Journal of Medical Imaging

MedicalImaging.SPIEDigitalLibrary.org

OpenID Connect as a security service in cloud-based medical imaging systems

Weina Ma
Kamran Sartipi
Hassan Sharghigoorabi
David Koff
Peter Bak

Weina Ma, Kamran Sartipi, Hassan Sharghigoorabi, David Koff, Peter Bak, "OpenID Connect as a security service in cloud-based medical imaging systems," *J. Med. Imag.* **3**(2), 026501 (2016),

OpenID Connect as a security service in cloud-based medical imaging systems

Weina Ma,[a,*] Kamran Sartipi,[b] Hassan Sharghigoorabi,[a] David Koff,[c] and Peter Bak[c]
[a]University of Ontario Institute of Technology, Department of Electrical, Computer and Software Engineering, 2000 Simcoe Street North, Oshawa, Ontario L1H 7K4, Canada
[b]McMaster University, Information Systems, 1280 Main Street West, Hamilton, Ontario L8S 4M4, Canada
[c]McMaster University, Department of Radiology, 1280 Main Street West, Hamilton, Ontario L8S 4L8, Canada

Abstract. The evolution of cloud computing is driving the next generation of medical imaging systems. However, privacy and security concerns have been consistently regarded as the major obstacles for adoption of cloud computing by healthcare domains. OpenID Connect, combining OpenID and OAuth together, is an emerging representational state transfer-based federated identity solution. It is one of the most adopted open standards to potentially become the *de facto* standard for securing cloud computing and mobile applications, which is also regarded as "Kerberos of cloud." We introduce OpenID Connect as an authentication and authorization service in cloud-based diagnostic imaging (DI) systems, and propose enhancements that allow for incorporating this technology within distributed enterprise environments. The objective of this study is to offer solutions for secure sharing of medical images among diagnostic imaging repository (DI-r) and heterogeneous picture archiving and communication systems (PACS) as well as Web-based and mobile clients in the cloud ecosystem. The main objective is to use OpenID Connect open-source single sign-on and authorization service and in a user-centric manner, while deploying DI-r and PACS to private or community clouds should provide equivalent security levels to traditional computing model. [DOI: 10.1117/1.JMI.3.2.026501]

Keywords: medical imaging; picture archiving and communication systems; security service; OpenID Connect; cloud.

Paper 15145PR received Jul. 16, 2015; accepted for publication May 18, 2016; published online Jun. 16, 2016.

1 Introduction

In medical imaging, picture archiving and communication system (PACS) is a complex integrated system equipped with a series of hardware and software components, including digital imaging acquisition devices, namely modalities (e.g., CT scanner, MRI system, and x-ray); digital image storage and archives where the acquired images are stored; and workstations where radiologists view the images and produce diagnostic reports.[1] With the increasing demand of collaborative work and sharing of medical information, PACS systems in different hospitals or image centers are interconnected across a distributed environment. Diagnostic imaging repository (DI-r) provides a solution for sharing (reliably storing, publishing, discovery, and retrieving) of DI documents across affiliated healthcare organizations. According to the status of DI-r projects across Canada,[2] provincial DI-r's have been developed to deliver fast and easy access to diagnostic images to all authorized healthcare providers.

With the exploding rate of medical imaging data and the fast growth of the medical imaging market, migrating PACS systems and DI-r services to cloud computing is cost-effective and improve the quality of medical imaging services. Cloud computing is a preferred solution for information sharing over the Internet using external infrastructure, which allows access to applications and data on demand, at any time, and from anywhere. By taking advantage of the cloud capabilities, the medical imaging systems would significantly benefit both from allowing organizations to deploy applications without constraining to their own physical facilities, and from delivering better services to patients using the variety of medical imaging service providers. However, it is a major challenge to manage the identity of various participants (i.e., users, devices, and applications) and ensure all service providers can provide authentication and authorization mechanisms with the same level of security in the cloud ecosystem. Meanwhile, maintaining a separate user identity repository in each individual domain can lead to information inconsistency and synchronization problems.

Federated identity management enables the users in one domain to securely and seamlessly access data in another domain. With the shift of federated identity solutions from organization-centric to user-centric, account information is persisted and managed by third-party services and the users are authenticated by cooperating sites (e.g., PACS and DI-r services) using these services. An innovative single sign-on (SSO) solution that provides delegated authentication service is necessary for cloud-based medical imaging systems because it is able to (i) authenticate users without exposing user credentials to third-party service provides (PACS systems and DI-r services); (ii) apply the authentication mechanism to all participants; and (iii) provide a common method to integrate with heterogeneous medical imaging systems.

OAuth 2.0 is an open standard for authorization.[3] It defines specific authorization flows for conveying authorization decisions across the network for web applications, desktop applications, and mobile applications. OAuth 2.0 authorization server provides client applications (e.g., medical imaging services) a "secure delegated access token" which permits client

*Address all correspondence to: Weina Ma, E-mail: weina.ma@uoit.ca

2329-4302/2016/$25.00 © 2016 SPIE

applications to access resource owners' (RO) (e.g., patients) resources (e.g., medical imaging documents) if ROs approve the actions performed by the client application. In addition, user managed access (UMA) allows RO to define access control policies at a centralized authorization server to protect his distributed resources.[4] UMA eliminates the need for the RO presence to grant access requests from arbitrary client applications, and the authorization server can make the access decision based on RO defined access control policies. "OpenID Connect" provides an identity layer on top of the OAuth 2.0 protocol. It is a token-based authentication standard that allows client applications to verify the identity of the end-user based on the authentication performed by an authorization server, as well as to obtain basic profile information about the end-user.[5] OpenID Connect has broad support from major cloud service providers, enterprise companies, and social networking companies (e.g., Google, Yahoo, Microsoft, and Facebook). Google+ sign-in also uses OpenID Connect technology. According to the OpenID Foundation, the Department of Health and Human Services of the US Government has joined the OpenID Foundation to create a profile of OpenID Connect and associated projects.[5]

We propose the design of OpenID-Connect-as-a-service (Fig. 1) that provides universal identity management and ease of applying a consolidated authentication mechanism. It also provides advanced authentication technology (e.g., biometrics and hardware authentication devices) to authenticate all users (using traditional desktop or mobile devices) who request to access resources stored in traditional or cloud-based medical imaging systems. OpenID-Connect-as-a-service also provides centralized authorization mechanism by allowing ROs to define access control policies to protect their distributed sensitive data, such as health insurance and mental health records. Canada Health Infoway[6] stated that the healthcare services deployed in a private or community cloud, rather than a public cloud, can provide equivalent security level to traditional computing models. So deploying PACS systems and DI-r's to private or community clouds is the preferred cloud-based medical imaging solution. OpenID-Connect-as-a-service is a simple and standard facility to delegate application login to third-party [OpenID provider (OP)] who continuously invests in advanced authentication techniques. As a case study, we developed a prototype for implementing an authentication and authorization service using OpenID Connect, and then integrated OpenID-Connect-as-a-service with medical imaging services from PACS systems.

The main contributions of this paper can be summarized as follows: (i) designing an SSO service for cloud-based medical imaging systems; (ii) applying open source authentication delegation and customized user attribute claims to feed existing authorization services in medical imaging systems; (iii) providing on-line authorization which allows RO control permissions of access requests from arbitrary client applications to protect their distributed sensitive resources; (iv) providing off-line authorization which allows RO to define a set of consent-based access control policies to eliminate the RO's presence; and (v) designing common OpenID-Connect-as-a-service application program interface (API) to easily integrate with medical imaging service providers and web-based and mobile applications.

The remainder of this paper is organized as followings. Related work is discussed in Sec. 2, and the relevant background technologies are presented in Sec. 3. In Sec. 4, our OpenID-Connect-as-a-service design architecture and workflow are explained. Section 5 is allocated to the case study, and finally the conclusion and a discussion are presented in Sec. 6.

2 Related Work

In this section, we discuss the approaches that are related to our work.

Security assertion markup language (SAML) is a dominant and mature technology for enterprise SSO, which is well supported by the existing enterprise infrastructure. Shibboleth[7] is a standard-based, open source software package which implements SAML protocol to provide a federated SSO and attribute

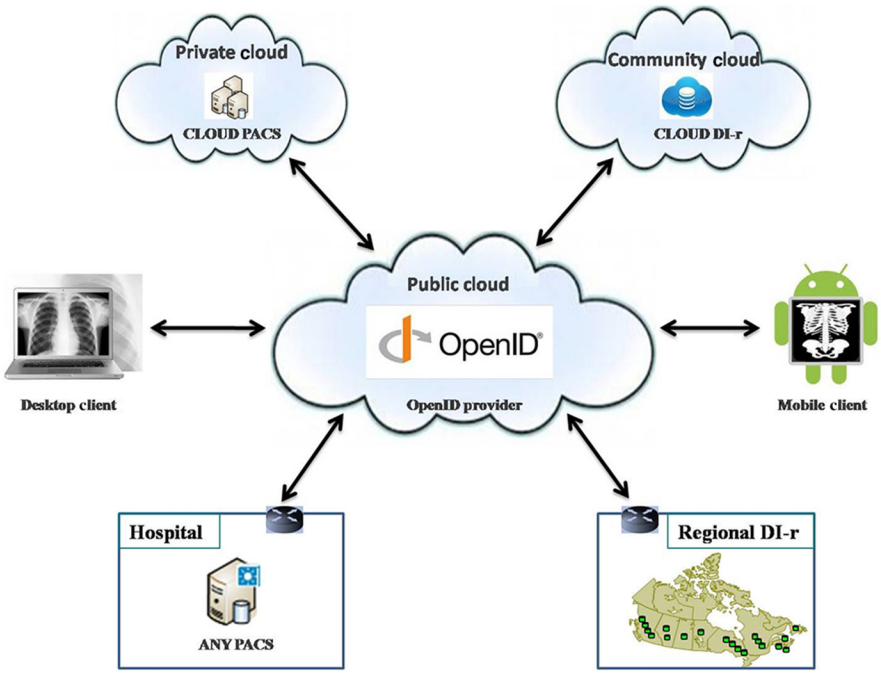

Fig. 1 Overview of OpenID-Connect-as-a-service in cloud-based medical imaging systems.

exchange framework. The main elements of a web-based SSO system are web browser (representing user), resource (with restricted content), identity provider, and service provider (performing SSO process). A user authenticates with her organizational credentials, and the organization (or identity provider) passes the minimal identity information necessary to the service provider to enable an authorization decision. For stand-alone systems, the identity provider and service provider are in the same organization. However, in the case of a "federated SSO," these two systems are not in the same organization. In this case, a given service provider may wish to work with more than one identity provider (e.g., commercial services with multiple customers, resources used by researchers at multiple organizations), and similarly, a given identity provider might want to work with multiple service providers. When a group of identity and service providers agree to work together, this group is called a federation. These cases will be explored in details in Sec. 3.3.

The existing open source cloud solutions have little flexibility in authentication since they are based on proprietary mechanisms. To solve this problem, Khan et al.[8] proposed OpenID-authentication-as-a-service in OpenStack (an open source cloud computing software platform for creating private and public clouds) and designed a decentralized authentication service in OpenStack using OpenID 2.0 as an open authentication platform. Two APIs are defined: authentication API and identity verification API. OpenID 2.0 is an open standard for authentication that is also developed by OpenID Foundation (same with OpenID Connect but completely different technology). However, most of the organizations have started migrating OpenID 2.0 to OpenID Connect. For example, Google has deprecated OpenID 2.0 and shut it down by April 2015. All Google apps using OpenID 2.0 are migrating to OpenID Connect.[9] Cloud Foundry is an open source platform as a service, offering application provisioning and management in a cloud environment.[10] User account and authentication management[11] is the identity management service of cloud foundry, which is also built on top of OAuth 2.0 and OpenID Connect.

Ma and Sartipi[12,13] introduced an agent-based infrastructure for secure medical image sharing between legacy PACS systems, which allows for capturing PACS communication messages and authenticating users against OpenID protocol. Implementing OpenID Connect as an authentication service at the web server of image gateway is not our main target. OpenID Connect also provides claims of attribute about the authenticated users, which can be integrated with the authorization workflow inside medical imaging systems to make access decisions.

Kakizaki and Tsuji[14] proposed a decentralized user attribute information management method using OpenID Connect for identity verification. Some enterprises may prefer to manage user information independently. OpenID Connect identity provider assigns a uniform resource identifier (URI) to each enterprise persisted attribute. After acquiring user's information from the identity provider, the client application is able to discover and retrieve enterprise-specific attributes using attribute URI. This feature caters to the healthcare organizations that have concerns about exposing some sensitive patient information to a third-party.

Fast healthcare interoperability resources (FHIR) is an HTTP-based, resource-oriented RESTful API that supports operations (create, read, update, delete, and search) on resources (clinical, administrative, financial, and infrastructure). FHIR is not a security protocol and does not define any security functionality. However, it can be used with different standard security protocols. OpenID Connect protocol can provide security for FHIR resources where an RO could grant the client application a limited access to a protected FHIR resource, which is determined by the scopes of access.[15,16] This method allows more control and protection over the FHIR resource. In OAuth, scopes define individual pieces of authority that can be requested by clients, granted by ROs, and enforced by resource servers (RSs). A scope is defined as: "scope: = permission/resource.access"[16] where permission represents either a single patient record (i.e., patient) or a group of patient records that a specific user (e.g., healthcare provider) has permission to access (i.e., user). "Resource" represents the kind of FHIR resource (e.g., MedicationOrder, observation, appointment, *). Two examples are:[16] (i) patient/MedicationDispense.write: read and write access to the supply of medications for a single patient; and (ii) user/observation.*: full access to all authorized observations performed by a healthcare provider.

3 Background

In this section, we provide an overview and comparison of SSO solutions and discuss the best-practice of identity management, authentication, and authorization frameworks in different scenarios.

3.1 *Single Sign-on Solutions*

Table 1 presents an overview and comparison of three widely used SSO solutions. SAML[17] is an XML-based, open-standard SSO solution. SAML is an agreement between enterprise systems to share information about who a user is and what attributes the user has. SAML relies on an explicit trust relationship between service provider and identity provider, which means the selected identity providers have to be coded in advance into service providers. SAML is a complicated and expensive

Table 1 An overview and comparison of SSO solutions.

Standard	SAML 2.0	OAuth 2.0	OpenID Connect 1.0
Introduced	2008	2012	2014
Object	XML	JavaScript object notation (JSON)	JSON
Web service	SOAP	representational state transfer (REST)	REST
Open standard	√	√	√
Authentication	√		√
Attribute claim	√		√
Authorization		√	√
Dynamic client registration		√	√
Support mobile app		√	√
Costly to implement	√		

protocol to implement. Historical evidence suggests that only large enterprises can justify going through a costly SAML implementation.

OAuth 2.0[3] is an open protocol to allow secure authorization from web, mobile, and desktop applications. OAuth provides client application an access token, an encoded string containing scope, lifetime, and other access attributes, for granting limited access to a protected resource on behalf of the RO without sharing credentials, such as a password. UMA[18] is an enhancement of OAuth 2.0 which allows an individual to define access control policies to protect his/her sensitive resources. Integrating OAuth 2.0 and UMA into e-health systems, patients can determine precise and customized access rules on their medical data. OAuth 2.0 authorization server governs access both online and offline: the patient can grant or deny access requests with presence; authorization server determines the access permission based on predefined policies if the patient is offline.

OpenID Connect[4] adds an open and decentralized authentication layer on top of OAuth 2.0 that provides a way to verify a user for cooperating sites without sharing user credentials. Unlike SAML, OpenID Connect provides an identity provider discovery protocol which dynamically discovers the corresponding identity provider once a user unique ID is given. Apart from identity verification, OpenID Connect allows service providers to use more extensible features, such as encryption of identity data, dynamic discovery of identity provider, session management, and to obtain user attributes after authentication.[5]

3.2 *Best Practices Under Different Scenarios*

This section indicates four typical scenarios of medical imaging systems and the best practice of identity frameworks under different scenarios. The consumer service can be a desktop application client or a mobile client; the access request for the protected resource is retrieved from a web service; identity provider is responsible for identity verification and issuing a dedicated security token; the user directory stores user attributes.

a. Direct authentication of single domain. Figure 2 shows the direct authentication pattern[4] of a single domain. The protected resources can be secured with HTTP basic/digest authentication by sending the username and password in an HTTP header. Plaintext username and password may be leaked while in transit, so the communication with remote users who sign in from outside a firewall must use an encrypted channel. The resource domain needs to own and maintain the identity management, authentication, and access control services. Direct authentication does not work in a federated scenario where the resource domain wants to give access to the users owned by other domains. This is the case for the legacy PACS system internal infrastructure.

b. Trust model across multiple domains. With the increasing demand of medical image sharing, legacy PACS systems are interconnected across a distributed environment through a trust model as shown in Fig. 3. Different from password-based direct authentication, certificate-based authentication is used to recognize services to be trusted across several domains. Once the validation process in SSL mutual authentication is completed, both consumer service and web service check whether the certificate authority (CA) that signed the certificate is trusted. If the certificate is from a trusted CA, the web service will let the user in. Each domain is responsible for ensuring the restricted resource is adequately protected. A key challenge of this trusted model is the lack of federated capabilities. Managing and authenticating users locally imposes a significant administrative burden to ensure that persons are uniformly identified in each system.

c. Cross-enterprise user assertion using SAML. In Fig. 4, cross-enterprise user assertion (XUA) provides identity federation by using external centralized identity provider to generate identity claims that are communicated across enterprise boundaries.[19] X-user assertion (typically relies on SAML assertion) plays the role of security token and carries identities and additional user attributes, such as user address, role, and preferences. X-user assertion feeds access control

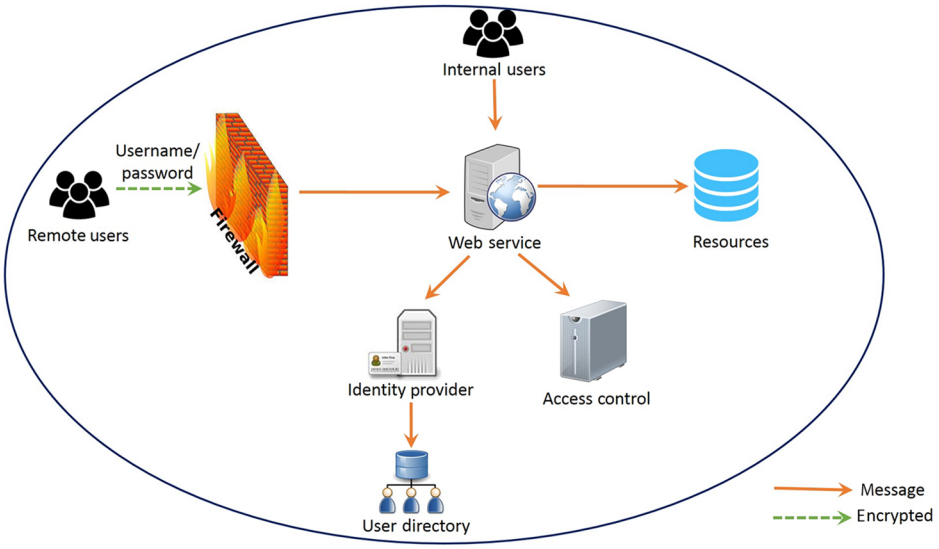

Fig. 2 Direct authentication of single domain.

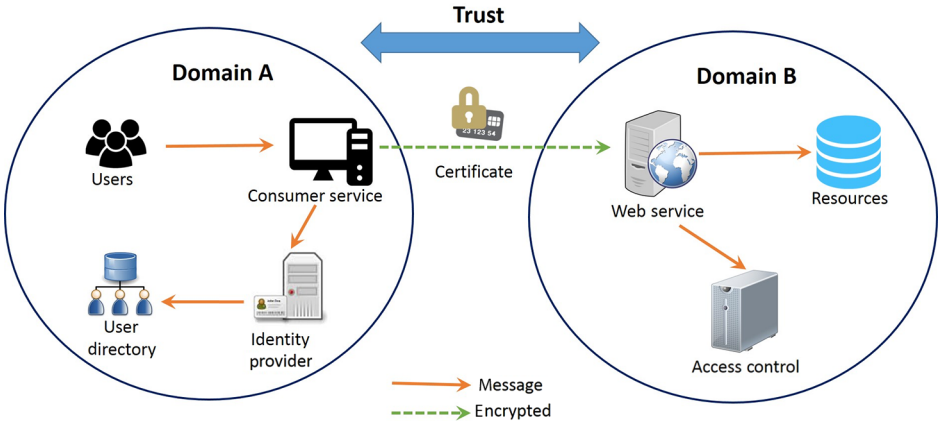

Fig. 3 Trust model across multiple domains.

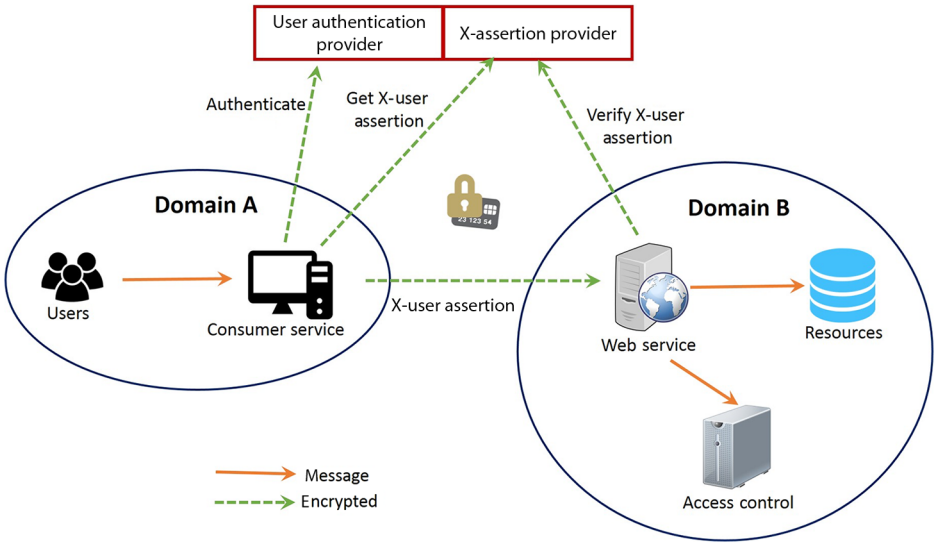

Fig. 4 XUA using SAML.

policy enforcement in a resource domain to determine access permission. The participating enterprises that are unwilling to expose the sensitive user information to third-party identity providers can have their own user directory. However, access control policies are local to each system. Hence, ensuring the consistency of access control rules across all domains has to be managed manually. Moreover, patient consent directives and their impact on access control are communicated neither electronically nor automatically to each domain.

d. Authentication and authorization using OpenID Connect. Figure 5 shows the user-centric model using OpenID Connect, which allows an RO to manage his own identities and to define access control policies at a centralized authorization server. User registers to a centralized authentication provider and gets a unique identifier; a user can use this identifier to login to different consumer services; authentication provider issues an ID token to consumer service if the user is authenticated successfully without credential disclosure. If the user is requesting his own resources, authorization server will ask user to control the permissions of consumer service. Also, a user may define consent policies at a centralized authorization server to control his/her distributed health records without his presence. All healthcare services that access a patient's protected resources are controlled by predefined consent policies. Authorization server evaluates the access control policies and issues consumer service an access token as a grant. User-centric model not only frees healthcare enterprises from administrative burden to identity and policy management, but also gives individuals low barriers and comfort of entry to healthcare services.

3.3 *Performance Comparison*

Table 2 presents the performance overview and comparison of four security architectures in Figs. 2–5 based on different delays caused by additional communications in the advanced security protocols. The performance that the users experience varies depending on many factors, including throughput of the network, the processed workload, the I/O and storage configuration, and the design and development decisions. We compare the performances of these architectures considering the

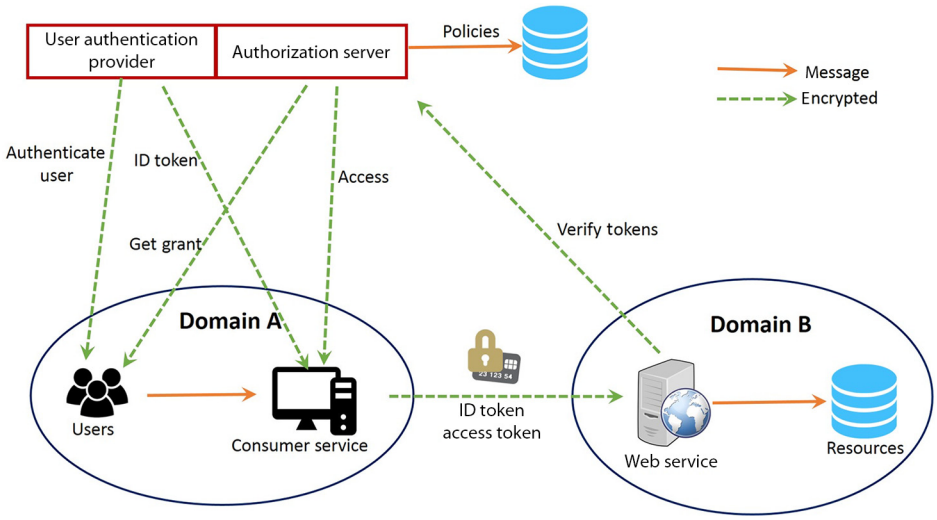

Fig. 5 User-centric authentication and authorization using OpenID connect.

Table 2 Performance overview and comparison of different security models.

Standard	Direct authentication Fig. 2	Trust model Fig. 3	Cross-enterprise using SAML Fig. 4	User-centric using OpenID connect Fig. 5
Authentication	Local	Local	Remote	Remote
Authorization	Local	Local	Remote	Remote
Encryption	Yes	Yes	Yes	Yes
Signing	No	Yes	Yes	Yes
Certification trust	No	Yes	Yes	Yes
Attribute provider	Local	Local	Requires further step to populate assertion	UserInfo endpoint provides claims
Discovery service	No	No	No	Dynamically discover identity provider
Token format	No	No	XML	JSON
Transport	TCP/IP	TCP/IP	SOAP	REST

locations of actors that are involved in authentication and authorization flow, the additional processes caused by the protocol, and protocol data format.

First, adopting the architecture of SAML or OpenID Connect requires that the consumer service, identity provider, and service provider be located in different domains, and consequently increases the communication latency. However, deploying identity providers into distributed regions to server requests from local regions might decrease the communication latency. Second, SAML and OpenID Connect introduce extra steps into authentication and authorization flow, such as populating assertions, acquiring claims, and dynamically discovering identity provider, which may cause the identity provider to suffer from a heavy workload. Deploying identity provider as a cluster helps to increase service availability and scalability. For example, a load-balancer is required to ensure that user login requests can be evenly dispatched to different instances of identity provider. At the same time, autoscaling helps maintain availability with reasonable cost by dynamically scaling identity provider cluster capacity according to the amount of workload. Moreover, SAML is an XML-based protocol and OpenID Connect is a JOSN-based protocol. JSON is more compact and it can be faster in transition because less data is transferred.

4 Approach

OpenID Connect works with the existing standard Internet browsers without requiring any client-software so that the users, physicians, and patients can set up their devices and applications independently to access medical imaging documents from anywhere. The design of OpenID-Connect-as-a-service is open to integrate with existing healthcare services and resources.

4.1 *Architecture*

OpenID-Connect-as-a-service is a cloud service to provide user-centric authentication and authorization, which allows ROs to have more control and flexibility to protect their distributed resources. Figure 6 presents the OpenID-Connect-as-a-service architecture. Electronic medical record is an individual's health-related information (e.g., examination, laboratory test, allergies, medication history, and claims) from family doctors, hospital patient's health record in some scenarios: physician needs to access patient's diagnostic, treatment, and/or care information; tax institution needs to access to an individual's benefit plan, claims, and personal information; healthcare researchers want to query and view a patient's medication and treatment history for a specific disease. Uniformly identifying individuals by each healthcare service is required.

To protect a patient's privacy, clinicians normally asks the patient to sign a consent form to specify who they want their personal health information shared with. However, paper-based consent for disclosure is not uniform to all patients. Patients cannot flexibly express their particular requirements. Furthermore, paper-based consent and their impact on access control policies can neither be electronically nor automatically communicated between enterprises. OpenID-Connect-as-a-service solves these problems by providing user-centric authentication and authorization: an individual registers at OpenID-Connect-as-a-service to acquire a unique identifier which is used to login to any service; the individual defines access control policies at OpenID-Connect-as-a-service to control his/her resources which reside on any RS to be accessed from arbitrary health services.

4.2 *Security Considerations in Workflow*

OpenID-Connect-as-a-service workflow defines five roles as follows:

- End user (EU): is a human participant who wants to access the service (e.g., physicians or patients who request to access images from PACS system or DI-r services).
- Resource owner: is capable of granting access to a protected resource (e.g., a patient may grant a physician or a healthcare organization to access his/her medical images).
- Relying party (RP): is application that requires authentication and access grant from OP (e.g., PACS system and DI-r services).
- Resource server: manages resources (e.g., medical images and reports) and their metadata.
- OpenID provider: is an authorization server, i.e., responsible for issuing tokens to RP after successfully authenticating the EU and obtaining granting authority from RO.

Assume a patient (RO) has taken medical exams, and his images and reports are stored at RS (e.g., PACS database). The patient also defined customized consent policies that specified who he wants his information shared with. In the case where EU and RO are not the same person, RO may be off-line and hence unable to grant or deny the access request. So patient consent-based access control policies must be defined and integrated with the OpenID connect process flow as shown in Fig. 7.

(1) Initiate access request: suppose EU has already registered an account to OP. Then he/she initiates an access request and provides his/her OpenID identifier to the medical imaging service (RP). Username and password are only persisted at OP without disclosure to any RP. However, a malicious RP could attempt to phish EU passwords by presenting its own interface instead of a trusted system, so all information in the authentication exchange including username and

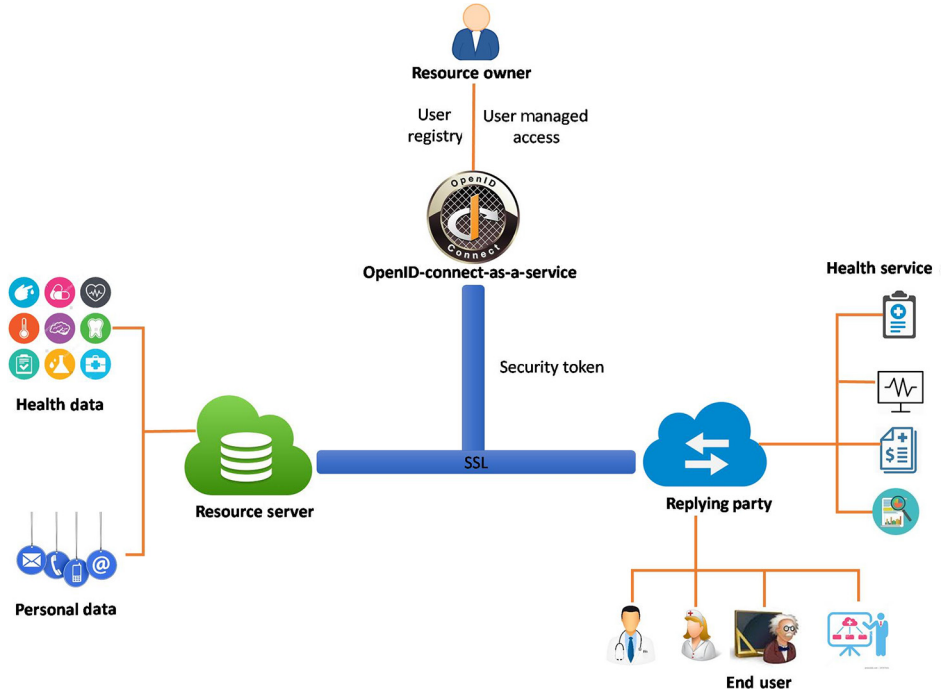

Fig. 6 OpenID-Connect-as-a-service architecture.

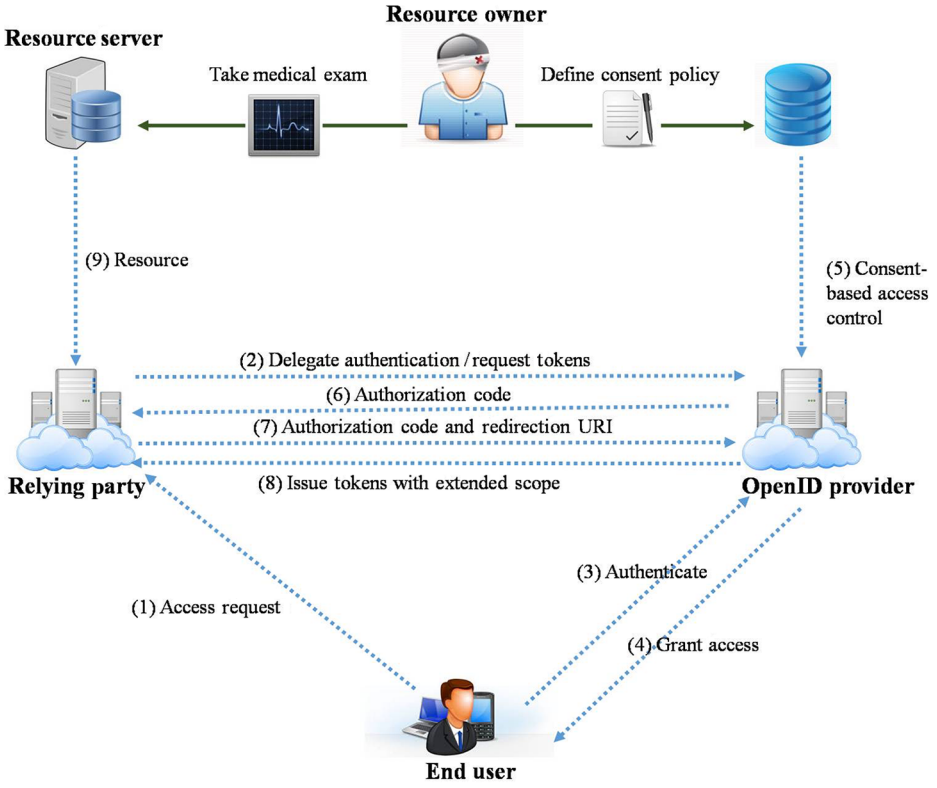

Fig. 7 OpenID-Connect-as-a-service process flow.

password can be captured.[20] EU should be educated about phishing attacks, e.g., only accessing trusted RP.

(2) Delegate authentication: RP can dynamically discover the location of corresponding OP according to the URL "http://example.com." Then RP delegates OP to authenticate EU and asks for an access token if the access is granted. A malicious RP can impersonate another RP and obtain access to a protected resource.[21] OP must authenticate RP whenever RP registers and delegates authentication to OP.

(3) EU authentication: OP redirects EU's browser to an OP login page to perform authentication. Besides a username and password, a strong authentication method can be used (e.g., information card and biometrics). OP should not process repeated requests without authenticating RP and EU to ensure that the repeated request comes from the original RP and not an impersonator.

(4) Grant access: if EU and RO are the same individual, OP provides EU with information about RP and the requested access permission. EU reviews the information to grant or deny the access request and limits the access scope and lifetime. In some cases, RP might acquire excessive privileges because EU may not understand the scope of the access being granted and to whom. OP must explain the required scope in a clear and accurate way, then EU should only grant the minimal scope necessary to RP. EU engagement in authorization also assists OP in identifying an impersonal RP.

(5) Apply consent policies: after authentication, OP evaluates the access request against RO predefined consent policies. If RO and EU are not the same person, consent policies are mandatory because RO is not run-time present and step (4) will be ignored.

(6) and (7) Authorization code: assuming RO grants the access request, OP redirects EU's browser back to RP and returns RP a short lived and single-use authorization code. OP authenticates RP by ensuring the authorization code was issued to the same RP.

(8) Issue tokens with scope: if EU is authenticated by OP, an ID token is issued to RP; if RP is authorized by RO to access the resource, an access token is returned to RP. An ID or access token is an encoded JSON string that is digitally signed using a secret key. An access token is used to access a protected resource within the scope of a period of time. The string is usually opaque to the RP. ID token and access token must be kept confidential both in transit and storage. Attackers might obtain ID token and access token from the OP database if OP persists such sensitive information in a database, which might cause a disaster as all tokens could be disclosed. So only token hashes should be stored, and OP has to enforce the database security.[20]

(9) Retrieve resource: OP must ensure that the ID token and access token cannot be generated, modified, or guessed by unauthorized parties. Access token must be presented to RS and RS validates the access token from OP before returning the demanded resource to RP, which is consequently returned to EU.

4.3 *Authentication and Authorization Flow*

In addition to being user-centric consent-based access control, OpenID-Connect-as-a-service can be integrated with existing

authorization workflow inside medical imaging systems to enforce enterprise-specific access control policies. In the architecture of integrating OP with medical imaging services, we define two services as follows:

- AuthN service: this service implements the "OpenID Connect authentication" using two operations: (i) authentication request from medical imaging services; and (ii) user information query from authorization service.
- AuthZ service: this service represents the authorization service in the existing medical imaging systems. In this setting, AuthN service provides the attributes of authenticated users which feed AuthZ service to make access control decisions.

Taking web access to DICOM persistent object (WADO) service as an example, Fig. 8 shows the sequence of messages through which the user uses a browser to access digital imaging and communications in medicine (DICOM) images stored in the PACS system. This is done through the following steps:

Step 1 (Initiate access request): first the user enters the URL of WADO service in the browser. WADO service receives an HTTP request and invokes an authentication operation from AuthN Service. AuthN service redirects the user's browser to an SSO page and asks for an OpenID unique identifier (simply called "OpenID identifier"). An OpenID identifier can be generated in the form of an e-mail address or in URL syntax. For example, an OpenID identifier may look like "myname@example.com" or "http://example.com/myname."

Step 2 (Delegate authentication and request token): AuthN service can dynamically discover the OP (simply called "OP") location using the endpoint URL http://example.com/. In order to utilize OP services, AuthN service needs to obtain the OP's configuration metadata and register with OP as a client.

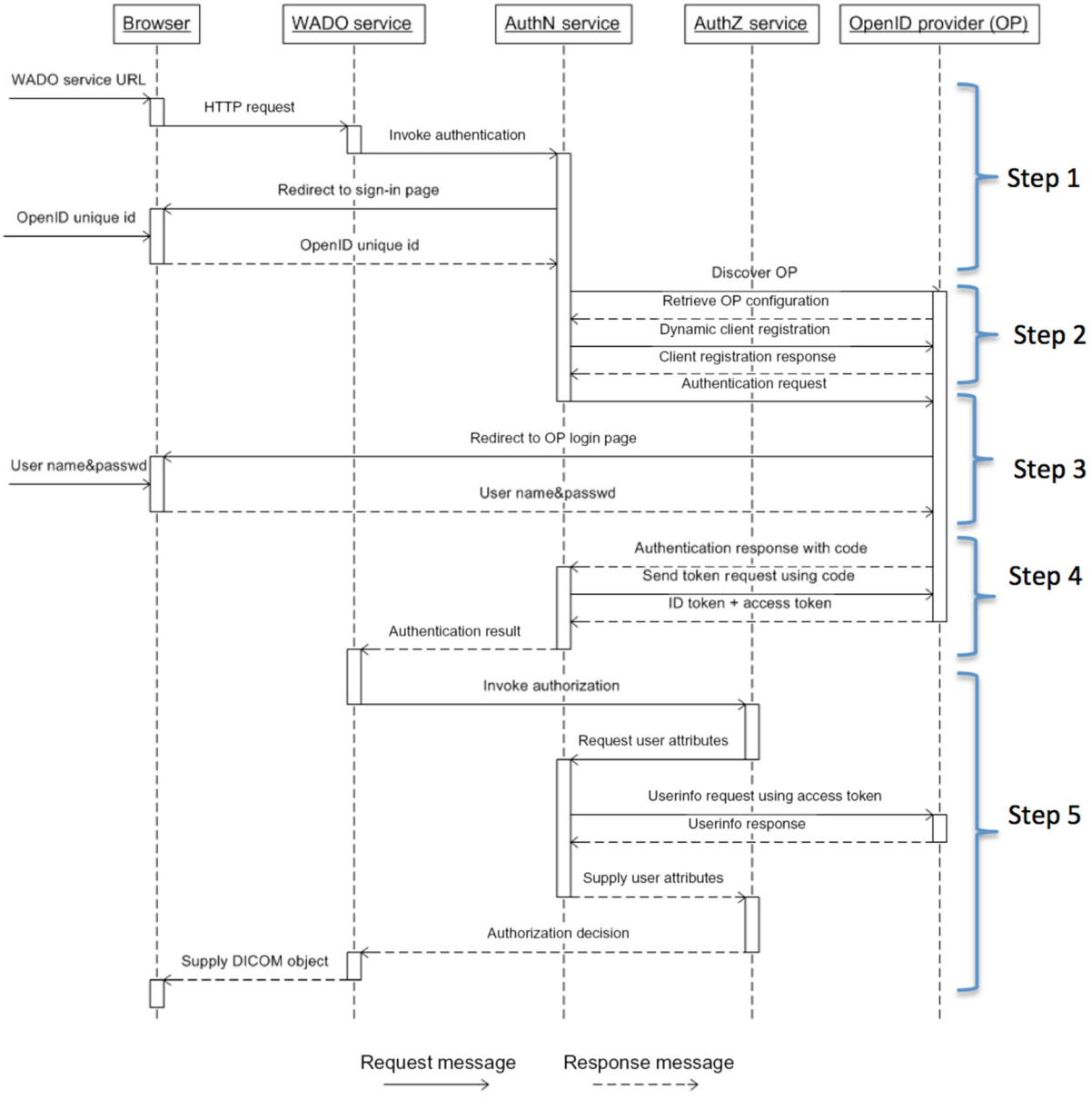

Fig. 8 The sequence for authentication workflow using OpenID-Connect-as-a-services.

Step 3 (Authentication): after registration, AuthN service sends an authentication request containing the desired request parameters to OP. OP redirects the user's browser to a login page to authenticate the user. Any method to authenticate the user can be used (e.g., password, credentials, information card, and biometrics). After authentication, OP evaluates this access request against the defined patient (RO) consent policies as presented in Sec. 4.5.

Step 4 (Issue tokens with extended scope): after a successful authentication response, OP returns an authorization code to AuthN service. ID token (JSON web token that contains claims about the authentication event) and access token (JSON web token that represents the responding authorization grant) are what we need to access the protected resource. Authorization code is a temporary credential that makes sure ID token and access token can be sent on a secured connection. Once an authorization code is obtained, AuthN service can use this code to obtain an ID token and an access token. At this point, the authentication process that was started in step 3 is completed.

Step 5 (Access control decision): WADO service invokes AuthZ service to obtain an authorization decision. AuthZ service invokes AuthN service to receive the required user attributes. Using access token obtained through authentication flow, AuthN service makes a request to OP for attribute claims (normally represented by a JSON object that contains a collection of name and value pairs) of the authenticated user, such as user's full name, email address, health card number, and picture. Such user attributes feed AuthZ service to make authorization decisions. Once the access request is granted by AuthZ service, the DICOM object can be viewed in the browser.

AuthN service is in charge of interactions with OP, and managing (obtaining, verification, and refreshing) ID tokens and access tokens. OpenID-Connect-as-a-service design separates authentication from authorization and provides common and easy APIs for integration. The existing medical imaging services only need to call AuthN service API to do authentication operation, and AuthZ service in medical imaging systems needs to call AuthN service API to retrieve user related information.

4.4 *Multifactor Authentication*

New threats, risks, and vulnerabilities around cloud applications and mobile devices underscore the need for a strong authentication approach based on simple service delivery. Multifactor authentication[22] is a method of authenticating a user by successfully presenting several separate authentication stages. The more factors that are used to ensuring that a user is who he claims to be, the greater the trust of authentication will be. Knowledge factor is the most commonly used authentication factor that can be a password or a personal identity number. Possession factor usually depends on the user's Internet device (e.g., smart phone and laptop) as "something only the user has" for authentication, eliminating the need to deploy additional hardware to users. Inherent factor is associated with users themselves, by using biometric methods for authentication, such as fingerprint readers and voice recognition. A typical two-factor authentication service might require a user to sign in with a user name and password, and then enter a code generated by a smart phone app or text message.

OpenID connect supports integration with multifactor authentication methods, which enables medical imaging service developers to build a simple authentication process by outsourcing SSO and strong authentication techniques to identity providers that specialize in the security and privacy protection field. In authentication request, OpenID connect protocol defined a parameter named authentication context class reference (ACR). RP might set ACR that OP requested to use for processing the authentication request. OP may ask EU to re-authenticate with additional factors for meeting the ACR requirement. OpenID Connect specifies the multifactor authentication process flow which makes it easy to integrate with a strong authentication method. Multifactor authentication is already implemented in production operations at some OpenID connect identity providers.[5]

4.5 *Consent-Based Access Control*

We summarized several existing paper-based patient consent disclosure forms used in Canadian hospitals[23–26] and produced a UML class diagram, as shown in Fig. 9, to model privacy-based access control policies. A consent-based policy is associated with the patient who defined the policy; the health and personal information that are allowed or denied to share with others; effective period that the policy is in effect; the disclosure type of how to share the author's information; purpose of disclosure; and the specific context to allow or deny disclosure.

OpenID Connect is based on OAuth 2.0, which defines a mechanism to allow RO to delegate access to his/her protected resources with limited scopes. Along with access token, OpenID connect uses parameter "scope" values to specify what access privileges are being granted. A healthcare service provides "scope" of the access request through a request parameter; in turn, the authorization server dispatches access token together with "scope" as response parameter to inform the healthcare service of the constrains of the issued access token. We call them "requested scope" and "issued scope," respectively.[27] However, the scope is simply expressed as a list of strings, which is not adequate to describe complicated consent policies both in syntax and in semantics. Totally, around 20 attributes about a person's birth date, contact address, picture, and so on are defined in OAuth 2.0 protocol. A healthcare relationship trust profile for FHIR enhances OAuth 2.0 scopes in the format of "scope: = permission/resource.access," where "permission" indicates whether the access request is for a single patient record or a set of patient records; "resource" indicates an FHIR resource, such as patient, medication order, observation, appointment, and so on. However, their proposed schema still cannot describe the complex patient consent policy defined in Fig. 9, including access context, access purpose, disclosure type, and effective period. Accordingly, instead of inventing some totally new policy definition and enforcement mechanism, we aim at exploring the possibilities of expressing the scope of access token by using extensible access control markup language (XACML) represented in JSON, and then evaluating the access token based on existing XACML access control infrastructure.[28]

XACML is an extremely fine grained policy language framework. With the adoption of JSON, XACML has a good chance of describing complicated consent policies for OpenID Connect. We reuse the XACML model to provide applicable policies to authorization server, and to evaluate issued access token.

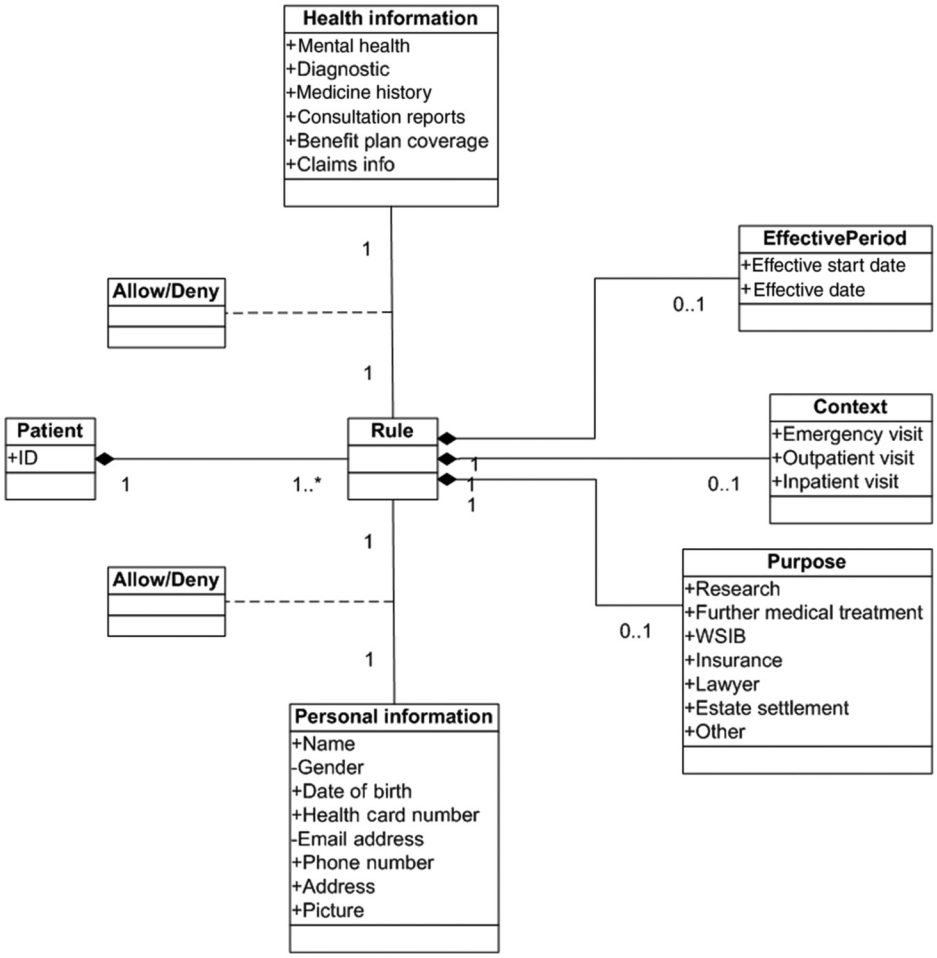

Fig. 9 Electronic patient consent class diagram.

Figure 10(a) shows the mechanism of employing XACML policy decision point (PDP) to make access decisions according to an access token issued by an authorization server. XACML request designates the actual actions on the resource that are limited by the scope of the access token. Policy information point provides environment attributes—the variable factors of the scope of access token—to assist PDP in decision making. The XACML response either grants or denies the request. An example of XACML request and consent policy presented in JSON is shown in Figs. 10(b) and 10(c). XACML 3.0 has a working draft about the request and response interface based on JSON and HTTP;[29] nevertheless, JSON representation of XACML policies must be an alternative to current XML representation in the near future.

5 Prototype Implementation

OpenID Connect uses REST/JSON message flows, which are easy for developers to integrate, compared to preceding federated identity protocols. JSON is a simple text-based message format that is often used with REST web services. Open source tool "PYOIDC" is a complete implementation of OpenID Connect specifications written in Python.[30] It is developed and maintained by members of the OpenID community. Open source tool CherryWado is selected to simulate a WADO server which is also written in Python.[31] CherryWado uses web server gateway interface (WSGI) server to implement Python web applications. DICOM images are handled by Python imaging library and provides powerful image processing and graphics capabilities. The WADO service does not make any assumption about how/where the DICOM files are stored. We implemented a DICOM access layer that specifies a way to retrieve the corresponding DICOM files from DICOM image repositories. Figure 11 indicates the class diagram of our prototype implementation.

The class WADO provides a mechanism for accessing DICOM objects (e.g., medical images and medical image reports) through HTTP/HTTPS protocols using DICOM UIDs as query parameters. By using Python web application framework WSGI, the access requests are routed to WADO service based on the target URL. Once an access request arrives, WADO first invokes the method "authenticate()" of AuthN service to identify and verify the user, then invokes the method "authorize()" of AuthZ service to make authorization decision if the user is authenticated successfully. After obtaining a grant, WADO retrieves the DICOM file from DICOM image repository using "DICOM access layer" as an intermediary. At last, the DICOM file is converted to a more compatible format, such as JPEG through class Image, and is displayed in the browser.

AuthN service exposes an API "authenticate()" to WADO service to do authentication, and an API "getAttrs()" to AuthZ service to retrieve user related attributes. To set up an OpenID Connect relaying party (RP, which is a service provider requiring user authentication and claims from OP), a client API layer

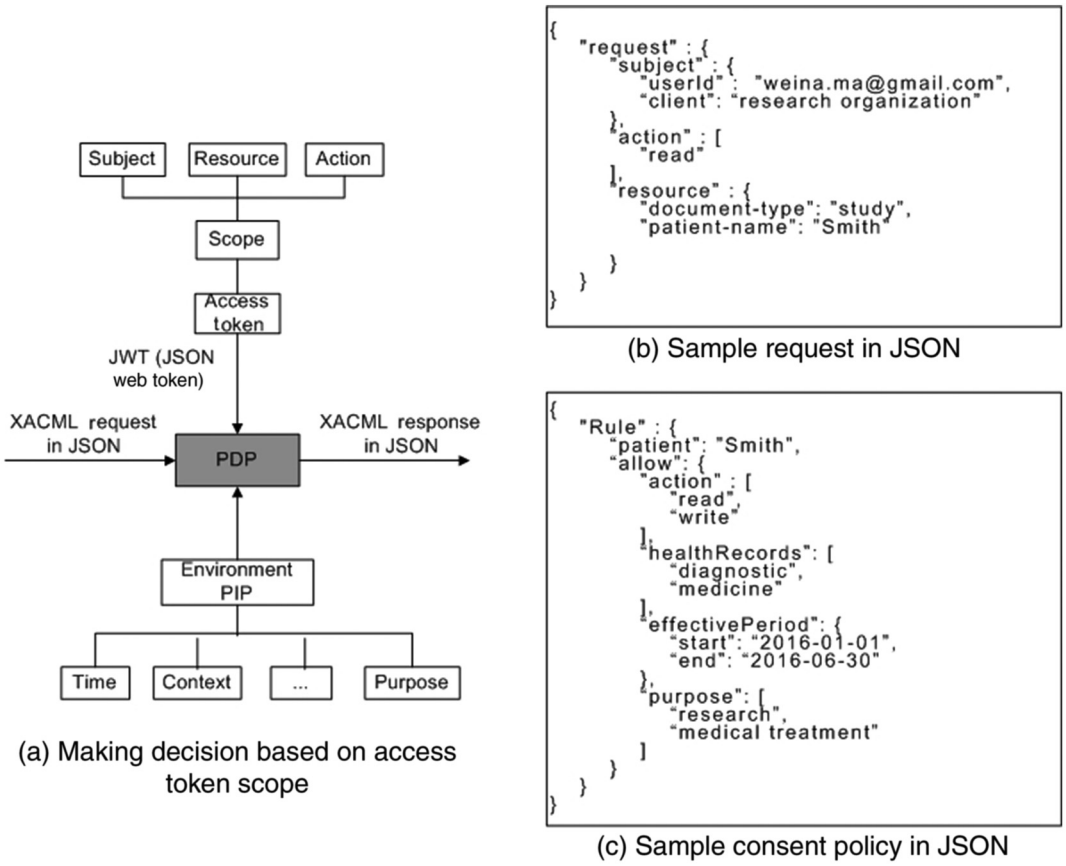

Fig. 10. Consent-based access control enforcement.

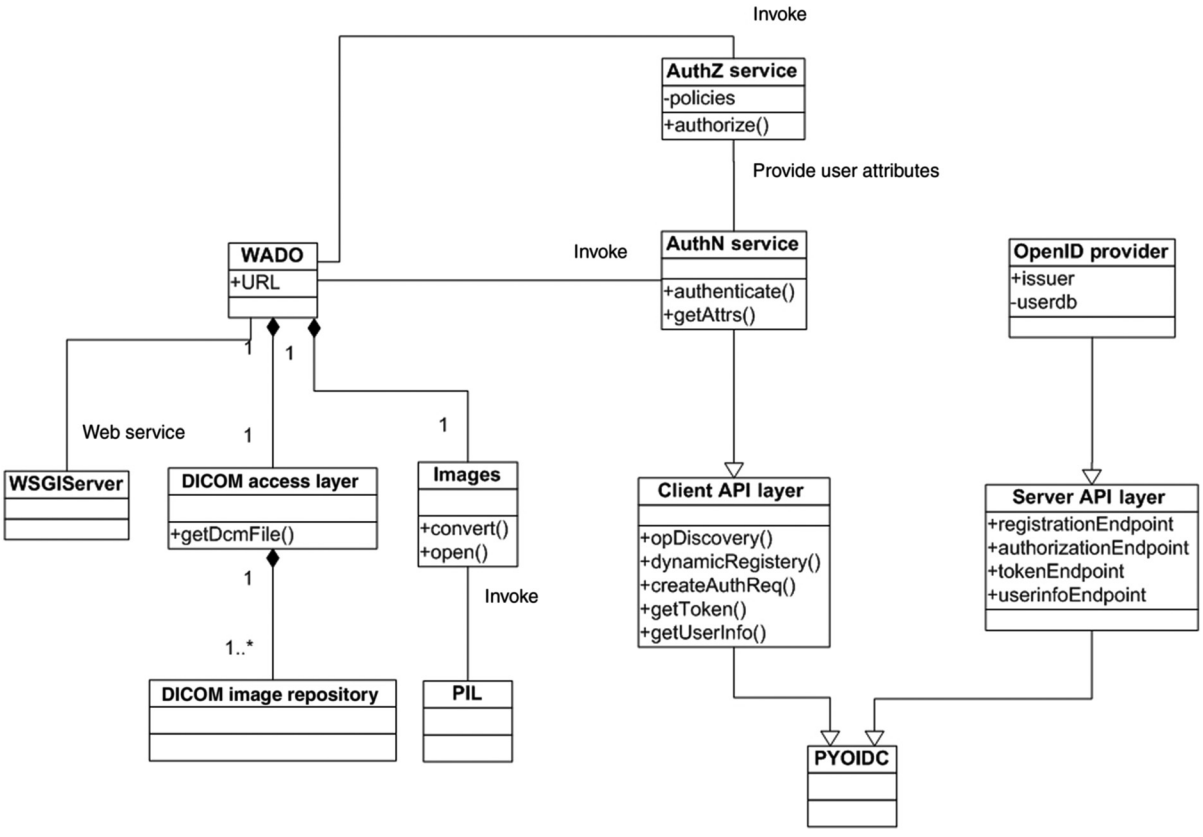

Fig. 11 The class diagram describes the structure of prototype implementation.

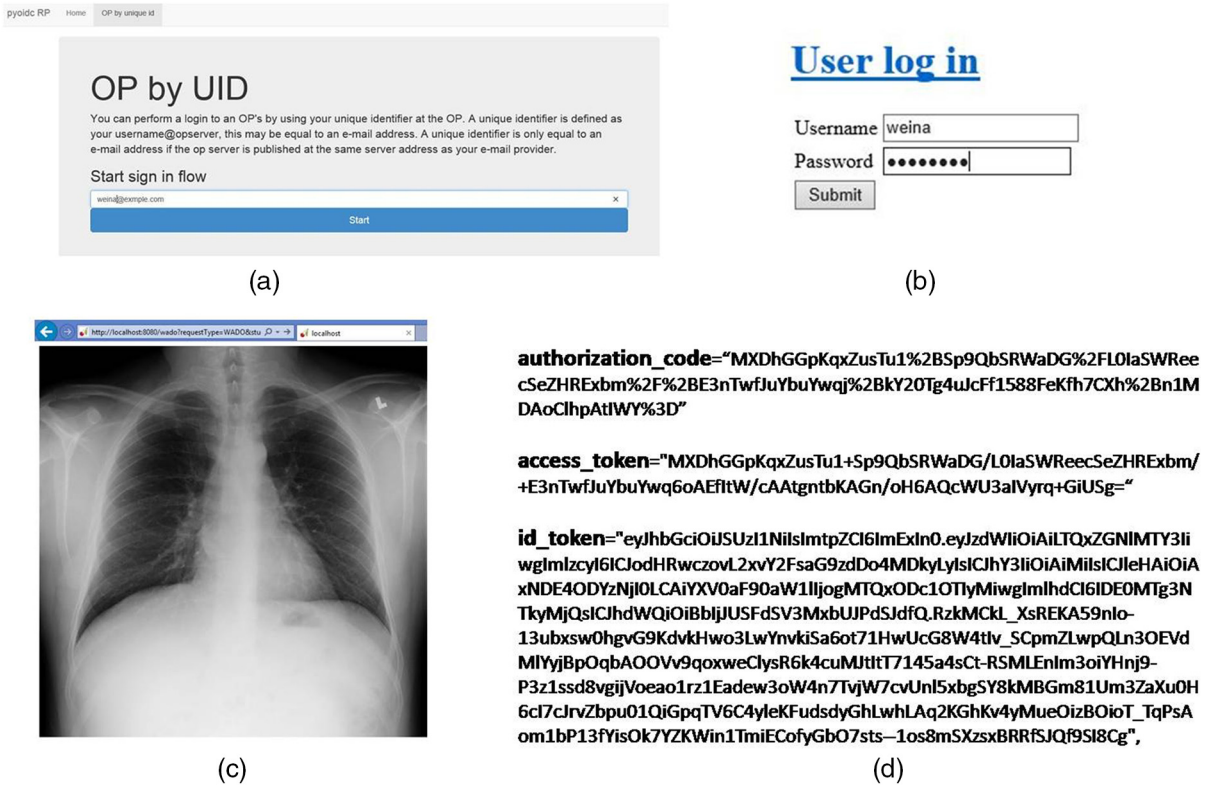

Fig. 12 Experimental result: (a) ask user to input an OpenID unique identifier; (b) user login page; (c) the diagnostic image is displayed in the browser; (d) a snapshot of authorization code, access token, and ID.

is implemented that inherits the functionalities provided by PYOIDC. The method "opDiscovery()" is used to find the location of the desired OP, and method "dynamicRegistery()" registers RP to OP as a client on behalf of the AuthN service. The methods "createAuthReq()," "getToken()," and "getUserInfo()" implement the interactions with OP following the process presented in Sec. 4.

Through a new layer "Server API layer," an OP is implemented using the functionalities provided by PYOIDC. The class member "RegistrationEndpoint" allows a client to register at OP; "authorizationEndpoint" is in charge of authenticating users and issues authorization code to client; "tokenEndpoint" issues ID token and access token for authenticated users; and "userinfoEndpoint" returns attributes about the authenticated user when access token is presented. The class member of OP "issuer" enables determining the location of the OP; a "userdb" that persists and manages user information, centralized or decentralized, can be integrated with OP.

A typical authorization service, AuthZ service, makes determination for an access request based on available information (e.g., user attributes and protected resource attributes) and applicable security policies. The user-related information could be retrieved from AuthN service from OP.

6 Experimental Result

The following is the experimental result of the implementation prototype presented in Sec. 5. A user account is predefined in OP containing: OpenID identifier "weina@example.com," username and password, and user attributes such as address and phone number. An access control policy is created at AuthZ service: "only users living in a specific community can access images from a specific DICOM repository."

First, the user wants to view an image stored in DICOM repository through a browser in the format of JPEG. She enters the WADO service URL, DICOM study, series and object UIDs, and content format in the browser that may look like: Ref. 32.

WADO service receives the access request and asks for AuthN service to do authentication. Figure 12(a) shows a redirected page asking for the user to enter OpenID identifier. A unique identifier is defined as username@opserver. An email address is entered which includes the unique account name "weina" and OP host "example.com." AuthN service is able to find the location of OP using "example.com." OP redirects to a user login page and needs user input username and password as shown in Fig. 12(b). After authentication and authorization are completed, the DICOM image is retrieved from DICOM file repository and displayed in browser in the format of JPEG, Fig. 12(c). The authorization code, access token, and ID token, exchanged between AuthN service and OP, are shown in Fig. 12(d).

7 Conclusion and Discussion

In this paper, we introduced OpenID-Connect-as-a-service to provide a user-centric SSO solution in the cloud-based medical imaging systems. It allows the user to use one account to sign in to multiple healthcare services without exposing a password to these services. OpenID Connect is a decentralized open authentication protocol and provides REST-based APIs that allows different types of healthcare services, including web-based and mobile applications, to delegate OP to do authentication. The

design of AuthN service is flexible, scalable, easy to integrate, and provides several options to integrated systems. In the case of the healthcare organizations with concerns about exploring some sensitive patient information to a third-party, these organizations can manage their own user information but just outsource the identity verification to OP. In addition to user verification, our proposed AuthN service is able to provide user attribute claims to feed existing authorization services in the integrated systems. Moreover, OpenID Connect is open to use any modern authentication technology, such as smart card and biometrics, which offer the healthcare service providers easier and faster access to the advanced identity management with lower investment. We also proposed a UML model for electronic patient consent representation to provide users the flexibility of defining their own policies at a centralized authorization server. Instead of inventing some totally new policy definition and enforcement mechanism, the consent policies are capable of integration with existing authorization infrastructure XACML to perform policy enforcement. This research attempts to provide a design for common authentication services in cloud-based medical imaging ecosystem and the implemented prototype proves the feasibility of the design.

OAuth 2.0 and OpenID Connect are a set of open standards rather than an end-to-end solution, and more and more extensions are being added to enhance the standard. The integrating systems need their own implementation and make sure the implementations are compatible with each other. One challenge of applying our proposed model into practice is that medical imaging systems such as PACS do not provide any external interfaces that allow systems to interact with or impose on the PACS any form of authentication and authorization. To solve this issue, we have proposed an agent-based infrastructure for secure medical imaging system integration.[11] Furthermore, different from cloud computing providers, medical imaging vendors are very slow to change their products, especially for local country specific features. As such, if medical imaging systems are to move from a trusted security model to an integrated security model, integration into current medical imaging system workflow and achieving the desired level of security must be required.

Acknowledgments

This research was conducted with collaboration of Dr. David Koff and Dr. Peter Bak at MIIRCAM Centre of McMaster University. This research was funded by an ORF grant for the project "Secure Intelligent Content Delivery System for Timely Delivery of Large Data Sets in a Regional/National Electronic Health Record."

References

1. B. F. Branstetter, Ed., *Practical Imaging Informatics: Foundations and Applications for PACS Professionals*, pp. 33–47, Springer (2009).
2. A. Gauvin, "Status of diagnostic imaging repository (DI-r) projects across Canada," 2010, http://www.camrt.ca/ (31 May 2016).
3. OAuth Authorization Framework, http://oauth.net/ (31 May 2016).
4. P. Siriwardena, *Advanced API Security: Securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE*, Apress (2014).
5. OpenID Connect Website, 2016, http://openid.net/ (31 May 2016).
6. Canada Health Infoway, "Cloud computing in health white paper," 2012, https://www.infoway-inforoute.ca/ (31 May 2016).
7. Shibboleth Web Single Sign-on, 2000, https://shibboleth.net (31 May 2016).
8. R. H. Khan, J. Ylitalo, and A. S. Ahmed, "OpenID authentication as a service in OpenStack," in *7th Int. Conf. Information Assurance and Security (IAS),* pp. 372–377, IEEE (2011).
9. Google Identity Platform, "Migrating from OpenID 2.0 to OpenID connect, 2016, https://developers.google.com/identity/protocols/OpenID2Migration (31 May 2016).
10. M. Sellami et al., "PaaS-independent provisioning and management of applications in the cloud," in *2013 IEEE Sixth Int. Conf. on Cloud Computing (CLOUD '13)*, pp. 693–700, IEEE (2013).
11. Cloud Foundry, User Account and Authentication Service, 2012, http://blog.cloudfoundry.org/2012/07/23/introducing-the-uaa-and-security-for-cloud-foundry/ (31 May 2016).
12. W. Ma and K. Sartipi, "An agent-based infrastructure for secure medical imaging system integration," in *IEEE 27th Int. Symp. on Computer-Based Medical Systems (CBMS '14)*, pp. 72–77 (2014).
13. K. Sartipi, K. Kuriakose, and W. Ma, "An infrastructure for secure sharing of medical images between PACS and EHR systems," in *Int. Conf. on Computer Science and Software Engineering (CASCON '13)*, pp. 245–259 (2013).
14. Y. Kakizaki and H. Tsuji, "A decentralized attribute management method and its implementation," *Int. J. Inf. Process. Manage.* **3**(1) (2012).
15. HEART Working Group, 2016, http://openid.net/wg/heart/ (31 May 2016).
16. J. Mandel, "Health relationship trust profile for fast healthcare interoperability resources (FHIR) OAuth 2.0 scopes," 2015, http://openid.bitbucket.org/HEART/openid-heart-fhir-oauth2.html (31 May 2016).
17. Wikipedia, "SAML 2.0," 2016, http://en.wikipedia.org/wiki/SAML_2.0 (31 May 2016).
18. Kantara Initiative, "User-managed access (UMA) profile for OAuth 2.0," https://docs.kantarainitiative.org/uma/rec-uma-core.html, 2012, (31 May 2016).
19. Integrating the Healthcare Enterprise, "IHE radiology technical framework, volume 1: integration profiles," http://www.ihe.net/, pp. 190–205 (2012).
20. T. Lodderstedt et al., "OAuth 2.0 threat model and security considerations," (2013).
21. D. Hardt, "The OAuth 2.0 authorization framework," (2012).
22. D. Ting, O. Hussain, and G. LaRoche, "Systems and methods for multi-factor authentication," U.S. Patent Application 11/698 , 271 (2007).
23. Hamilton health sciences, "Consent to disclosure personal health information," http://www.hhsc.ca/ (2013).
24. Alberta blue cross, "Consent to disclosure personal health information," https://www.ab.bluecross.ca/ (31 May 2016).
25. DMHS, "Durham mental health services consent to disclosure personal health information," 2014, http://www.dmhs.ca/ (31 May 2016).
26. National Institutes of Health, "HIPAA authorization for research," 2012, http://privacyruleandresearch.nih.gov/ (31 May 2016).
27. OASIS, "Using XACML policies as OAuth scope," 2013, https://www.oasis-open.org/ (31 May 2016).
28. OASIS, "eXtensible access control markup language (XACML) version 3.0," 2013, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html (31 May 2016).
29. OASIS, "Request/response interface based on JSON and HTTP for XACML 3.0 version 1.0," 2012, https://www.oasis-open.org/ (31 May 2016).
30. GitHub, Inc., "Source code of OpenID connect implementation in Python," 2016, https://github.com/rohe/pyoidc (31 May 2016).
31. GitHub, Inc., "Source code of CherryWado," https://github.com/malaterre/GDCM/tree/master/Applications/Python (31 May 2016).
32. http://localhost:8080/WADO?requestType=WADO&studyUID="1.3.76.13.10010.0.5.74.3996.1224256625.4053"&seriesUID="1.3.12.2.1107.5.4.4.1053.30000008100608242373400002493"&objectUID="1.3.12.2.1107.5.4.4.1053.30000008100608324685900001822"&contentType=image/jpeg (July 2015).

Weina Ma is a PhD candidate in the Department of Electrical, Computer, and Software Engineering at the University of Ontario Institute of Technology (UOIT), with research interests primarily in digital health services, data privacy and security, knowledge engineering and data mining, and cloud computing. Her work focuses on secure information sharing in distributed PACS systems. She participated in

several commercial and open source software developments. She is a member of SPIE.

Kamran Sartipi received BSc and MSc degrees in electrical engineering from the University of Tehran, and MMath and PhD degrees in computer science (software engineering) from the University of Waterloo. He is a professional engineer. He has over 70 publications in computer science with a focus on information security, software and knowledge engineering, service computing, and medical informatics. He has supervised more than 30 graduate students in interdisciplinary fields and developed several software tools.

Biographies for the other authors are not available.