Security Perspective," in 2019 IEEE Conference on Standards for Communications and Networking (CSCN), pp. 1–7.
14. A study on secure user authentication and authorization in OAuth protocol | SpringerLink.
15. Rahman SS, Hossain N, Hossain MA,Hossain MZ, Sohag MHI (2020) OAuth 2.0: A Framework to Secure the OAuth-Based Service for Packaged Web Application, in Innovative Perspectives on Interactive Communication Systems and Technologies, IGI Global, 2020, pp. 92–139.
16. Li W, Mitchell CJ, Chen T (2018) Your Code Is My Code: Exploiting a Common Weakness in OAuth 2.0 Implementations. In: Security Protocols XXVI. Security Protocols 2018. (pp. 24-41). Cham, Switzerland: Springer.
17. Analysing the Security of Google's Implementation of OpenID Connect | SpringerLink.
18. Li W, Mitchell CJ (2014) Security Issues in OAuth 2.0 SSO Implementations. In: Chow SSM, Camenisch J, Hui LCK, Yiu SM (eds) Information Security. ISC 2014. Lecture Notes in Computer Science, vol 8783. Springer, Cham.
19. Li W, Mitchell CJ, Chen T (2018) Mitigating CSRF attacks on OAuth 2.0 Systems," in 2018 16th Annual Conference on Privacy, Security and Trust (PST), pp. 1–5.
20. Yang R, Li G, Lau WC, Zhang K, Hu P (2016) Model-based Security Testing: An Empirical Study on OAuth 2.0 Implementations, in Proceedings of the 11$_{th}$ ACM on Asia Conference on Computer and Communications Security, New York, NY, USA, pp. 651–662.
21. Varga A (2010) OMNeT++," in Modeling and Tools for Network Simulation, Wehrle K, Güneş M, and Gross J, Eds. Berlin, Heidelberg: Springer, pp 35-59.

This article was originally published in a special issue:

Computational Health Informatics

Handled by Editor:

Prof. Eisuke Hanada
Department of Information Science
Saga University
Graduate School of Science and Engineering
Japan