Service Token for Identity Access Management

WeiYi Liu, Yue Tan, Enwei Zhang
Tsinghua University
Beijing, China 100084.

***Abstract* --** This paper proposes a new token structure for efficient handling of identity access management for online composite software services. With the requirement of "Single Sign-On (SSO)" for atomic services in a given composite service, this token structure binds service attributes, including workflow, providers, users, operator permission, and operation environment, together in its creation process. The token can be viewed as a deciphered string produced by IDP (identity provider) and consumed by SP (service provider). The concept of conference key distribution is also used to distribute the token and to secure the transportation procedure. Furthermore, the Security Assertion Markup Language (SAML) is adopted to support the exchange of authentication and authorization information between SPs and IDPs. Finally, we apply our service token concept to SourceID Liberty 2.0 (an open source implementation for Liberty Alliance Project) as an illustration of its feasibility and practicability.

Keywords: Service, Identity Management, Access Control, Key Generation, Service Token

I. INTRODUCTION

Starting with the concept of SOA (Service-Oriented Architecture) in 1996, it has generally been agreed that offering software as services online is one key direction for software engineering. Many infrastructure-as-a-service platforms such as Amazon's Elastic Compute Cloud (EC2) and Microsoft's Azure are being offered; applications-as-a-service platforms such as Google App Engine are also available.

With these platforms, increasing number of independent online software services are being published on them and they can be used, either as single services or composite services. This results in an increasing demand for Single Sign-On (or SSO) support so that a composite service only requires user to sign on once, yet the component services offered by different providers inside can be used.

Given that a service instance is characterized by attributes such as workflow, providers, users, operators, and operation environment, we would like to propose a new token that includes all these features in its generation for efficient handling of identity access management. Under this new token structure for service, technique of conference key distribution is also used to distribute the token and to secure the transportation procedure. To illustrate the feasibility and practicability of our proposal, it is also implemented in SourceID Liberty 2.0, which is an open source implementation for Liberty Alliance framework.

The rest of the paper is organized as follows. Section 2 surveys related previous research efforts, which include single sign-on techniques, Liberty Alliance initiative, and conference key distribution techniques. Section 3 proposes the definition and formation of the service token for identity access management. Section 4 details our token implementation under SourceID platform. Finally, the paper concludes in Section 5.

II. RELATED WORK

There are three areas of previous research work related to our investigation here. They are: Single Sign-On, identity and Liberty Alliance, and conference key distribution.

Increasing online web services being offered on Internet results in the demand for Single Sign-On (SSO) technology. Traditional SSO methods mainly focus on single domain and realize using uniform identity authentication such as LDAP; all

978-1-4244-5336-8/09/$26.00 ©2009 IEEE

Authorized licensed use limited to: BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE. Downloaded on September 21,2023 at 15:28:46 UTC from IEEE Xplore. Restrictions apply.

subsystems get authentication information from the central center. Examples of SSO solutions in distributed systems include Microsoft's .NET Passport [1], Sun Microsystems' Liberty Alliance initiative [2], IBM and Microsoft's WS-Federation [3], and OASIS and SSTC's Security Assertion Markup Language (SAML) [4]. SAML is used among different domains to communicate security related information. It supports assertion mechanism which can be used to perform authentication and authorization. For services, SAML's Binding [4] defines how to bind SAML to SOAP-based services.

Related to identity, one of the most comprehensive efforts is the Liberty Alliance initiative [2]. Its main objective is to create open, technical specifications that enable SSO mechanism through federated network identification and support a permission-based attribute sharing framework to enable users' control over the use and disclosure of their personal information. The Liberty Alliance initiative has obtained support from over 150 well-known businesses and organizations in the last few years. Open source initiatives for Liberty Alliance such as SourceID are also available.

The Liberty architecture consists of a multi-level specification set with three main components. The first component is the Liberty ID-FF which defines a framework for federating identities and a mechanism for SSO using a federated identity. ID-FF allows a user with multiple accounts at different Liberty-Enabled (LE) sites to link these accounts for future SSO. The second component is the Liberty ID-WSF which defines a web service framework that allows providers to share users' identities in a permission-based mode. ID-WSF offers features such as permission based attribute sharing and identity service discovery. The third component is the Liberty Identity Service Interface Specifications (ID-SIS) that defines service interfaces for each identity-based Web service. ID-SIS tries to ensure interoperability among service providers with respect to the identity information exchange.

Our solution makes use of the conference key distribution such as threshold Asmuth-Bloom Secret sharing cryptography [6][7]. Under conference key distribution, a key management scheme plays an important role in key-controlled cryptographic algorithms. In 1976, Diffie and Hellman [8] invented the concept of public key cryptography, which is also referred to as a key distribution system. There are several conference key distribution systems [9][10][11][12][13] proposed for generating a conference key. In recent years, some conference key distribution methods based on elliptic curve are also invented [13][14][15].

III. GENERATION OF SERVICE IDENTITY TOKEN

In this section, we are going to describe how service identity token should be generated. To do this, we need to firstly define the structure of such token, followed by its supporting architecture for token generation.

A. Definition of Token Structure

To define the token structure for service identity access management, we need to identify what kind of service properties characterizes a service instance.

In the general case of a composite service, we argue that there are at least five attributes that contribute to the differentiation between two service instances. They are:

- I_i, the identity of a user i, (e.g. name or email address)
- SP_1, SP_2, …, SP_N, service providers who offer component services for a given composite service.
- WF, workflow of the given composite service, with SP_1, SP_2, …, SP_N as its component services. Note that here the order or sequence of the workflow needs to be taken into consideration. It is possible to have two workflows, each of which has the same set of component services.
- SP_iOp, operations allowed by SP_i
- SP_iOpEn, operations enciphered by SPs' public key

With these attributes as input, the service token for identity access management can be defined as follows:

Authorized licensed use limited to: BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE. Downloaded on September 21,2023 at 15:28:46 UTC from IEEE Xplore. Restrictions apply.

- *Function*:

  $SP_iOpEn = RSA(SP_iOp)\ |_{privatekeyofSPi}$

- *Token:* $f(I_i, SP_1, SP_2, \ldots, SP_M, WF, SP_1OpEn, SP_2OpEn, \ldots, SP_NOpEn)$

  where f ( ) can be a DES algorithm or other symmetric enciphering algorithms.

The generation of the token is done by IDP according to this f ( ) function.

Each time a user logs into an IDP, he will choose SPs to form a workflow, which we call it a federation according to the Liberty Protocol. The IDP is like a convoker of a conference who will generate the identity token. Different users, workflow, or SP's allowed operations will altogether contribute to form a unique token.

B. Architecture Support

With the service token structure that we propose in the last section, we apply it to the Liberty Alliance architectural framework for identity access management.

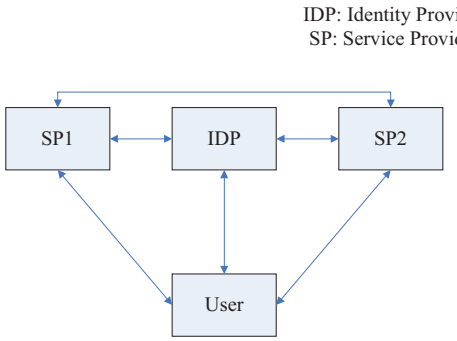

Figure 1: Relationship of SPs, IDP, and Users in Liberty Alliance Framework Using Service Token

Figure 1 shows the relationship among SPs, IDP and users in this framework. The process flow inside can be described as follows:

- A user logs into the IDP and selects the SPs he needs for the workflow (e.g. SP1 → SP2) he is working on.
- The IDP generates the conference key and uses it to encipher the user's identity, SPs, and workflow. After that, the IDP also uses SPs' public key to encipher its allowed operation, just like what we described in the last section.
- For each SP_i in the workflow, the IDP will send the computed result to SP_i, who will calculate the conference key, decipher the workflow and check against the allowed operations for the given user.

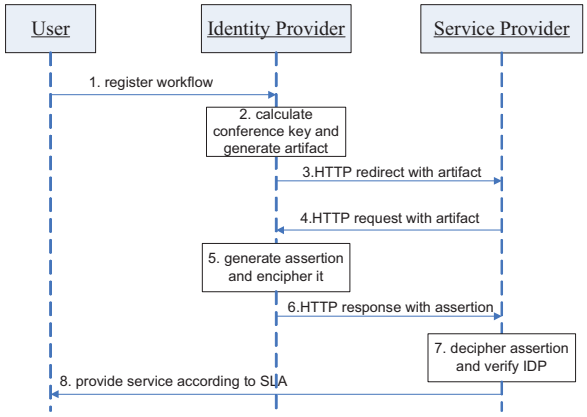

Figure 2. Sequence Diagram of the Architecture Framework

The sequence diagram for the architecture framework of Figure 1 is shown in Figure 2. Each IDP or SP has two sets of private/public keys. One set is used to calculate the conference key and the other is used to decipher its allowed operation. We call these two sets `ConferenceSet` and `RSASet`.

1. A registered user logs in the IDP and selects the workflow WF of SP.
2. The IDP is now the convoker of a meeting that includes all SPs the user selected before and forms a federation. It also computes the key of the conference (`key`) and the public parameter (`Param`) used by SP. The IDP will then add the `Param` to the artifact. One artifact will be connected to one assertion.

Authorized licensed use limited to: BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE. Downloaded on September 21,2023 at 15:28:46 UTC from IEEE Xplore. Restrictions apply.

3. From the IDP, the user can link to one SP. The redirected address will include the artifact that contains the `Param`.
4. The SP requests the assertion that is connected to the artifact.
5. The IDP uses SP's `RSASet` public key to encipher SP allowed operations for the user. Then it will use the conference key to encipher the user's identity, workflow and operations to get an `Identity Token` of the user. This `Identity Token` will be added to Assertions.
6. The IDP returns the assertion to SP.
7. SP gets conference `key` by public parameters `Param` in the artifact and `ConferenceSet` private key. With this key, SP can use it to decipher the workflow and use his own `RSASet` private key to encipher the service it will offer to the user.
8. Finally, SP provides the agreed service to the user.

IV. Implementation

To further investigate the feasibility and practicability of our proposal, we implemented our solution in SourceID Liberty 2.0 Beta, which is an open source for Liberty Alliance Initiative [13]. The conference key distribution technique that we used is one described in [15]. The architecture for SourceID Liberty 2.0 Beta is shown in Figure 3. It is a Java application developed on JBOSS application server.

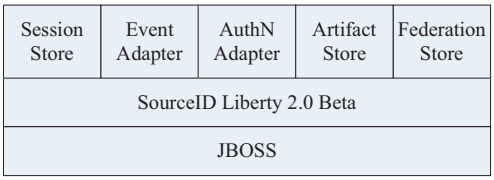

Figure 3: Architecture for SourceID Liberty 2.0 Beta

From our development's perspective, the adapter tier is a critical aspect of the architecture. The adapter tier is a set of Java interfaces that allow developers to customize how data is stored and how interactions with the web application can occur. We implement our solution mainly by changing the adapter tier. To select the implementation for a given adaptor, see the *sourceid-core-config.xml* file.

Session Store — It is the mechanism the SourceID implementation uses to track which users have logged in and logged out. In environment with more than one IDP, this part is also responsible for implementing the domain cookies. An in-memory implementation is provided in (*org.sourceid.idff12.adapters.impl.SimpleSessionStore*). No additional configuration is required to use this default adapter.

Event Adapter — It is a notification mechanism that is used to update the local session system when an SSO event occurs. Separate adaptor instances must be created for handling the behavior of IDP and SP. An implementation of this adaptor must be provided for specific deployments; there is no default implementation provided. As we will discuss later, our solution mainly affects this component.

AuthN Adapter — SourceID uses this interface to retrieve the session identifier provided in a previous call to the 'onSessionCreated' method on the EventAdapter interface. The session identifier is used by SourceID to track state information about a user's current session so that functionality such as Single Log Out works correctly. Separate adaptor instances must be created for handling IDP and SP side behavior.

Artifact Store — It supports the artifact profile defined in the Liberty specification by keeping track of an associated array of artifacts to assertions. An in-memory implementation is provided with SourceID Liberty 2.0 beta (*org.sourceid.idff12.adapters.impl.SimpleArtifactStore*). No additional configuration is required to use this default adapter. This part also defines the format of the artifact, including some operations correlated to the artifact.

Federation Store — It keeps track of information about account linkages and hides all implementation details of mapping user account identifiers to pseudonyms. An in-memory

Authorized licensed use limited to: BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE. Downloaded on September 21,2023 at 15:28:46 UTC from IEEE Xplore. Restrictions apply.

implementation is provided by SourceID Liberty 2.0 beta (*org.sourceid.idff12.adapters.impl.SimpleFederationStore*). In the current version of implementation, it just sets the mapping between different users. No additional configuration is required to use this default adapter.

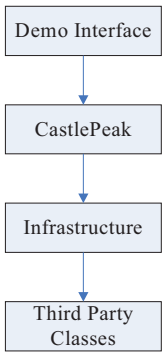

Figure 4: Plumb View of SourceID Liberty 2.0 Beta Architecture

Figure 4 gives a plumb overview of SourceID Liberty project. The calling relations are revealed by the arrows. Third Party Classes is OBE (Open Business Engine) [16] workflow control system which masters SP and IDP interaction. The infrastructure is the implementation for some basic operations like the format of the artifact. CastlePeak is the implementation of other parts including Federation Store, Artifact Store, AuthN Adapter, Event Adapter and Session Store.

Based on the SourceID Liberty 2.0 beta described above, we realize our service identity token proposal to prove the truth and applicability of above design. Our implementation is done by mainly changing the Event Adaptor and Artifact Store.

In Artifact Store, we change the format of artifact. The old format of artifact includes two bytes of TypeCode, 20 bytes of `sourceId` and 20 bytes `assertionHandle`, resulting in a total of 42 bytes. We add 3 bytes to it making it to 45 bytes.

```
SAML_artifact := B64(TypeCode sourceId
assertionHandle extraInf )
```

```
TypeCode := Byte1Byte2

sourceId := Byte3 to Byte22

assertionHandle := Byte23 to Byte42

extraInf := Byte43Byte44Byte45
```

We focus on the calculation done by the IDP which acts as a conference key convoker. SP will do some work to decipher the token and offer the corresponding service to users. In the infrastructure layer, we create SP and IDP classes to store information and do calculation. In the Event Adaptor, we add Identity Token creation and analysis mechanism and Conference Key distribution architecture. The interaction process is as follows:

- At the adaptor of the IDP, we compute the conference key and other parameters for making the artifact.
- At the adapter of the SP, when SP sends an authentication request to the IDP, the IDP will return a HTTP redirection with an artifact. SP will parse the artifact and get the conference key. Then it will ask the IDP for the corresponding assertion connected to the artifact.
- At the adaptor of the IDP, the IDP will create the assertion containing the calculated identity token, and return the assertion to SP.
- At the adaptor of the SP, SP will decipher the identity token and will verify the IDP legality. If the IDP is legal, SP will prepare the agreed service and provide it to the user.

During our implementation, there are two design considerations worth mentioning here. The first one is the transfer of public parameters to SP for the computation of the conference key. As our solution depicts, we add the parameter to the artifact. In order to not influence the other operations of the system, we should also change the operations correlated with the artifact including base64 coding and the parsing of the artifact.

The second one is related to the transfer of the assertion

Authorized licensed use limited to: BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE. Downloaded on September 21,2023 at 15:28:46 UTC from IEEE Xplore. Restrictions apply.

connected to the artifact. We choose to add the token in `<saml:Advice>`, but the format of this field is quite restrictive. Therefore, after we encipher the token to byte, we convert the byte to binary strings which can satisfy `<saml:Advice>` requirements. Of course, at the SP side, we should firstly convert binary strings to byte format and then decipher it.

V. Conclusion and Future Work

In this paper, we have presented a new service token proposal to handle identity access management for composite services. Our proposal captures the characteristics that define a service and uses them in the token generation so that unique service instance can be recognized. We also make use of the conference key distribution mechanism to provide a secure method for message transport among SPs and IDP. It can help SP to verify the IDP's identity in case a false IDP convokes a meeting and steals private information from SP.

Acknowledgement

This work is supported by the 863 project #2008AA01Z12 and the National Natural Science Foundation of China Project #90604028.

References

[1] Microsoft .Net Passport Review Guide 2003. http://download.microsoft.com/download/a/f/4/af49b391-086e-4aa2-a84b-ef6d916b2f08/passport_reviewguide.doc

[2] Liberty Alliance Project. Liberty ID-FF Architecture Overview 2005. http://www.projectliberty.org/resource_center/specifications/liberty_alliance_id_ff_1_2_specifications.

[3] IBM. Web Services Federation Language (WS-Federation) Version 1.1, 2007. http://www.ibm.com/developerworks/library/specification/ws-fed/.

[4] OASIS. SAML v2.0, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip.

[5] SourceID. Liberty ID-FF 1.2, 2005. http://www.sourceid.org/download/index.cfm

[6] Kamer Kaya, Ali Aydm Selquk and Zahir Tezcan, "Threshold Cryptography Based on Asmuth-Bloom Secret Sharing," Proceedings of ISCIS, Springer-Verlag Lecture Notes in Computer Science, Volume 4263, 2006.

[7] Bo Yang, Shengli Liu and Yumin Wang, "Conference Key Distribution Based on Asmuth-Bloom," Journal of Communication and Cryptography, Volume 4, 1998.

[8] W. Diffie and M. Hellman, "New Directions in Cryptography," IEEE Transaction on Information Theory, Vol. IT-22, 1976.

[9] I. Ingemarsson, D.T. Tang, and C.K. Wong, "A Conference Key Distribution System," IEEE Transaction on Information Theory, Vol. IT-22, 1982.

[10] K. Koyama and L. Ohta, "Identity-Based Conference Key Distribution System," Proceedings of CRYPTO'87, also Springer Lecture Notes in Computer Science, Springer-Verlag, 1988.

[11] K. Koyama and L. Ohta, "Security of Improved Identity-Based Conference Key Distribution system," Proceedings of EUROCRYPT'88, Springer-Verlag, 1989.

[12] E. Okamoto and K. Tanaka, "Key Distribution System Based on Identification Information, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 4, May 1989.

[13] N. Koblitz, A. Menezes and S. A. Vanstone, "The State of Elliptic Curve Cryptography," Designs, Codes and Cryptography 9(2/3) 2000.

[14] R. Schroeppel, H. Oman, S. O'Malley and O. Spatscheck, "Fast Key Exchange with Elliptic Curve Systems," Proceedings of Advances in Cryptology, CRYPTO'95, 1995.

[15] Chou-Chen Yang, Ting-Yi Chang and Min-Shiang Hwang, "A New Anonymous Conference Key Distribution System Based on the Elliptic Curve Discrete Logarithm Problem," Computer Standards and Interfaces, Elsevier Science B.V., Volume 25, Issue 2, May 2003.

[16] Adrian Price. Open Business Engine. http://obe.sourceforge.net/.

Authorized licensed use limited to: BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE. Downloaded on September 21,2023 at 15:28:46 UTC from IEEE Xplore. Restrictions apply.