Introducing a Dynamic Federation Model for RESTful Cloud Storage

Yang Xiang
Rechenzentrum Garching
Max-Planck-Society
Garching, Germany
yang.xiang@rzg.mpg.de

Sebastian Rieger
GWDG
Max-Planck-Society
Göttingen, Germany
sebastian.rieger@gwdg.de

Harald Richter
Department of Informatics
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
hri@tu-clausthal.de

***Abstract*—This paper presents a solution for RESTful cloud storage in a dynamic identity federation. With dynamic federations, Cloud Service Providers are able to find Identity Providers autonomously in the cloud in order to make services flexible, scalable and interoperable. By combining a Representational State Transfer architecture with SAML-based identity federation, a distributed and decentralized cloud storage is provided which allows users to access files via the Internet seamlessly and transparently.**

***Keywords*-Dynamic Identity Federation; REST; Cloud Computing; Storage; SAML**

I. Introduction

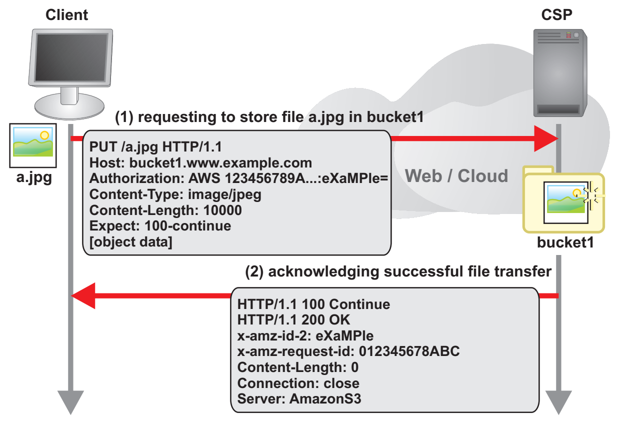
Cloud computing is a modern way to provide IT services as a kind of commodity to customers via the Internet. In storage clouds, which are a specialization of cloud computing, files can be kept at different sites, and the user is able to mount his virtual storage from any computer connected to the Internet independently of which site actually hosts the data. This is called transparent and seamless file access and it is usually accomplished by offering a web interface to the user.

This paper introduces a new solution for authentication and authorization (AA) for storage clouds which employ dynamic identity federation. Its focus lies on contemporary storage clouds such as Amazon S3 [1] and Google Storage [2]. Both clouds are decentralized, web-based and use the Representational State Transfer architecture (REST) [3] as a communication framework between the server(s) of the storage cloud and the users. Though having commonalities, S3 and Google Storage are incompatible with each other. In the following, a model for dynamic federation is described that simultaneously supports multiple cloud service providers (CSPs) by augmenting "RESTful" storage clouds. Here RESTful means that the clouds have to be compatible with protocol definitions and constraints according to REST. However, different storage cloud providers normally do not federate their services, thus user files stored on a given provider will not be accessible from another provider. The model which we present here has the advantage that a user will be presented with a single means of access which spans CSPs. Furthermore the user has more flexibility and can change CSPs completely or switch between CSPs temporarily at will. However, more flexibility for the user also requires a faster establishment of trust between user and CSP. Establishing, measuring and predicting trust in an automatic manner is one of the targets of our model. Please note that this model is not intended to act as an identity management system, and that it is not limited to storage clouds only.

Since decentralized AA, as used in identity federations, offers a unified means of authentication and authorization across different storage cloud providers, both S3 and Google Storage can be accessed in a uniform manner. Here, it is necessary to focus on web browser clients and on RESTful file access together with virtual file systems as described by the Storage Cloud Initiative of SNIA (Storage Networking Industry Association) [4]. Our model follows this industry standard and implements a unified AA for cloud-based storage solutions.

A. State-of-the-Art

In recent years, RESTful web services have gained popularity and may be a potential alternative to SOAP solutions [5][6]. Compared to SOAP, REST directly uses HTTP methods to transfer data between client and server without much protocol overhead. REST is less complex and thus less resource-intensive than SOAP which is the reason why it has gained interest over SOAP. REST data structures can be encoded in HTML or XML, and RESTful web services are easily understandable and human-readable since they are reduced to a minimum and in plain text. S3, for example, uses HTTP PUT, GET and DELETE methods in a RESTful style to read, write and manage files via so-called buckets [1]. Access control, i.e., file access authorization is accomplished by extra protocol data called authorization header that contains all user credentials. This header and the REST-based file access structure of S3 is shown in Figure 1 as an example. Other cloud storage providers such as Ubuntu One [7] use the same technique or even extend the REST functionality by using WebDAV [8]. Regarding the prevalence of RESTful applications in contemporary storage clouds, REST could become a basis also for future clouds which is why our AA system uses this technology. However, the RESTful approach in S3 has two major drawbacks: First, the user needs a special client or middleware that is able to create and send the proprietary authorization header to the CSP. As a result,


Client
CSP
a.jpg
(1) requesting to store file a.jpg in bucket1
Web / Cloud
bucket1
(2) acknowledging successful file transfer

```
PUT /a.jpg HTTP/1.1
Host: bucket1.www.example.com
Authorization: AWS 123456789A...:eXaMPle=
Content-Type: image/jpeg
Content-Length: 10000
Expect: 100-continue
[object data]
```

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: eXaMPle
x-amz-request-id: 012345678ABC
Content-Length: 0
Connection: close
Server: AmazonS3
```

Figure 1. Basic access structure of Amazon S3

the interoperability between different CSPs is low. Second, the user needs multiple credentials, one for every CSP he uses. While the interoperability problem is being addressed by SNIA in their CDMI standard [4], the multiple credential problem still limits the simultaneous usage of different storage clouds. Furthermore, if all users of a storage cloud are not under the umbrella of the same scientific or commercial organization, difficulties to authenticate and authorize them arise and the following two problems have to be solved: How to authenticate an individual user accessing the service anonymously from the Internet? How to manage AA of thousands of users that belong to different organizations which are all different cloud customers? As a solution to these problems, an extension to existing AA infrastructures is proposed that is based on identity federation.

In an identity federation, a CSP does not have to care about the user's identity by itself. Instead, it delegates this task to a so-called identity provider (IdP) which is responsible for the user, and who acts as a user proxy. The idea is to augment the identity management system of a CSP by the AA concept of an identity federation. For this purpose, we suggest a "trust estimation system" (TES) as a key stone of our model to quickly establish and estimate the quality of the trust relationship between CSP and user that computes numerical quantities for the representation of trust and reputation. Our TES uses SAML [9] for communication and can be easily incorporated into a Shibboleth [10] IdP or SP. SAML was chosen because it is widely supported by major service providers as their de-facto AA standard.

The remainder of the paper is organized as follows: in Section 2, related work is presented. In Section 3, the requirements for identity federations that hold in storage clouds are defined. Section 4 describes the set-up of the proposed dynamic identity itself. Section 5 presents the TES and its implementation. In Section 6, conclusion and future work are given.

II. Related work

There are efforts to combine the advantages of REST and SAML. A US patent [11], for instance, describes a method to securely invoke a REST API-call by means of a SAML security token. With this method, a client obtains a security token from an authentication server beforehand. When a user sends a request to an application server to invoke a REST API-call, authentication is performed by means of an HTTP-digest. After successful authentication, the client computes a token digest by using the security token it has received from the authentication server, together with a NONCE (number used once) and a time stamp from the application server. Finally, the client sends this token digest back to the application server. This method is considered good but not appropriate for the environment of CSPs.

There are a few open source projects that are focusing to provide cloud-based virtual file systems, for example iRods and GridFS from mongoDB [12], beside the large commercially available clouds Amazon S3 and Google Storage. Furthermore, SAML-based solutions for identity federation such as Shibboleth have been used already in several research projects in order to provide federated AA to users of iRods [13]. However, such AA is based on statically-federated Shibboleth-implementations only, and the cloud storage is also not RESTful in these projects.

III. Requirements for Identity Federations in Storage Clouds

Large scientific and commercial communities typically have a spatially distributed structure. The information and communication equipment may be disjoint and in part even incompatible between organizational branches. Different hardware, operating systems and middleware may be in use, as well as different storage techniques, either on the record level or the file system level. Furthermore, different qualities in trust and reputation may exist for users and user communities, as a result of how they have behaved in the past. Regarding this situation, there are several requirements for AA in storage clouds:

- AA should serve for multiple storage clouds simultaneously, and the designated AA mechanism should support multiple users in each identity federation.
- There should be no central AA instance because this would be not scalable and a single-point of failure as well. A distributed AAI is needed instead.
- There shall be a mechanism for adding and removing users dynamically for short-term projects which is similar to virtual organizations (VOs) known from grid computing.
- Each home institute or subsidiary may use a different AA system. For all institutes that are not willing to use SAML, a backdoor solution should be available by means of an AA gateway that translates local AA tokens into SAML.
- AA should also work together with CSPs that do not support entering the home organization of a user in the CSP's web form. As a consequence, direct file access without a web form is needed. This means that the global verification of locally known user names and passwords must work under all circumstances. Therefore, a mechanism is sought to automatically detect the user's

home organization, instead of requesting him to select manually his home organization in a web form as is common in current practice.
- A trust estimation service (TES) is required which can ensure that identity assertions truly come from an IdP as a user proxy rather than from an intruder. The TES must be resistant against several types of attacks.
- Users must be able to apply the same credentials they gained from their home institute or subsidiary for accessing different CSPs (i.e., single sign-on).
- User credentials must not be transmitted beyond an organizational border in non-encrypted form in order to protect privacy. Furthermore, to make the communication secure, data transfer between identity providers (IdPs) and cloud service providers (CSPs) must be encrypted to prevent user name and password phishing.
- Access to user files should be based on existing de facto standards of storage clouds. Furthermore, AA should be compatible with existing file system and file-backup tools.

In our opinion, these requirements lead to a need for a dynamic federation model, where values for trust and reputation are calculated quickly and automatically for each user and IdP. More information about the means of calculating the trust and reputation values can be found in Xiang et al. [14].

IV. IDENTITY FEDERATIONS

The following two subsections illustrate the advantages of dynamic federations, as presented in this paper, over existing static federation solutions.

A. Static Identity Federation

There are different types of static identity federations with respect to the underlying software technologies. For example, OpenID and Windows Card Space belong to the user-centric category, while Shibboleth is an institution-centric system. However, both categories exhibit the same problems which are listed as follows:

1) There is substantial manual operator effort to maintain existing identity federations which lies in the order of $O(n^2)$, resulting from the fact that trust relationships between every pair of CSPs and user must be defined.
2) Trust relationships are expressed by static values only. This limits the cloud's scalability since increasing the number of users becomes very time-consuming.
3) It is difficult or impossible to dynamically connect independent federations to form a confederation because an entity from one federation does not know and hence does not trust entities from the other federations.
4) If a major customer of a CSP wants to resell the service he obtains from his CSP to his own sub-customers, then the CSP has to add all sub-customers to its trusted IdP list. No hierarchic linking is possible.
5) The costs of adding customers into an identity federation increases substantially if customers join and leave the cloud at a high rate. This is a hindrance for clouds to sell storage as a commodity.

B. Dynamic Identity Federations

In a dynamic identity federation, a CSP does not need to know an IdP beforehand. A trust relationship is created on demand, and a corresponding trust value will be determined on-the-fly. This is beneficial to those CSPs who want to provide identity as a service (IDaaS) as a new business model in cloud computing. CSPs which are specialized on IDaaS act as an identity provider for other CSPs that want to sell more elaborated services such as infrastructure as a service (IaaS), platform as a service (PaaS), or software as a service (SaaS). This means, CSPs for IDaaS bridge the gap between end-customers and other CSPs that are specialized in IaaS, PaaS or SaaS. As a result, a new business model may be established since CSPs for IaaS, PaaS and SaaS can simply delegate AA to an IDaaS provider. Hence, the 1:n relationship of an existing static federation is reduced to a 1:1 relationship in a dynamic federation as shown in Figure 2 . In Figure 3, the

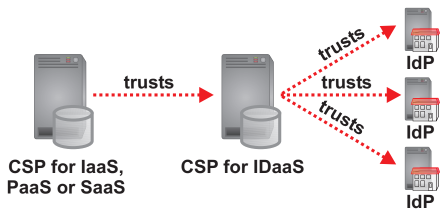


Figure 2. CSP for IDaaS acts as an identity store for other CSPs

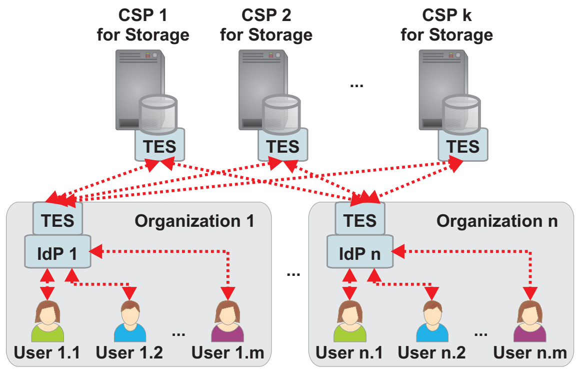


Figure 3. AA for storage clouds serving multiple identity federations

dynamic identity federation and the TES for storage clouds are illustrated. It supports AA for multiple storage clouds that in turn serve multiple organizations or institutions which form a dynamic federation. AA inside an organization or institution is accomplished via local IdPs which are responsible for subsets of users. The TES performs the communication between IdPs and CSPs and computes trust values. in Figure 4, an arbitrary organization i and a random user j inside of i are chosen as an example scenario in order to explain the function of the TES. In this figure, the individual protocol steps between user, IdP and CSP are as follows:

1) User $i.j$ shall be a customer of CSP k which has electronically signed a contract with IdP i that all users from organization i are allowed to access the storage

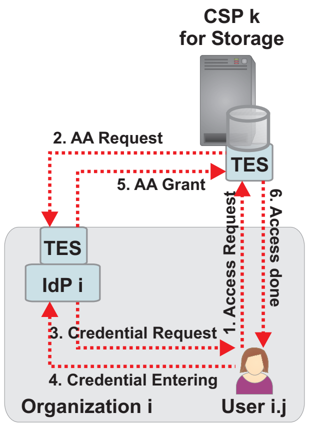


Figure 4. AA protocol inside an identity federation with TES

service of CSP k. Now, user $i.j$ is requesting access to a file at CSP k.

2) Then, CSP k determines that IdP i is responsible for user $i.j$ by means of a dynamic discovery service (DDS) - that is part of our TES - and asks IdP i for authentication, instead of authenticating $i.j$ by itself.
3) IdP i requests user $i.j$ to enter his user name and password.
4) User $i.j$ supplies his username and password to IdP i.
5) After the IdP i has successfully authenticated user $i.j$ it will send a positive response back to CSP k. CSP k may then request more attributes of user $i.j$ from IdP i for subsequent authorization.
6) Finally, the file access is executed by CSP k and the operation is acknowledged positively or negatively to user $i.j$.

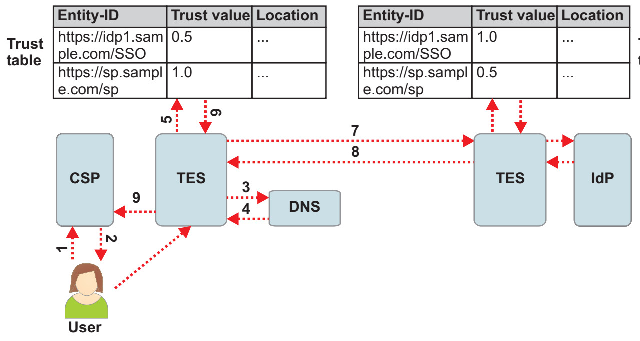
After the first successful AA procedure, user $i.j$ can access his files without reentering his user name and password as long as the HTTP session with the IdP persists. With identity federation, CSPs and IdPs trust all entities in that federation more or less, varying on their trust value. The trust value the user's IdP i has with respect to the user's CSP k is estimated by the TES.

V. Implementation of Dynamic Identity Federations with TES for RESTful Cloud Storage

In the next two subsections, the structure of the TES will be presented.

A. Extending Shibboleth with a TES

For our TES prototype, Shibboleth was chosen as a basis because it is widespread and functional. Shibboleth is a SAML-based framework for static identity federations and consists of two parts, an IdP which is written in Java, and a SP which is implemented in C++. It can be used as an authentication module for Apache web servers. In Shibboleth, the trust relationship between entities is established with static meta data containing one or more X.509-certificates. To make Shibboleth dynamic by exchanging meta data on demand and in real time, Shibboleth is extended by a TES as shown in Figure 5. The data flow of the new functionality is as follows:



Entity-ID	Trust value	Location
https://idp1.sample.com/SSO	0.5	...
https://sp.sample.com/sp	1.0	...

Entity-ID	Trust value	Location
https://idp1.sample.com/SSO	1.0	...
https://sp.sample.com/sp	0.5	...

Figure 5. TES as a Shibboleth extension

1) When an end user accesses a CSP, he will be redirected to a dynamic discovery service (DDS) which is a part of our TES. He can then input his e-mail address (steps 1 and 2 in Figure 5).
2) After step 1, DDS/TES sends a query to the domain name system of the Internet to obtain the Entity-ID of the user's home IdP (steps 3 and 4 in Figure 5).
3) The obtained Entity-ID of the user's home IdP is used as a search criterion in a table of the local TES, and the local TES looks-up the corresponding end-location of the home IdP (steps 5 and 6). Every end-location is an Internet URL. For each IdP, which has already been queried, an entry will exist. Beside the IdP's end-location, each entry contains a trust estimation value for the IdP.
4) If the IdP can not be found in the trust table of the local TES, or if its trust value is lower than a configurable minimum, then the IdP is treated as not trustworthy, and the local TES issues an error message to the end user. If the IdP is found in the trust table, and if its trust value is above the minimum level, then the IdP is treated as trustworthy. Subsequently, the local TES sends a request to the IdP's TES in order to retrieve the IdP's meta data (steps 7 and 8).
5) Finally, the local TES forwards the IdP's meta data to the CSP (step 9).

The content of the trust table is propagated among the TESs by using a self-developed protocol [14], which is similar to OSPF. In doing this, the IdP's meta data is retrieved and updated dynamically by TES.

B. Using DNS NAPTR Record for Discovery Service

As described in the previous section, the Internet DNS is used to resolve the Entity-IDs of IdPs which are URLs. Since DNS is a distributed system, it matches the decentralized nature of the dynamic identity federations proposed here. In decentralized federations, each institution or organization maintains the Entity-IDs of its IdPs in local name server(s). That information is disseminated over the Internet to other entities. When a user requests a CSP service, then he enters his email address via a dedicated user client or a web browser, and the DDS asks its local name server to resolve the Entity-ID of the user's home IdP. Although there is an existing approach for this described procedure [15] using DNS SRV records according to RFC 2782 [16], we chose another solution because DNS SRV records are limited in their capability to map a service onto a fully qualified URL. For example, DNS SRV records cannot contain paths like "http://idp.aai.mpg.de/idp/". Since most Entity-IDs have URLs with paths included, we opted for DNS NAPTR records instead, according to RFC 3403 and RFC 3404 [17]. Also OASIS [18] recommends NAPTR records for resolving meta data via DNS.

A DNS query by means of NAPTR records is employed to map pairs of URNs, URLs onto DNS domain names. The DNS query in turn returns a record that has the following data elements: a) Order, b) Preference, c) Flags, d) Services, e) Regexp and f) Replacement. The elements of this list have the subsequent meaning:

- Order field: A 16-bit unsigned integer specifying the order in which a set of NAPTR records must be processed.
- Preference field: A 16-bit unsigned integer specifying the order in which NAPTR records with equal order fields should be processed. Records with a lower value in the preference field should be processed before records with a higher value.
- Flag field: A <character-string> containing control bits for the rewriting and interpretation of the subsequent fields following the flag field.
- Service field: A <character-string> that specifies the parameters for this service, including the delegation path. The semantics of this field are service-specific.
- Regexp field: A <character-string> that contains a regular expression which substitutes the original input string obtained from the client to construct the proper domain name for address resolving.
- Replacement field: Contains the next domain name to be queried. Depends on the flags field.

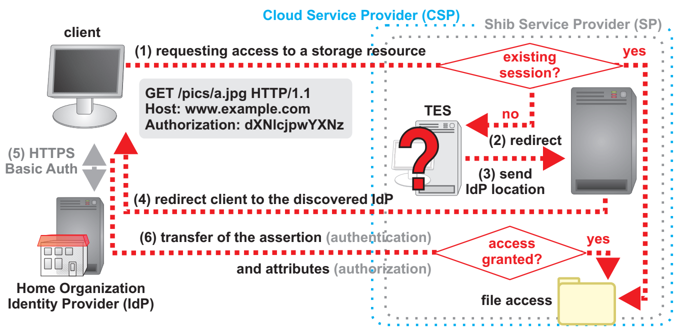
An example of the NAPTR record is as follows:

```
$ORIGIN example.com.
IN NAPTR 100 10 "u" "aai+idp"
"!^.*$!http://aai.mpg.de/idp/!" .
```

This record has an order value of 100 and a preference of 10. The flag "u" is a terminal symbol and indicates that the output of the regular expression is a URI. The next field "aai+idp" indicates that this is an AA service and that the record deals with an IdP instead of a SP. The replacement field means that a domain name such as "mpg.de" has to be replaced by the full URL "http://aai.mpg.de/idp/".

C. Implementation of a RESTful Storage Client

The TES is designed as an extension to Shibboleth which requires a fully-featured web browser and interaction with the end user. In the sign-on process, the Shibboleth SP uses a so-called SAML HTTP POST profile [19] that needs support from JavaScript on the client side to automatically return the web form back to the IdP in order to get the user authentication data. The same JavaScript functionality is used to transmit the SAML assertion with one or more authentication statements back to the SP [20]. This works fine with all current web browsers that have JavaScript enabled. However, for some RESTful API-calls, manual user interaction via a browser are not possible. One solution to this problem is the enhanced client and proxy (ECP) profile defined in SAML [19], but ECP is currently offered only as an experimental extension to Shibboleth, and it is limited to SOAP. Another solution for transmitting the assertion back is an immediate URL-encoding in a HTTP redirect request, but this is not supported by Shibboleth [21]. Therefore, another method that does not need JavaScript was chosen. Here, the HTTP-Response with the SAML assertion is interpreted directly by the REST-client without JavaScript functionality. The client extracts the content of the web form using XPATH and forwards the assertion to the SP. The entire data flow between client, IdP and the CSP is depicted in Figure 6.


Cloud Service Provider (CSP)
Shib Service Provider (SP)
client
(1) requesting access to a storage resource
existing session?
yes
no
TES
(2) redirect
(3) send IdP location
(4) redirect client to the discovered IdP
(5) HTTPS Basic Auth
(6) transfer of the assertion (authentication)
and attributes (authorization)
access granted?
yes
file access
Home Organization Identity Provider (IdP)

```
GET /pics/a.jpg HTTP/1.1
Host: www.example.com
Authorization: dXNlcjpwYXNz
```

Figure 6. Data flow of federated access to cloud based storage solutions

As in Amazon S3, a base64-encoded authorization header is included in the HTTP request from the client to the CSP. The whole data flow comprises 6 steps which are described below:

1) In step 1, the client transmits an access request for file "a.jpg", for example, via a RESTful HTTP message. Its authorization header is compliant with the basic HTTP authentication process described in [22] and therefore uses HTTPS.
2) On the CSP side, first a check to determine if a session exists is performed. If none exists then the HTTP request is redirected to the TES. If a session already exists file access is granted.
3) If there is no session then the user name contained in the HTTP header (e.g., user@institute-a.mpg.de) is checked by the TES to discover the user's home organization and his home IdP. After having found the user's IdP, the TES sends the IdP's location to the CSP.
4) Subsequently, the client is redirected to the discovered IdP.
5) If the client was not previously authenticated at the IdP (no existing session at the IdP), the authorization header is used to authenticate the client.

6) After a successful authentication, the IdP transmits a response which contains the SAML assertion as a hidden HTML input field (SAML HTTP POST profile). Since our RESTful storage client may work independently of a web browser, the value of the hidden input field is extracted by using XPATH and forwarded to the CSP with a POST request. If the authentication was successful, the client is redirected again to the resource "a.jpg" which was requested by the end user, and the data flow cycle is completed.

The delegation of AA to the IdP has an important advantage: if a client accesses multiple cloud storage providers, within the same session, the client does not need to authenticate again. Hence, a single sign-on solution (SSO) is achieved. The same holds for private clouds, as well as for services beside storage, thus allowing for a flexible and comfortable exploitation of the cloud paradigm.

VI. Conclusion and Future Work

This paper describes the usage of dynamic identity federations together with a trust estimation system as an extension to Shibboleth. Furthermore, the integration of dynamic federation into a RESTful cloud storage environment is presented. A dynamic federation model allows for the discovery of the users' home IdPs by means of DNS NAPTR queries. In a distributed storage cloud, files are kept at different sites, and it is necessary to enhance the user client for proper handling SAML assertions. In our model, these assertions are contained in the HTTP-response and do not require web browsers which have JavaScript enabled. Furthermore, the model can be used in private clouds and we are currently evaluating a private storage cloud based on Eucalyptus Walrus [23] which is Amazon S3-compatible. Our next goal is to modify the existing client software and tools of Walrus to support federated authentication. Finally, because our proposed solution is not yet able to provide virtual file systems for common operating systems, future work is needed to adapt WebDAV clients to virtualize the access to dynamically federated cloud storage. Existing WebDAV clients support redirections without problems, but the extraction of assertions and the handling of the subsequent HTTP POST has still to be implemented. Last but not least, authorization issues beyond the available access control scheme of Shibboleth are not addressed yet. To implement fine-grained access control lists on the file level, e.g., according to the user attributes, it is necessary to integrate dynamically federated storage clouds more closely with the underlying operating systems.

References

[1] "Amazon Simple Storage Service (Amazon S3)." [Online]. Available: http://aws.amazon.com/de/s3/ [2010.06.05]

[2] "Google Storage for Developers - Developer's Guide." [Online]. Available: http://code.google.com/intl/en/apis/storage/docs/developer-guide.html [2010.06.16]

[3] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[4] "Cloud data management interface," SNIA Web Site, December 2010. [Online]. Available: http://cdmi.sniacloud. com/ [2010.06.16]

[5] F. AlShahwan and K. Moessner, "Providing SOAP Web Services and RESTful Web Services from Mobile Hosts," in *2010 Fifth International Conference on Internet and Web Applications and Services*. IEEE, 2010, pp. 174–179.

[6] C. Pautasso, "REST vs. SOAP: Making the Right Architectural Decision," in *SOA Symposium*, 2008, pp. 2009–01.

[7] "Ubuntu one." [Online]. Available: https://one.ubuntu.com/ [2010.06.22]

[8] L. Dusseault, "RFC 4918: HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)," RFC, IETF, June 2007., Tech. Rep.

[9] S. Cantor, J. Kemp, R. Philpott, and E. Maler, "Assertions and protocols for the oasis security assertion markup language (saml) v2. 0," 2005.

[10] "Shibboleth System." [Online]. Available: http://shibboleth. internet2. edu/ [2010.06.16]

[11] R. Lai and K. Chan, "METHOD AND APPARATUS FOR SECURELY INVOKING A REST API," Patent, Mar. 12, 2008, uS Patent App. 12/046,579.

[12] "Mongodb gridfs specification." [Online]. Available: http://www.mongodb.org/display/DOCS/GridFS+Specification [2010.06.15]

[13] "ASPiS: Architecture for a Shibboleth-Protected iRODS System." [Online]. Available: http://mykcl.com/iss/cerch/ projects/completed/aspis.html [2010.06.18]

[14] Y. Xiang, J. Kennedy, H. Richter, and M. Egger, "Network and Trust Model for Dynamic Federation," The Fourth International Conference on Advanced Engineering Computing and Applications in Sciences. IARIA, 2010.

[15] S. Rieger and T. Hindermann, "Dezentrales Identity Management für Web- und Desktop-Anwendungen," in *Proc. 1. DFN-Forum Kommunikationstechnologien, Kaiserslautern 2008. Gesellschaft für Informatik, Bonn, 2008; S. 107-116*.

[16] A. Gulbrandsen, P. Vixie, and L. Esibov, "RFC2782: A DNS RR for specifying the location of services (DNS SRV)," *RFC Editor United States*, 2000.

[17] M. Mealling, "RFC3403: Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database," *RFC Editor United States*, 2002.

[18] S. Cantor, I. Moreh, S. Philpott, and E. Maler, "Metadata for the OASIS Security Assertion Markup Language (SAML) V2. 0," 2005.

[19] S. Cantor, J. Hughes, J. Hodges, F. Hirsh, P. Mishra, R. Philpott, and E. Maler, "Profiles for the oasis security assertion markup language (saml) v2. 0," 2005.

[20] "SWITCH AAI Expert Demo." [Online]. Available: http://www.switch.ch/aai/demo/2/expert.html [2010.06.22]

[21] "Shibboleth Native SP Assertion Consumer Service." [Online]. Available: https://spaces.internet2.edu/ display/SHIB2/NativeSPAssertion ConsumerService [2010.06.22]

[22] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart, "RFC2617: HTTP authentication: basic and digest access authentication," *Internet RFCs*, 1999.

[23] "Walrus Storage Service." [Online]. Available: http://open. eucalyptus.com/wiki/EucalyptusStorage_v1.4 [2010.06.22]