OpenID Connect as a Security Service in Cloud-based Medical Imaging Systems

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

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 obstacle for adoption of cloud computing by healthcare domains. OpenID Connect, combining OpenID and OAuth together, is an emerging REST-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 authentication and authorization service in cloud-based Diagnostic Imaging (DI) systems, and propose enhancements that allow for incorporating this technology within distributed enterprise environment. The objective of this study is to offer solutions for secure sharing of medical images among DI-r (Diagnostic Imaging Repository) and heterogeneous PACS (Picture Archiving and Communication Systems) 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, deploying DI-r and PACS to private or community clouds should provide equivalent security level to traditional computing model.

Keywords: medical imaging, PACS, security service, OpenID Connect, cloud.

Address all correspondence to: Weina Ma, University of Ontario Institute of Technology, Department of Electrical, Computer and Software Engineering, 2000 Simcoe Street North, Oshawa, Canada, ON L1H 7K4; Tel: +1905-721-8668; Fax: +1 905-721-3178; E-mail: Weina.Ma@uoit.ca

1 Introduction

In medical imaging, PACS (Picture Archiving and Communication System) 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 archive 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 medical imaging market, migrating PACS systems and DI-r services to cloud computing is cost-effective and improves 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, DI-r services); ii) easy to 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 applications to access resource owners’ (e.g., patients) resources (e.g., medical imaging documents) if resource owners approve the actions performed by the client application. Besides, user managed access (UMA) allows resource owner to define access control policies at a centralized authorization server to protect his distributed resources[4]. UMA eliminates the need for the resource owner presence to grant access requests from arbitrary client applications, and the authorization server can make the access decision based on resource owner 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 US Government has joined the OpenID Foundation to create a profile of OpenID Connect and associated projects[5].

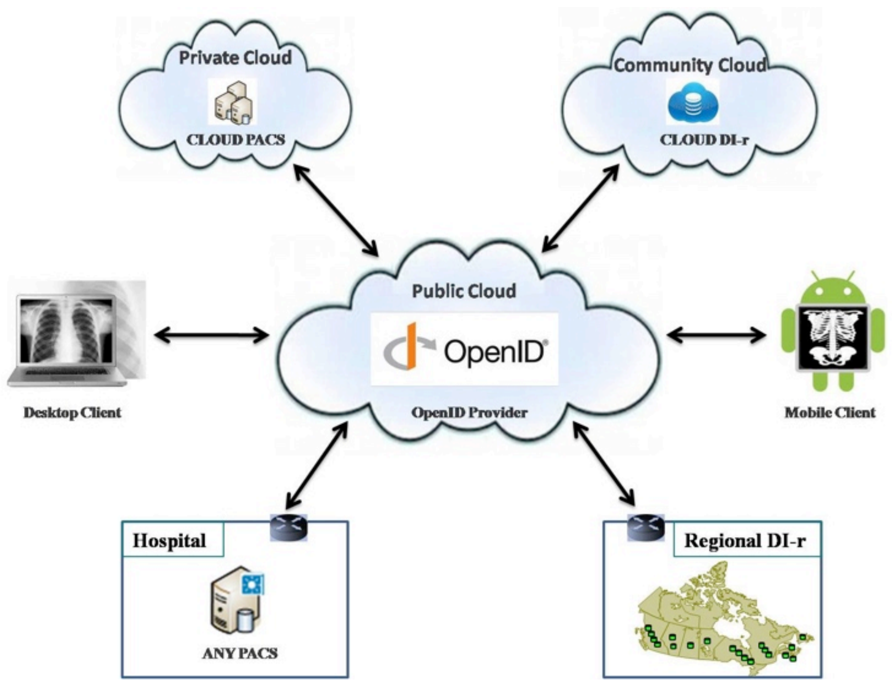


Fig. 1 Overview of OpenID-Connect-as-a-service in cloud-based medical imaging systems.

We propose the design of OpenID-Connect-as-a-service (Figure 1) that provides universal identity management and ease of applying 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 resource owners 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 private or community cloud, rather than public cloud, can provide equivalent security level to traditional computing models. So deploying PACS systems and DI-r's to private cloud or community cloud is 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) who continuously invests on 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 a single sign-on 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 resource owner control permissions of access requests from arbitrary client applications to protect their distributed sensitive resources; iv) providing off-line authorization which allows resource owner define a set of consent-based access control policies to eliminate the resource owner's presence; and v) designing common OpenID-Connect-as-a-service API to easily integrate with medical imaging service providers, web-based and mobile applications.

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

2 Related Work

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

SAML (Security Assertion Markup Language) is dominant and mature technology for enterprise single sign-on, which is well supported by existing enterprise infrastructure. Shibboleth[7] is a standards based, open source software package which implements SAML protocol to provide a federated single sign-on and attribute 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 “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 Subsection 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 cloud), 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 start migrating OpenID 2.0 to OpenID Connect. For example, Google has deprecated OpenID 2.0 and has 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 (PaaS), offering application provisioning and management in cloud environment[10]. User Account and Authentication Management[11] (UAA) is the identity management service of Cloud Foundry, which is also build 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 identity provider, 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 a 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 a resource owner 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 resource owners, and enforced by resource servers. An 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 supply of medications to 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 single sign-on solutions, and discuss the best-practice of identity management, authentication and authorization frameworks in different scenarios.

3.1 Single Sign-on Solutions

Table 1 An overview and comparison of single sign-on solutions.

Standard	**SAML 2.0**	**OAuth 2.0**	**OpenID Connect 1.0**
Introduced	2008	2012	2014
Object	XML	JSON	JSON
Web Service	SOAP	REST	REST
Open Standard	√	√	√
Authentication	√		√
Attribute Claim	√		√
Authorization		√	√
Dynamic Client Registration		√	√
Support Mobile App		√	√
Costly to implement	√		

Table 1 presents an overview and comparison of three widely used single sign-on solutions. SAML[17] is an XML-based, open-standard single sign-on solution. SAML is an agreement between enterprise systems to share information about who a user is, and what attributes the user has. SAML relays 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 protocol to implement. Historical evidence presents 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 protected resource on behalf of the resource owner without sharing credentials such as a password. User Managed Access[18] (UMA) is an enhancement of OAuth 2.0, which allows 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: patient can grant or deny access request with presence; authorization server determines the access permission based on predefined policies if 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 credential. 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 subsection indicates four typical scenarios of medical imaging systems and the best practice of identity frameworks under different scenarios. The consumer service can be desktop application client or mobile client; the access request for protected resource is retrieved from a web service; identity provider is responsible for identity verification and issuing dedicated security token; the user directory stores user attributes.

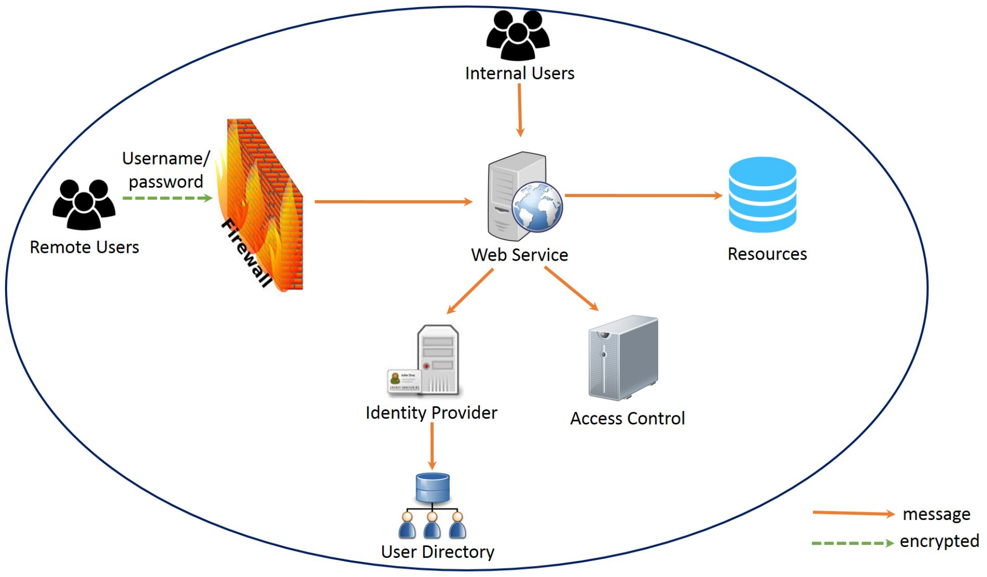


Fig. 2 Direct authentication of single domain.

a) *Direct Authentication of Single Domain*. Figure 2 illustrates the direct authentication pattern[4] of single domain. The protected resources can be secured with HTTP Basic/Digest Authentication by sending username and password in HTTP header. Plaintext username and password may be leaked while in transit, so the communication with remote users who sign in from outside 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.

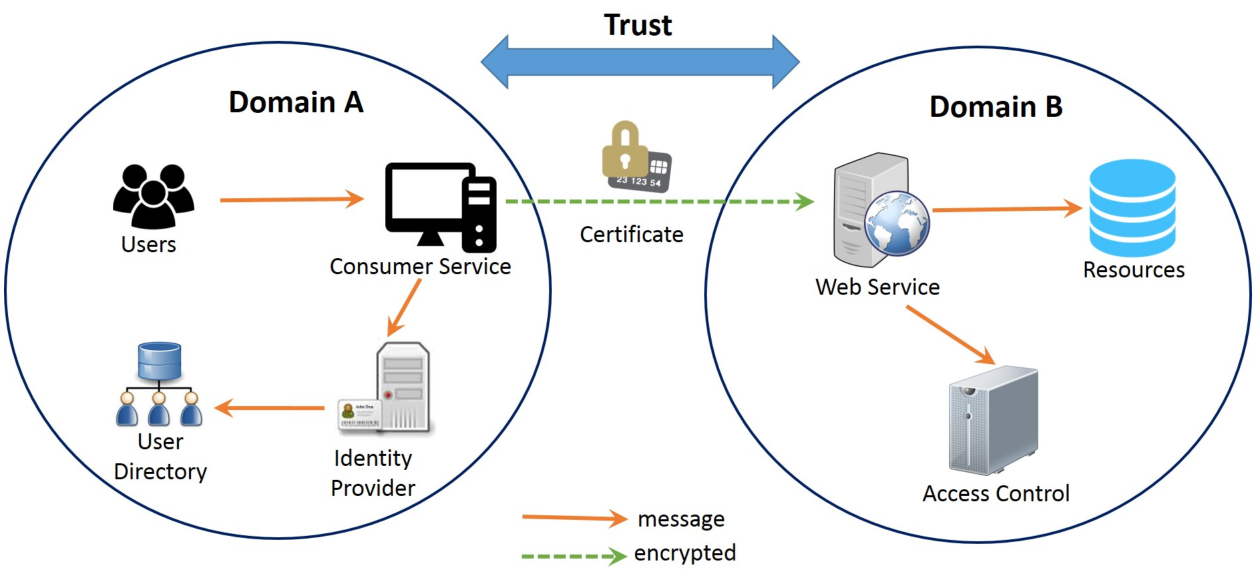


Fig. 3 Trust model across multiple domains.

b) *Trust model across multiple domains.* With the increasing demand of medical image sharing, legacy PACS systems are interconnected across distributed environment through a trust model as shown in Figure 3. Different from password-based direct authentication, certificate-based authentication is used to recognize services to be trusted cross 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.

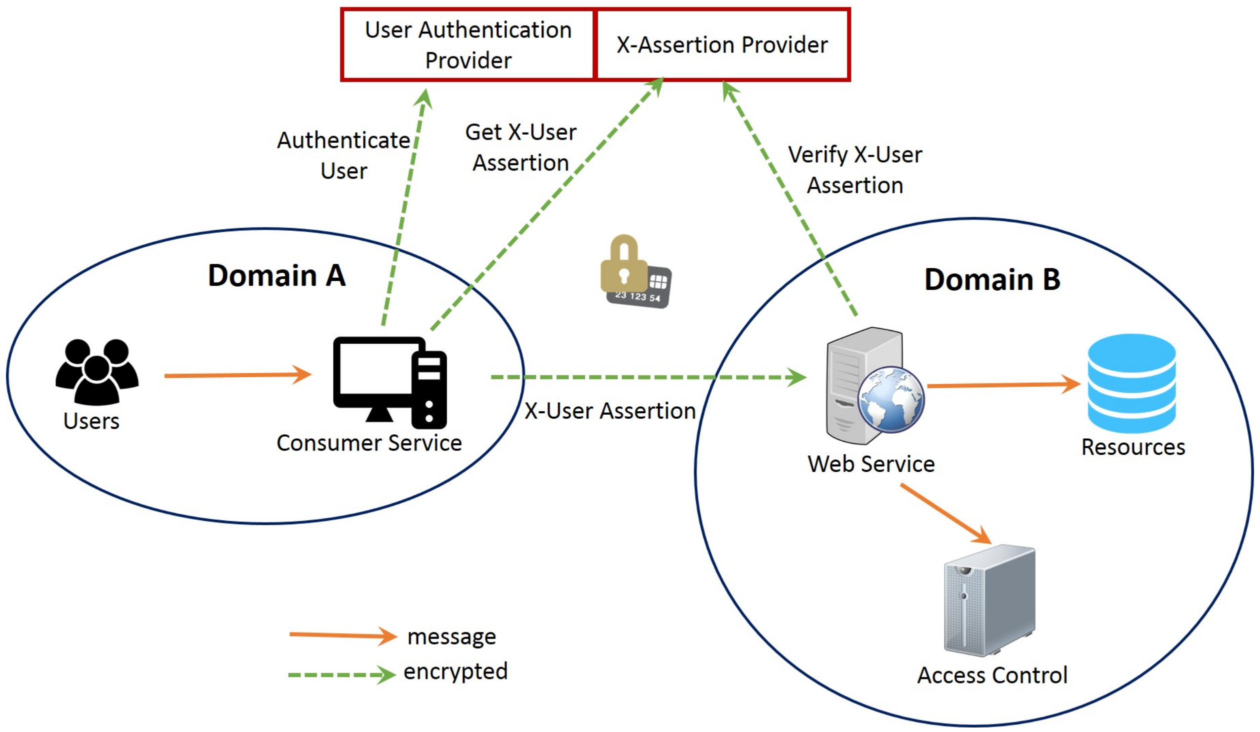


Fig. 4 Cross-enterprise user assertion using SAML.

c) *Cross-enterprise user assertion using SAML.* In Figure 4, Cross-enterprise User Assertion (XUA) provides identity federation by using external centralized identity provider to generate identity claims that are communicated cross 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 policy enforcement in resource domain to determine access permission. The participating enterprises that are unwilling to expose the sensitive user information to third party identity provider 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.

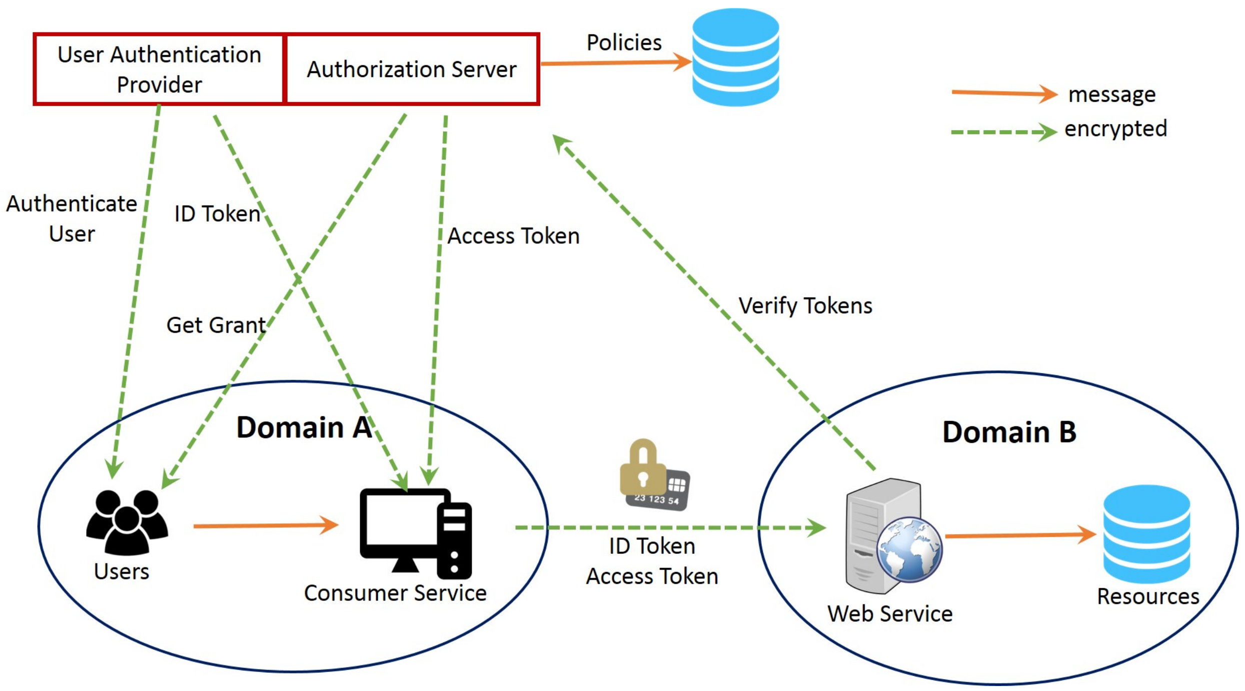


Fig. 5 User-centric authentication and authorization using OpenID Connect.

d) *Authentication and authorization using OpenID Connect.* Figure 5 illustrates the user-centric model using OpenID Connect, which allows resource owner to manage his own identities and to define access control policies at a centralized authorization server. User registers to a centralized authentication provider and get a unique identifier; user can use this identifier to login different consumer services; authentication provider issues an ID token to consumer service if the user is authenticated successfully without credential disclosure. If user is requesting his own resources, authorization server will ask user to control the permissions of consumer service. Also, 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 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 give individuals low barriers and comfort of entry healthcare services.

3.3 Performance Comparison

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

Table 2 presents the performance overview and comparison of four security architectures in Figures 2 to 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, the design and development decisions. We compare the performances of these architectures considering the 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 causes that the consumer service, identity provider, and service provider be located in different domains, and consequently increase 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 identity provider suffer from 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 auto-scaling helps maintain availability with reasonable cost by dynamically scaling identity provider cluster capacity according to the amount of workload. Moreover, SAML is XML-based protocol and OpenID Connect is 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 resource owners to have more control and flexibility to protect their distributed resources. Figure 6 presents the OpenID-Connect-as-a-Servicearchitecture. Electronic medical record (EMR) is individual's health related information (e.g., examination, laboratory test, allergic, medication history, and claims) from family doctors, hospital patient's health record in some scenarios: physician need access patient's diagnostic, treatment and/or care information; tax institution need access individual's benefit plan, claims and personal information; healthcare researchers want to query and view patient's medication and treatment history for a specific disease. Uniformly identifying individuals by each healthcare service is required.

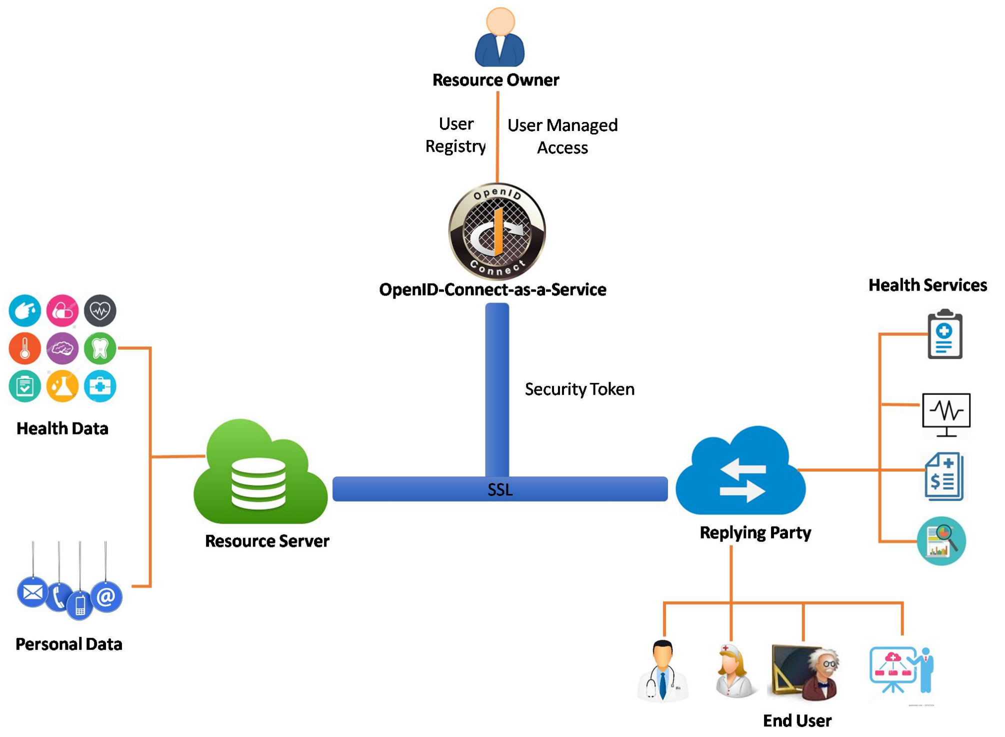


Fig. 6 OpenID-connect-as-a-Service architecture.

To protect 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 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 any service; the individual defines access control policies at OpenID-Connect-as-a-Service to control his/her resources reside on any resource server to be accessed from arbitrary health services.

4.2 Security Considerations in Workflow

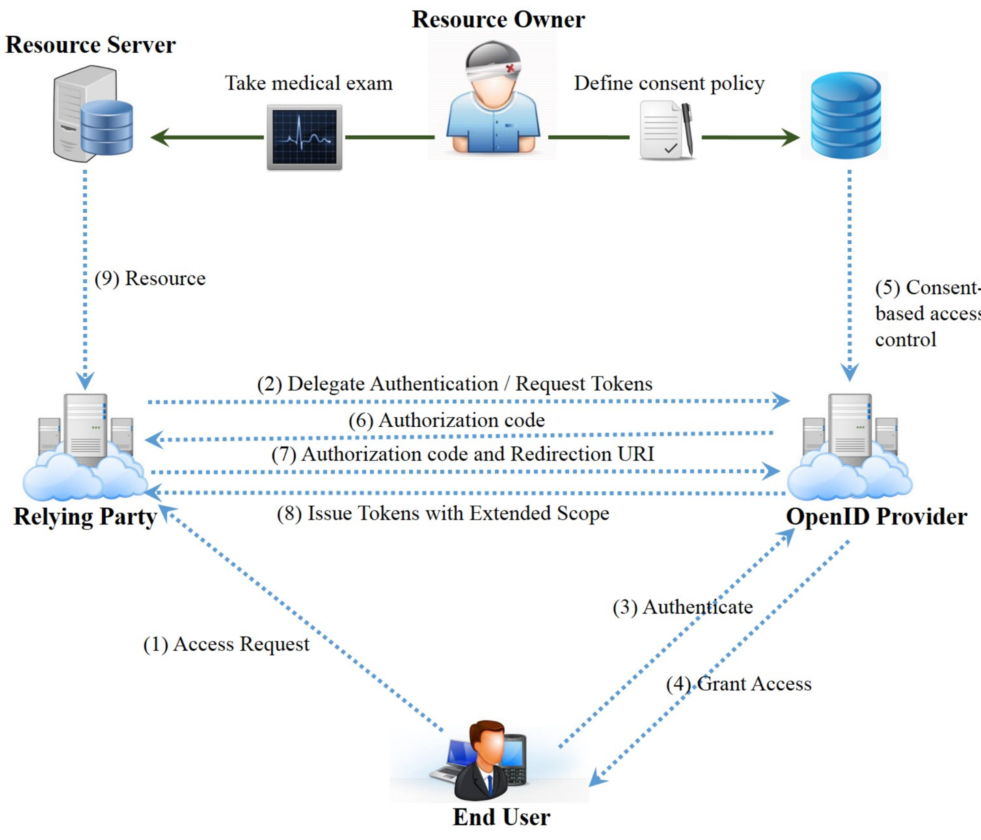


Fig. 7 OpenID-Connect-as-a-Service process flow.

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

- *End User (EU)*: is 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 (RO)*: is capable of granting access to a protected resource (e.g., patient may grant a physician or a healthcare organization to access his/her medical images).
- *Relying Party (RP)*: is application requiring authentication and access grant from OpenID Provider (e.g., PACS system and DI-r services).
- *Resource Server (RS)*: manages resources (e.g., medical images and reports) and their metadata.

- *OpenID Provider (OP)*: is an authorization server that is responsible for issuing tokens to RP after successfully authenticating the end user and obtaining granting 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 that EU and RO is 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 Figure 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 password can be captured[21]. EU should be educated about phishing attacks, for example, 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 access token if the access is granted. A malicious RP can impersonate another RP and obtain access to protected resource[20]. 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. Beside of username and password, 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 is 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 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 impersonal RP.

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

6) *& 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. Access token is used to access protected resource within scope during a period of time. The string is usually opaque to RP. ID token and access token must be kept confidential both in transit and storage. Attackers might obtain ID token and access

token from OP database is OP persist such sensitive information in a database, which is a disaster as all tokens are disclosed. So only token hashes should be stored, and OP has to enforce database security[21].

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

4.3 Authentication and Authorization Flow

In addition of 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 OpenID Provider 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 authentication 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 WADO service as an example, Figure 8 shows the sequence of messages through which the user uses a browser to access DICOM (Digital Imaging and Communications in Medicine) 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 browser. WADO service receives an HTTP request and invokes an authentication operation from AuthN

Service. AuthN Service redirects user's browser to a 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".

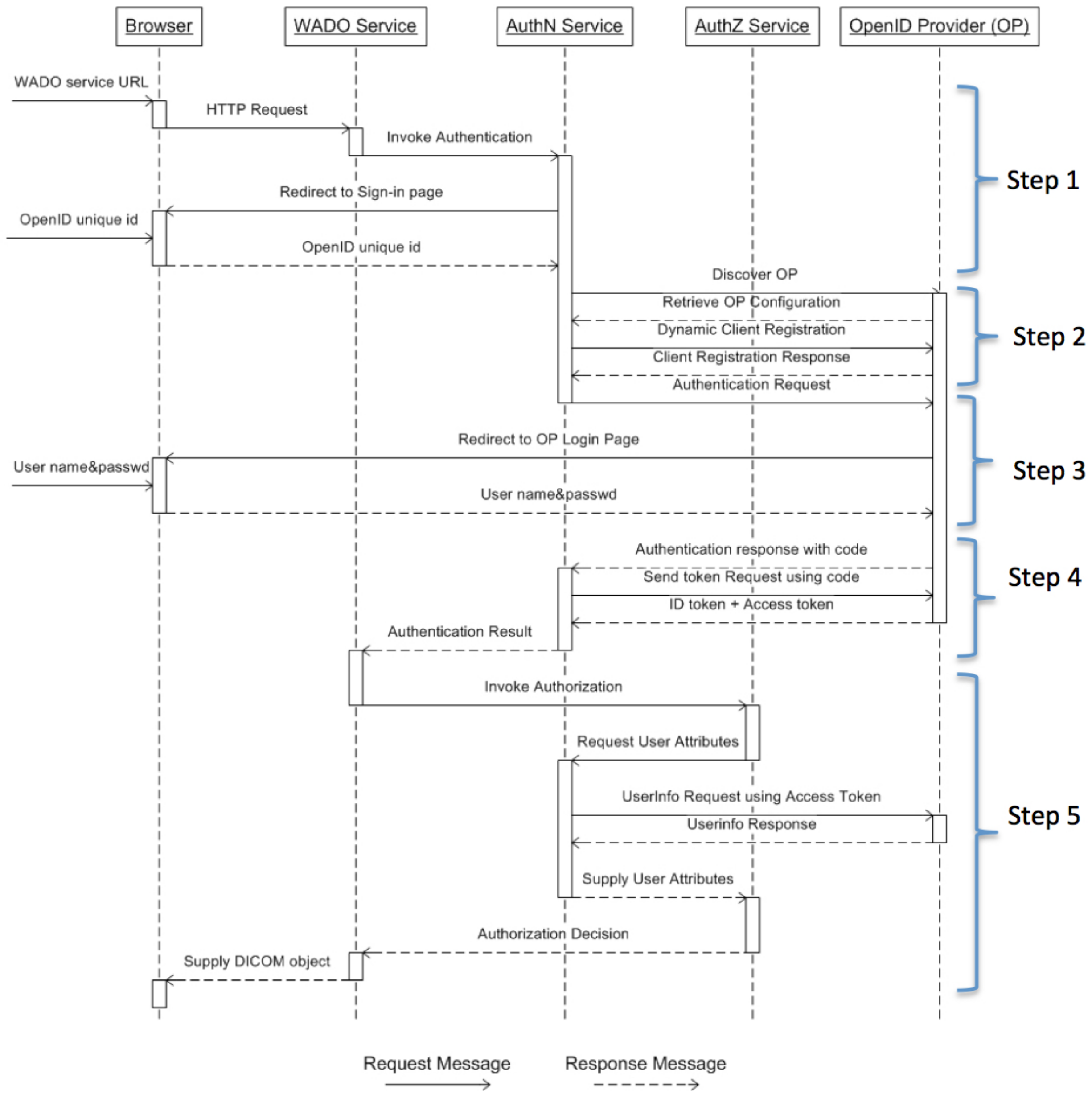


Fig. 8 The sequence for authentication workflow using OpenID-Connect-as-a-services.

Step 2 (*Delegate Authentication and Request Token*): AuthN Service can dynamically discover the OpenID Provider (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.

Step 3 (*Authentication*): after registration, AuthN Service sends authentication request containing the desired request parameters to OP. OP redirects user's browser to a login page to authenticate the user. Any method to authenticate 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 section 4.5.

Step 4 (*Issue Tokens with Extended Scope*): as 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 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 OpenID Provider, 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 need to call AuthN Service API to retrieve user related information.

4.4 Multi-factor Authentication

New threats, risks, and vulnerabilities around cloud applications and mobile devices underscore to the need for a strong authentication approach based on simple service delivery. Multi-factor authentication[22] is a method of authenticating a user by successfully presenting several separate authentication stages. The more factors 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, 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 multi-factor 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). Relying Party might set ACR that OpenID Provider requested to use for processing the authentication request. OpenID Provider may ask End User to re-authenticate with additional factors to meeting the ACR requirement. OpenID Connect specifies the multi-factor authentication process flow which makes it easy to integrate with strong authentication method. Multi-factor authentication is already implemented in production operations at some OpenID Connect identity providers[5].

4.5 *Consent-based Access Control*

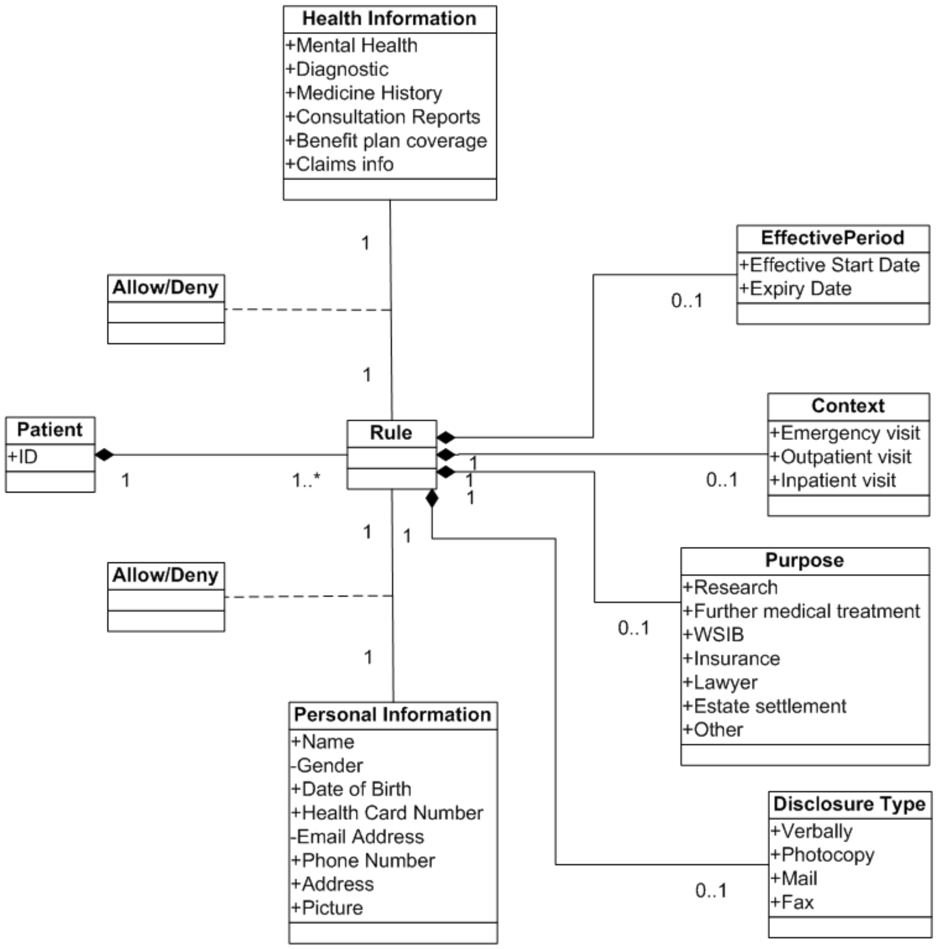
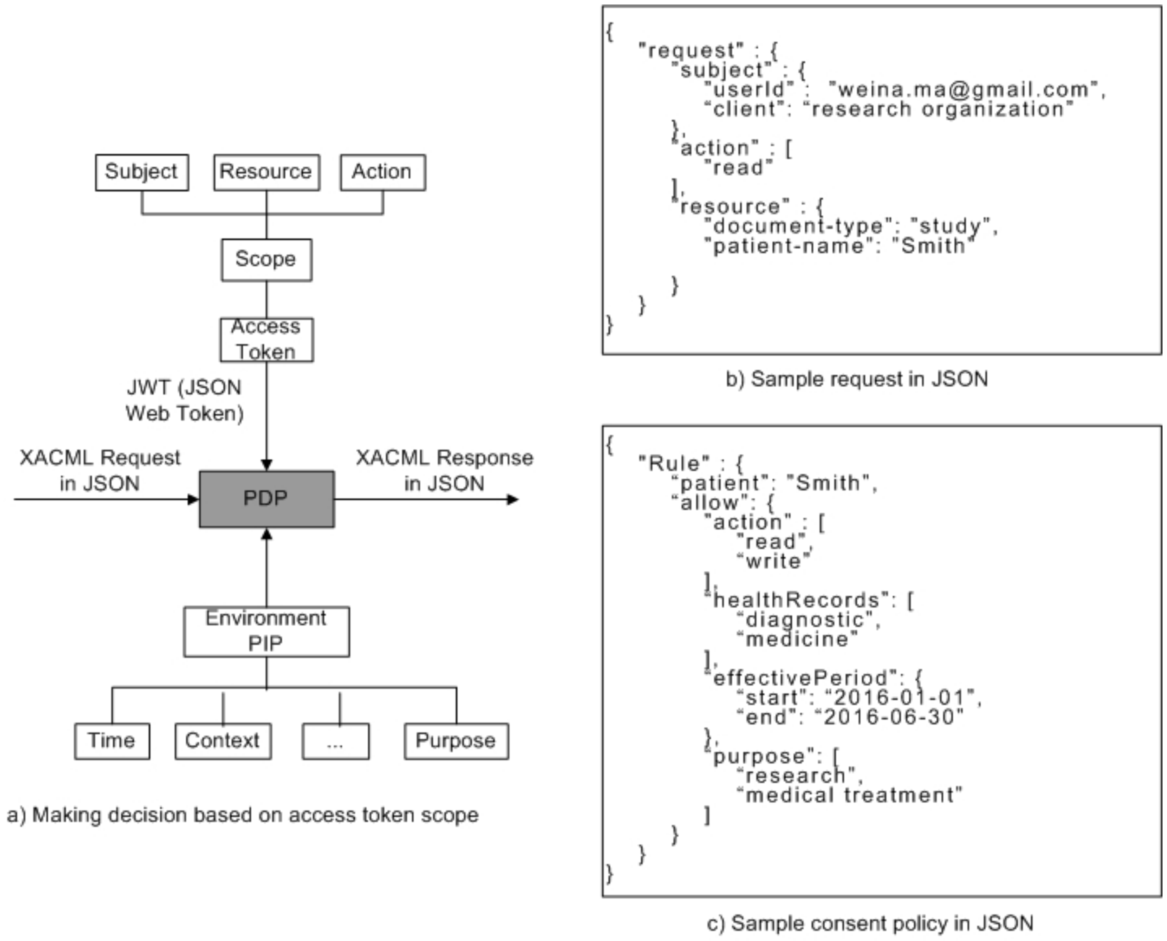


Fig. 9 Electronic patient consent class diagram.

We summarized several existing paper-based patient consent disclosure forms used in Canadian hospitals[23,24,25,26] and produced a UML class diagram, as shown in Figure 9, to model privacy-based access control polices. 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 takes effect; the disclosure type of how to share 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 resource owner 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 request parameter; in turn, authorization server dispatches access token together with "scope" as response parameter to inform the healthcare service of the constrains of 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 people birth date, contact address, picture and so on are defined in OAuth 2.0 protocol. 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 FHIR resource such as patient, medication order, observation, appointment, etc. However, their proposed schema still cannot describe the complex patient consent policy defined in Figure 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 XACML representing in JSON, and then evaluating the access token based on existing XACML access control infrastructure[28].



a) Making decision based on access token scope

```
{
    "request" : {
        "subject" : {
            "userId" :  "weina.ma@gmail.com",
            "client": "research organization"
        },
        "action" : [
            "read"
        ],
        "resource" : {
            "document-type": "study",
            "patient-name": "Smith"

        }
    }
}
```

b) Sample request in JSON

```
{
    "Rule" : {
        "patient": "Smith",
        "allow": {
            "action" : [
                "read",
                "write"
            ],
            "healthRecords": [
                "diagnostic",
                "medicine"
            ],
            "effectivePeriod": {
                "start": "2016-01-01",
                "end": "2016-06-30"
            },
            "purpose": [
                "research",
                "medical treatment"
            ]
        }
    }
}
```

c) Sample consent policy in JSON

Fig. 10. Consent-based access control enforcement.

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 XACML model to provide applicable polices to authorization server, and to evaluate issued access token. Figure 10 a) illustrates the mechanism of employing XACML PDP (policy decision point) to make access decision according to access token issued by authorization server. XACML request designates the actual actions on the resource that are limited by the scope of the access token. PIP (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 Figure 10 b) & c). XACML 3.0 has a working draft about request and response interface based on JSON and HTTP[29]; nevertheless, JSON representation of XACML polices must be an alternative to current XML representation in near future.

5 Prototype Implementation

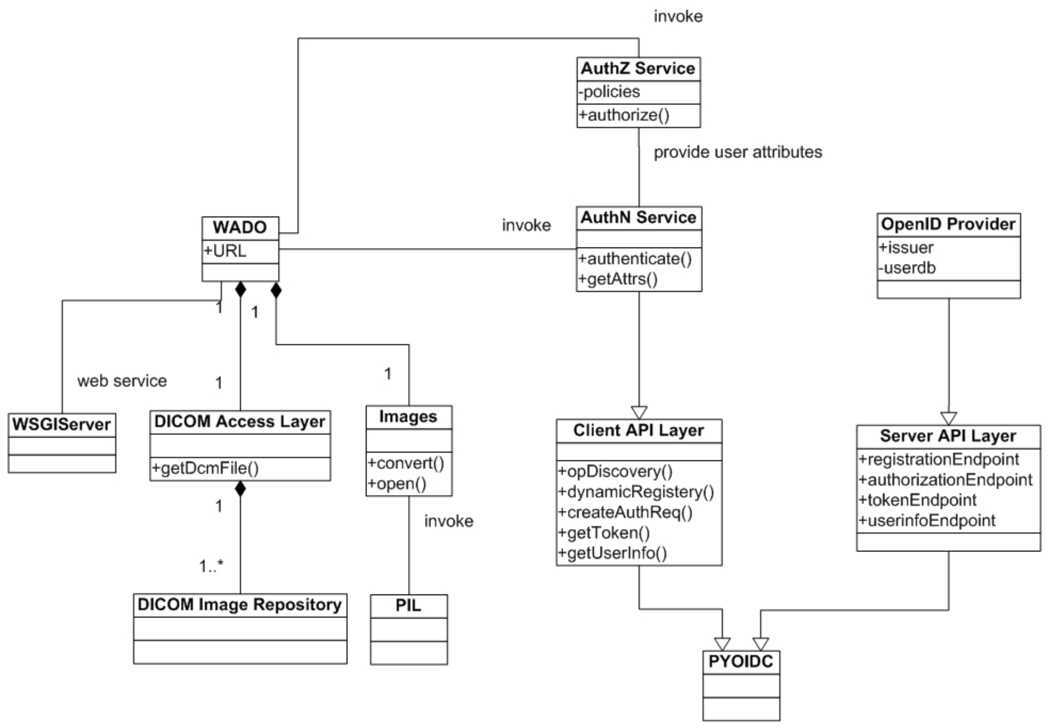


Fig. 11 The class diagram describes the structure of prototype implementation.

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 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 (PIL) that 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 grant, WADO retrieves DICOM file from DICOM image repository using “DICOM Access Layer” as an intermediary. At last the DICOM file is converted to more compatible format such as JPEG through class Image and is displayed in 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 service provider requiring user authentication and claims from OpenID Provider), a Client API Layer is implemented that inherits the functionalities provided by PYOIDC. The method “opDiscovery()” is used to find the location of desired OpenID Provider, and method “dynamicRegistery()” registers RP to OpenID Provider as a client on behalf of AuthN Service. The methods “createAuthReq()”, “getToken()”, and “getUserInfo()” implement the interactions with OpenID Provider following the process presented in Section 4.

Through a new layer “Server API Layer”, an OpenID Provider is implemented using the functionalities provided by PYOIDC. The class member “RegistrationEndpoint” allows a client to register at OpenID Provider; “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 OpenID Provider “issuer” enables determining the location of the OpenID Provider; a “userdb” that persists and manages user information, centralized or decentralized, can be integrated with OpenID Provider.

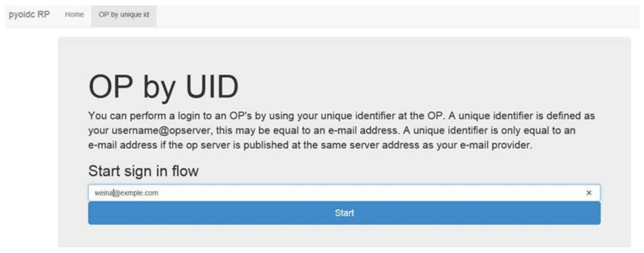
A typical authorization service, AuthZ Service, makes determination for an access request based on available information (e.g., user attributes, protected resource attributes) and applicable security policies. The user related information could be retrieved from AuthN Service from OpenID Provider.

6 Experiment Result

Following is the experiment result of the implementation prototype presented in section 5. A user account is predefined in OpenID Provider 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:

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".



pyoidc RP Home OP by unique id

OP by UID

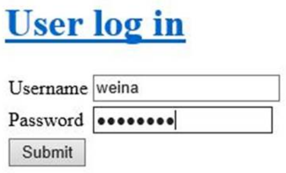
You can perform a login to an OP's by using your unique identifier at the OP. A unique identifier is defined as your username@opserver, this may be equal to an e-mail address. A unique identifier is only equal to an e-mail address if the op server is published at the same server address as your e-mail provider.

Start sign in flow

weina@exmple.com

Start

(a)



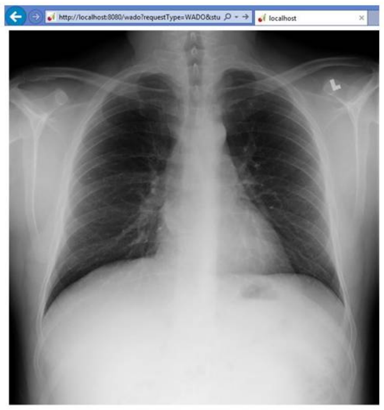
User log in

Username weina

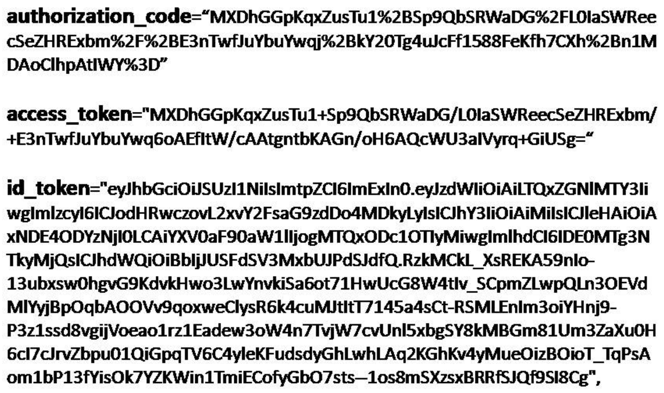
Password ••••••••

Submit

(b)



(c)



authorization_code="MXDhGGpKqxZusTu1%2BSp9QbSRWaDG%2FL0IaSWReecSeZHRExbm%2F%2BE3nTwfJuYbuYwqj%2BkY20Tg4uJcFf1588FeKfh7CXh%2Bn1MDAoClhpAtIWY%3D"

access_token="MXDhGGpKqxZusTu1+Sp9QbSRWaDG/L0IaSWReecSeZHRExbm/+E3nTwfJuYbuYwq6oAEfItW/cAAtgntbKAGn/oH6AQcWU3aIVyrq+GiUSg="

id_token="eyJhbGciOiJSUzI1NiIsImtpZCI6ImExIn0.eyJzdWIiOiAiLTQxZGNlMTY3IiwgImlzcyI6ICJodHRwczovL2xvY2FsaG9zdDo4MDkyLyIsICJhY3IiOiAiMiIsICJleHAiOiAxNDE4ODYzNjI0LCAiYXV0aF90aW1lIjogMTQxODc1OTIyMiwgImlhdCI6IDE0MTg3NTkyMjQsICJhdWQiOiBbIjJUSFdSV3MxbUJPdSJdfQ.RzkMCkL_XsREKA59nIo-13ubxsw0hgvG9KdvkHwo3LwYnvkiSa6ot71HwUcG8W4tIv_SCpmZLwpQLn3OEVdMlYyjBpOqbAOOVv9qoxweClysR6k4cuMJtItT7145a4sCt-RSMLEnIm3oiYHnj9-P3z1ssd8vgijVoeao1rz1Eadew3oW4n7TvjW7cvUnl5xbgSY8kMBGm81Um3ZaXu0H6cI7cJrvZbpu01QiGpqTV6C4yleKFudsdyGhLwhLAq2KGhKv4yMueOizBOioT_TqPsAom1bP13fYisOk7YZKWin1TmiECofyGbO7sts--1os8mSXzsxBRRfSJQf9SI8Cg",

(d)

Fig. 12 Experiment 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.

WADO service receives the access request and asks for AuthN Service to do authentication. Figure 12-(a) shows redirected page asking for 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 OpenID Provider host "example.com". AuthN Service is able to find the location of OpenID Provider using "example.com". OpenID Provider redirects to user login page and needs user input username and password as shown in Figure 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, Figure 12-(c). The Authorization Code, Access Token and ID Token, exchanged between AuthN Service and OpenID Provider, are shown in Figure 12-(d).

7 Conclusion and Discussion

In this paper, we introduced OpenID-Connect-as-a-service to provide user-centric single sign-on 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 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 OpenID Provider 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 OpenID Provider. 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 is a set of open standards rather than an end-to-end solution, and more and more extortions are adding to enhance the standard. The integrating systems need their own implementation and make sure the implementation is 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, integrating into current medical imaging system workflow and achieve 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. Branstetter, B. F. (Ed.), “Practical imaging informatics: foundations and applications for PACS professionals”, Springer, 33-47 (2009).

2. Gauvin, A., “Status of Diagnostic Imaging Repository (DI-r) projects across Canada”, http://www.camrt.ca/ (2010)
3. OAuth Community website, http://oauth.net/
4. Siriwardena P., Advanced API Security: Securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE. Apress (2014).
5. OpenID Connect website, http://openid.net/
6. Canada Health Infoway, “Cloud Computing in Health White Paper”, https://www.infoway-inforoute.ca/ (2012).
7. Shibboleth, https://shibboleth.net
8. Khan, R. H., Ylitalo, J., and Ahmed, A. S., “OpenID authentication as a service in OpenStack,” In Information Assurance and Security (IAS), 7th International Conference, IEEE, 372-377 (2011).
9. Migrating from OpenID 2.0 to OpenID Connect, https://developers.google.com/identity/protocols/OpenID2Migration
10. Sellami, M., Yangui, S., Mohamed, M., & Tata, S. PaaS-independent Provisioning and Management of Applications in the Cloud. In Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on (pp. 693-700). IEEE (2013).
11. Cloud Foundry User Account and Authentication Service, http://blog.cloudfoundry.org/2012/07/23/introducing-the-uaa-and-security-for-cloud-foundry/
12. Ma W, and Sartipi K., “An Agent-Based Infrastructure for Secure Medical Imaging System Integration," Computer-Based Medical Systems (CBMS), IEEE 27th International Symposium, 72-77 (2014)
13. Sartipi, K.,Kuriakose, K., and Ma, W., “An Infrastructure for Secure Sharing of Medical Images between PACS and EHR Systems,” International Conference on Computer Science and Software Engineering (CASCON), 245-259 (2013)
14. Kakizaki, Y., and Tsuji, H., “A Decentralized Attribute Management Method and its Implementation,” International Journal of Information Processing & Management 3.1 (2012)

15. HEATR WG, http://openid.net/wg/heart/

16. Health Relationship Trust Profile for Fast Healthcare Interoperability Resources (FHIR) OAuth 2.0 Scopes, http://openid.bitbucket.org/HEART/openid-heart-fhir-oauth2.html

17. SAML 2.0 wiki, http://en.wikipedia.org/wiki/SAML_2.0

18. User-Managed Access (UMA) Profile for OAuth 2.0, https://docs.kantarainitiative.org/uma/rec-uma-core.html

19. "IHE Radiology Technical Framework, Volume 1: Integration Profiles", http://www.ihe.net/ ,190-205 (2012)

20. Hardt, D., "The OAuth 2.0 authorization framework.". (2012)

21. Lodderstedt, T., McGloin, M. and Hunt, P., OAuth 2.0 threat model and security considerations. (2013)

22. Ting, D., Hussain, O., &LaRoche, G. "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/ (2013)

25. Durham Mental Health Services Consent to Disclosure Personal Health Information, http://www.dmhs.ca/

26. HIPAA Authorization for Research, http://privacyruleandresearch.nih.gov/

27. Using XACML Policies as OAuth Scope, https://www.oasis-open.org/ (2013)

28. eXtensible Access Control Markup Language (XACML) Version 3.0, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html (2013)

29. Request/Response Interface based on JSON and HTTP for XACML 3.0 Version 1.0, https://www.oasis-open.org/ (2012)

30. Source code of OpenID Connect implementation in Python, https://github.com/rohe/pyoidc

31. Source code of CherryWado, https://github.com/malaterre/GDCM/tree/master/Applications/Python

Weina Ma is a Ph.D 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.

Karmran Sartipi received B.Sc and M.Sc in Electrical and Computer Engineering from University of Tehran, and MMath and Ph.D in Computer Science (Software Engineering) from University of Waterloo. He is a Professional Engineer. Dr. Sartipi has over 70 publications in Computer Science with focus on Information Security, Software and Knowledge Engineering, Data Analytics, Service Computing, and Medical Informatics. He has supervised more than 30 graduate students in inter-disciplinary fields, and developed several software tools in different scientific areas. He has collaborated with researchers in computer science, engineering, health science, business, and entrepreneurship fields for several years.