SAML (Security Assertion Markup Language) Security Model for RESTful Web Services

By:

Shazia Sadiq 352-FBAS/MSCS/F07

Supervised by:

Prof Dr.Muhammad Sher

Department of Computer Science and Software Engineering International Islamic University, Islamabad

2012

A dissertation Submitted To

Faculty of Computer Science and Software Engineering,

International Islamic University, Islamabad

As a Partial Fulfillment of the Requirement for the Award of the

Degree of MSCS

Faculty of Computer Science and Software Engineering

International Islamic University, Islamabad

Date: [date of external examination]

Final Approval

This is to certify that we have read the thesis submitted by Shazia Sadiq, 352-FBAS/MSCS/F07 for the partial fulfillment of the requirements for the degree of the MSCS. It is our judgment that this report is of sufficient standard to warrant its acceptance by International Islamic University, Islamabad for the degree of MSCS.

Committee

External Examiner ______________________

Internal Examiner ______________________

Supervisor ______________________

Prof Dr.Muhammad Sher

Chairman,Department of Computer Science

International Islamic University

Islamabad

Declaration

We hereby declare that this Thesis "**SAML (Security Assertion Markup Language) Security Model for RESTful Web Services**" neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this research with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers especially our supervisor **Prof Dr.Muhammad Sher**. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from any of the training institute or educational institutions, we shall stand by the consequences.

Shazia Sadiq
352-FBAS/MSCS/F07

Dedicated to My Family

Acknowledgement

In the name of Allah who is more gracious and more merciful. First of all we are obliged to Allah Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this Project. My acknowledgment goes to my supervisor Prof Dr.Muhammad Sher with the deepest feeling of great gratitude on behalf of his comprehensible supervision and moral support throughout the study and made my research work easier and achievable.

Most significantly, love and patience of my family make this possible; especially my mother has been a constant source of love, concern support and strength all these years. I am also thankful to my sincere friends for their involvement and support in my study.

Shazia Sadiq

Project in Brief

Project Title:	**SAML Security Model for RESTfull applications**
Objective:	To present a Security Model which can fulfill new needs in Web Service security
Undertaken By:	Shazia Sadiq (352-FBAS/MSCS/F07)
Supervised By:	Prof Dr. Muhammad Sher
Technologies Used:	PHP, Mysql, Apache, AJAX,Jquery
System Used:	Intel® Core(TM) i3-2100 CUP @ 3.10 GHz 2Gb RAM
Operating System Used:	Windows 7 Ultimate
Web Browser Used:	FirFox, Google Chorme
Date Started:	September, 2011
Date Completed:	February, 2012

Contents

TABLE OF FIGURES

TABLE OF ALGORITHMS

TABLE OF TABLES

Abstract

This research showed security model for RESTful application which is latest trend in development and also due to Smart devices usage is becoming popular day by day we need Model Architecture which are efficient and less in cost. During Research and development it is considered that this Security Model will be as light as REST architecture is where as effective enough to handle latest Security Issues. Federated Identity Management which is basic for Single Sign On (SSO) is also one of need in new security models. This Research introduced a Security Model for REST in which it would be able to Handle Federated Identity Management. JSON which is alternative for XML and also becoming popular in development due to easy to understand and human readability it is also important to make sure this new proposed model adopt new technologies for which during development JSON is used, which is in other word fat free version of XML. SAML is open source protocol but it is SOPA base, so our new RESTfull model base on SAML with desire changes according to REST Architecture and also in place of XML, JSON form Data Assertion Packet were sent. In this Research it was also tired that new model would be not having all those vulnerability and security threat which are associated with any HTTP model.

Chapter 1. Introduction

This is first chapter of our Report in this chapter we will describe all the basic area and trends that are related to our research. This chapter will provide you information about importance of research in this area and how this field of research is important to study.

Day by day as technology is becoming used in common, implementation of systems is becoming complex in terms of security reliability and complexity in platform because as many systems are being integrated more and more setups are becoming decentralized.

Decentralization of system comes with one important issue that is how each system will communicate with other system. Remote Procedure Call (RPC) overcomes this issue to some extent but it also has some limitations like technology uniformity throughout system and coupling for small application or it was quiet manage but as system grows and needs extend corporate requirements changes which made to work with one technology and also their communication interfaces these problems had created need for technology which work like Remote Procedure call but no call procedure should be uniform in system which ever inner technology is used. Here comes the concept of Service Orient Architecture to be used in Development of systems and application.

Service Oriented Architecture (SOA) is basically a design principle which is used to integrate and implement complex system. In this architecture every object is service which is full system in itself so to develop a big system just need to integrate different Services like joining building Block. Obviously there must be some interface of interaction with each other in these services. Main principle of SOA development is Reusability, Granularity, Interoperability and Modularity. By using SOA if gradually decrease your cost of development a application due to just plug-in-play of different services.

Previously Remote Procedure Call (RPC) was the setup to invoke some action on remote system but it was also having its own limitation like platform dependency and couplings. In this whole decentralize System implementation there must be central Register which is having information about all services. So Service Consumer can get information about which Service will be accurate for it to give desire result.

Following figure is graphical introduction of Service Oriented Architecture (SOA)

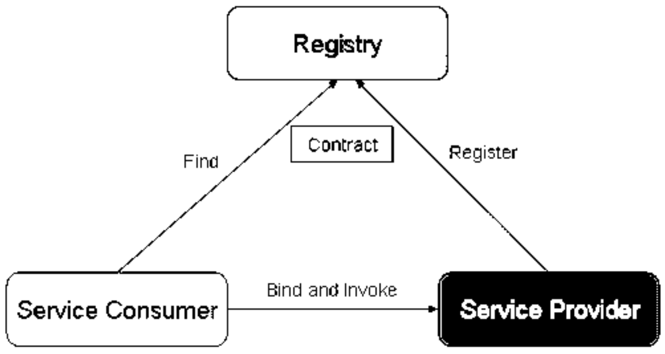


Figure 1 Service Oriented Architecture (SOA)

There are always two types of Services in this Architecture one is Service Provider and second is Service Consumer. Every Service Provider Register itself to Public Register which is accessible to every Service Consumer of System. When any Service Consumer needs some information from other service it first checks register which Service can provide its desire information. After confirmation of Service Provider, Register returns the interface value to Consumer by which information can be gathered from any Web Service

1.1 Web Services

[1]Web Services and SOA are two different things but now days they are associated with each other. Web Service is basically an application that can be discovered accessed and described and it is key requirement of implementation for SOA.

Web Service are available like Web and built using XML which is mostly acceptable protocol in word and by all technologies.XML Syntax and Hypertext Transport Protocol (HTTP), which send XML messages to Web services had made Web Service Interoperable. There are different technologies which help in discovering Web Services like Universal Description, Discovery, and Integration (UDDI). The message syntax for a Web service is described in WSDL (Web Service Definition Language) or WADL. WSDL contains document oriented or procedure oriented information about service. Definitions are separated from their concrete use or instance.

[1] http://www.wstutorial.com/what-is-web-services/

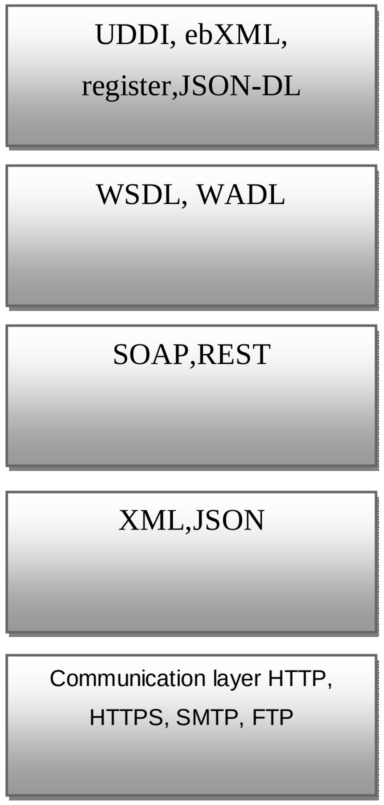

UDDI, ebXML, register,JSON-DL
WSDL, WADL
SOAP,REST
XML,JSON
Communication layer HTTP, HTTPS, SMTP, FTP

Figure 2 Web Services Layers

There are two communication models in Web Services which are known and used.

1. SOAP (Simple Object Access Protocol), this one is a "traditional" way in Web Services on how they communicate. Just like our explanation before, SOAP is using XML to transfer the messages over the network and its functions described in WSDL.
2. REST (Representational State Transfer) is the "new" way to communicate in Web Services. In REST, each resource has URI and communicates using HTTP Header Operations.

1.2 Web Services Security

There are four key concept of any type of security implementation i.e. Authentication, Authorization, Cryptography, Accountability same are for Web Service Security. There are two level of security threats Message Level and Service Level. Web Services are also vulnerable to same type of attacks which are possible on network.

1.2.1 Web Services Security Threats

Web services as works on http layer and protocol, so they are as vulnerable are any web application is especially in case of REST which is following complete architecture base on

URI. Basically security is related to four sections Authentication, Authorization, Confidentiality and Data Integrity. Most common threat for which every programmers need to do counter measure during development are given below.

[1] has mention few vulnerabilities and attacks which are most common in Web 2.0 like

Unrestricted Architecture Browser is considered to be trusted application for any operating system so by default in any script which is running inside browser is also consider trusted. This can exploit user's computer. Only effective encounter against this weakness is to make scripting off in user's browser, which can also some time make legal script to be ineffective.

Cross Site Request Forgery it is vulnerability on both side browser and client and target side as well. Client side application e.g. automatically some time includes few parameters like cookie IP address and domain related credentials. Hacker can mask request under fake labels. Vulnerable web services carry out actions in response to request and rely on saved credentials on the client side without any other verification.

XXS Cross Site Scripting these types of attacks mostly occurs when programmer is not having any mechanism for validating input from client or user end even in URL. Any parameter in input which contains JavaScript code which can leads to any other site and start extracting information from it. Cross Site Scripting is basic from which attack create whole in web service and then start further cracking of data or other confidential information.

To encounter such attacks programmers need to implement filtration of input or data coming from client side.

SQL Injection this attack is also due to lack of filtration on user input but this attack is targeting database. These attacks are related to breaking SQL e.g. break condition for authentication by omitting "AND". To avoid such condition in which attacker can drill in to your system with filtration of input you also need to make custom error message pages so in case of unexpected input or behaviors system will be able to hide itself rather than by displaying server default message all server related configuration get exposed.

XML Poisoning XML is essential component of SOAP and in many cases for REST XML is use to XML entity reference is XML property, attacker manipulates it and disturb whole parsing of XML.

WSDAL and WADAL Scanning and Enumeration WSDAL and WADAL both are use as interface to web service. This is very sensitive information and helps attacker to exploits any application

Replay Attack when old message is sent again and again by attacker it is called replay attack it become very critical in case when message was related to some transaction of e-commerce.

Digital signature cannot prevent any message from being reused as they can be resent by attacker. For protecting any message from reused few element needs to be added in message, like timestamp which shows when this message was actually created and also some “nonce” a randomly generated number which can inform server when message with particular nonce is old or new. Replay attack also cause Denial of Server when done with high frequency. Session hijacking is also related to replay attack in which attacker try to find out any Session ID and then use it for gaining access to other available resources by showing itself in same session.

There are other attacks as well which are related to network in general and also to Web Services like Man in Middle attack, IP spoofing, Denial of service attack, Phishing etc. [2]

Web Services Layer	**Attacks and Threats**
Web Services in Transit	Spoofing, Sniffing, Replay
Web Services Engine	Buffer overflow, XML parsing ,DoS
Web Services Deployment	Fault code leaks, Authentication and Authorization Issues and Certification issues
Web Service User Code	Data tempering, WSDL probing, Data type mismatch, Brute force, Information leakage

Table 1 Web Service Threats

1.2.2 Web Services Security Standard

[2]SOAP and REST by itself are not having any security specifications. So have to fallow other mechanism to provided security in messages. The major standards organizations are the W3C and OASIS, formally known as the Organization for the Advancement of Structured Information Standards. W3C tends to be the home for core XML Web services standards, including SOAP, XML Digital Signature (DSIG) and XML Encryption. OASIS tends to have higher-level standards. For example, WS-Security specifies how to use XML DSIG and XML-Encryption to secure the content of SOAP messages. WS-Security also provides a framework for carrying crypto material (such as keys) and identity information (such as a SAML assertion about the sending client), collectively known as "tokens."There are more than 40 Web Services security Standards Right now which are working at various stages of development and implementation. Here I will mention few of them.

[2]http://www.computerworld.com/s/article/99432/Sorting_out_Web_services_security_standards?taxonomyId=17&pageNumber=2

1.2.3 eXtensible Access Control Markup Language (XACML)

XACML is XML-bases language for access control that has been standardized in OASIS.XACML describes access control policy language and request/response language. The policy language is use to express access control police. The request/response language expresses queries about whether a particular access should be allowed and describes answers to those queries. New Version of XACML compliment SAML like XACML policy can specify what a provider should do when it receive SAML assertion, and XACML-bases attributes can be expressed in SAML.

1.2.4 WS-Security

It is open source standard authored by IBM, Microsoft and VeriSign. This standard describes how to associate XML encryption and XML signature services with XML messages in web services. It is use for message level security. It lets addresses message level security through binding security tokens to XML messages. These security tokens represent claims made by and or on behalf of a service requestor and may be used by the service provider of message-level authentication, authorization confidentiality and integrity services. This protocol allows token to be signed by both service requestor and provider which made its integration flexible.

1.3 Federated Identity Management System

It is somehow centralized identity management or authentication after which each service could share resource with other service. In other words it is also called as Single Sign On.

1.3.1 Security Assertion Markup Language (SAML)

Now days either it is Web Application or Web Service SAML token is most interoperable token format. SAML uses Assertion that is similar to WS-Security “Claims”. SAML is specific in use, like Single Sign On. SAML assertions have profile and bindings for each use case. SAML Assertion may be taken as XML security token representing authentication authorization, and attribute statements. SAML Assertion is bound to specific use cases like Single Sing On for Browser. SAML basically defines rules of data or can say what should be transfer it doesn’t not deal with how it should be transfer.

1.3.2 Liberty Alliance[3]

Liberty Alliance is industrial consortium defining standards for federated identify- including enabling simplified sign-on through federated network identification using current and emerging network access device, as well as supporting and promoting permission-based attributes sharing to enable a user's choice and control over the use and disclosure of his/her personal identification. It has released Frameworks that address Federation (since contributed to OASIS for the SAML standard), Identity Assurance, Identity Governance, and Identity Web Services, as various services applications. It has also been active in privacy and policy issues relative to identity[4].

1.3.3 Shibboleth[5]

Shibboleth is project within the Internet2 higher education consortium to develop technical and policy frameworks and an open software system for the sharing of online resource among researchers, professors, and student. The Shibboleth System is standards based, open source software package for web single sign-on across or within organizational boundaries. It allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner. The Shibboleth software implements widely used federated identity standards, principally OASIS' Security Assertion Markup Language (SAML), to provide a federated single sign-on and attribute exchange framework. Shibboleth also provides extended privacy functionality allowing the browser user and their home site to control the attributes released to each application. Using Shibboleth-enabled access simplifies management of identity and permissions for organizations supporting users and applications. Shibboleth is developed in an open and participatory environment, is freely available, and is released under the Apache Software License.

1.3.4 Single Sign On (SSO)

This is property of system that allows user to login once for one Web Service and then get access to other Web Service without again getting Login Interface. This Property helps user as well as lower the cost of maintain User records for every Web Service Separately. Many solution for Web Browser based Single Sign On haven been introduced so long like .net Passport from Microsoft, Liberty Alliance Project, Shibboleth Initiatives and Open ID. But

[3] SAML Executive Overview OASIS 2005 draft-06
[4] http://en.wikipedia.org/wiki/Liberty_Alliance
[5] http://www.shibboleth.net/

SAML SSO which is emerging new standards and much organization are using it for their Single Sign On setup establishment. Most prominent and well known of them is Google which is using SAML for Single Sign On for Web Browser Profile.

1.4 SAML (Security Assertion Markup Language)[6]

Simple Assertion Markup Language (SAML) is XML base security standard for exchanging information related to authentication or authorization. Most common use of SAML is for Single Sign On (SSO) with help of Identity Providing domain. SAML was introduced by Organization for the Advancement of Structured Information Standards (OASIS) in 2002 with V 1.0 after this it had V.1.1 and V 2.0 in September 2005. There are two type of Assertion in SAML IdP (Identity Provider Initiated) and Service Provider (SP) Initiated

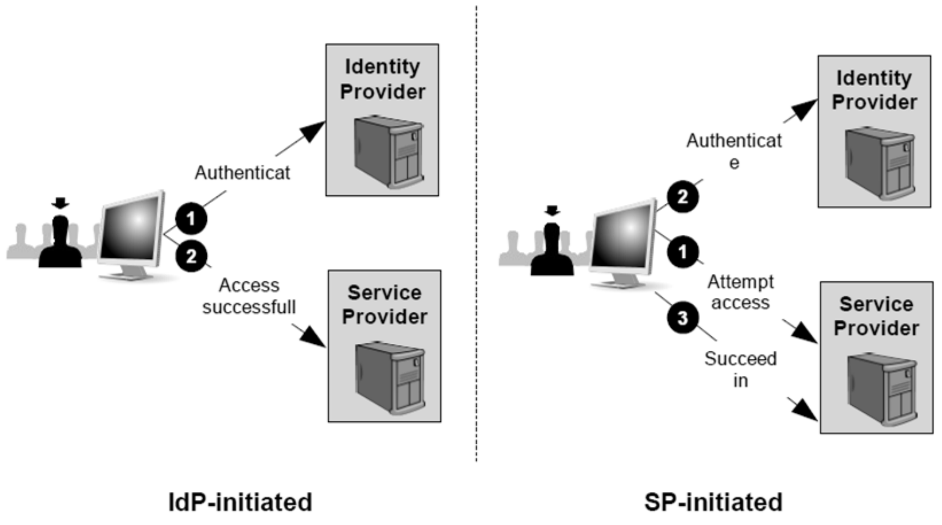


Figure 3 SAML IdP-Initiated and SP-Initiated

1.4.1 SAML Architecture

SAML consist of following Components

- Assertion is statement of principles define by asserted party.
- SAML Protocol defines request of protocol which have their own XML schema
- SAML Binding defines which messaging protocol will be used for SAML protocol.
- Combination of SAML protocol with Binding and Assertion is defined as SAML profile.

[6]http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

1.4.2 SAML Assertion

SAML Assertion is basically a package of information that supplies statements made by SAML authority. Figure 4

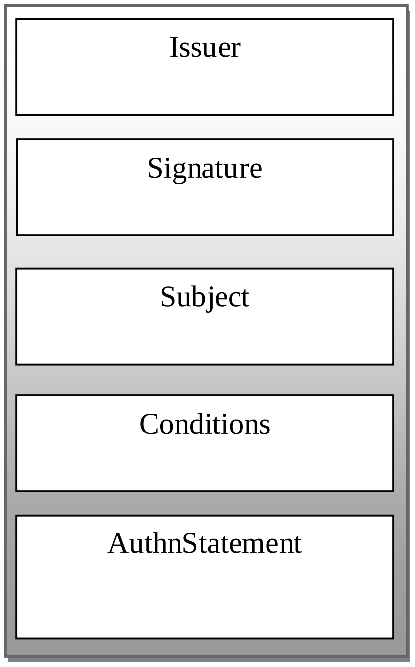


Figure 4 SAML Assertion

- Assertions: SAML allows for one party to assert characteristics and attributes of an entity. For instance, a SAML assertion could state that the user is “John Doe”, the user has “Gold” status, the user’s email address is john.doe@example.com, and the user is a member of the “engineering” group. SAML assertions are encoded in a XML schema. SAML defines three kinds of statements that can be carried within an assertion:
- Authentication statements: are issued by the party that successfully authenticated the user. They define who issued the assertion, the authenticated subject, validity period, plus other authentication related information.
- Attribute statements: contain specific details about the user (for example, that they have “Gold” status).
- Authorization decision statements: identifies what the user is entitled to do (for example, whether he is permitted to buy a specified item).

1.4.3 Protocols

SAML defines a number of request/response protocols. The protocol is encoded in an XML schema as a set of request-response pairs. The protocols defined are.

- Assertion Query and Request Protocol: Defines a set of queries by which existing SAML assertions may be obtained. The query can be on the basis of a reference, subject or the statement type.
- Authentication Request Protocol: Defines a <AuthnRequest> message that causes a <Response> to be returned containing one of more assertions pertaining to a Principal. Typically the <AuthnRequest> is issued by a Service Provider with the Identity Provider returning the <Response> message. Used to support the Web Browser SSO Profile.
- Artifact Protocol: Provides a mechanism to obtain a previously created assertion by providing a reference. In SAML terms the reference is called an “artifact”. Thus a SAML protocol can refer to an assertion by an artifact, and then when a Service Provider obtains the artifact it can use the artifact Protocol to obtain the actual assertion using this protocol.
- Name Identifier Management Protocol: Provides mechanisms to change the value or format of the name of a Principal. The issuer of the request can be either the Service Provider or the Identity Provider. The protocol also provides a mechanism to terminate an association of a name between an Identity Provider and Service Provider.
- Single Logout Protocol: Defines a request that allows near-simultaneous logout of all sessions associated by a Principal. The logout can be directly initiated by the Principal or due to a session timeout.
- Name Identifier Mapping Protocol: Provides a mechanism to enable “account linking”. Refer to the subsequent sections on Federation.

1.4.4 Bindings

This details exactly how the SAML protocol maps onto the transport protocols. For instance, the SAML specification provides a binding of how SAML request/responses are carried with SOAP exchange messages. The bindings defined are:

- SAML SOAP Binding: Defines how SAML protocol messages are transported within SOAP messages. In addition it also defines how the SOAP messages are transported over HTTP.

- Reverse SOAP (PAOS) Binding: Defines a multi-stage SOAP/HTTP message exchange that permits a HTTP client to be a SOAP responder. Used in the Enhanced Client and Pro xy Profile and particularly designed to support WAP gateways.
- HTTP Redirect Binding: Defines how SAML protocol messages can be transported using HTTP redirect messages (i.e. 302 status code responses).
- HTTP POST Binding: Defines how SAML protocol messages can be transported within the base64-encoded content of an HTML form control.
- HTTP Artifact Binding: Defines how a reference to a SAML request or response (i.e. an artifact) is transported by HTTP. Defines two mechanisms, either an HTML form control, or a query string in the URL.
- SAML URI Binding: it helps to retrieve data from URI.

1.4.5 Profiles

The core of the SAML specification defines how the SAML requests and responses are transported, however, a number of use cases have been developed that require the formulation of Profiles that define how the SAML assertions, protocols and bindings are combined. Some of these described in detail later on in the document, in summary they are:

- Web Browser SSO Profile: Defines how a Web Browser supports SSO, when using <AuthnRequest> protocol messages in combination with HTTP Redirect, HTTP POST and HTTP Artifact bindings.
- Enhanced Client and Proxy (ECP) Profile: Defines how <AuthnRequest> protocol messages are used when combined with the Reverse-SOAP binding (PAOS). Designed to support mobile devices front-ended by a WAP gateway.
- Identity Provider Discovery Profile: Defines how a service provider can discover which identity providers is a principal using with the Web Server.
- Single Logout Profile: A profile of the SAML Single Logout protocol is defined. Defines how SOAP, HTTP Redirect, HTTP POST and HTTP Artifact bindings may be used.
- Name Identifier Management Profile: Defines how the Name Identifier Management protocol may be used with SOAP, HTTP Redirect, HTTP POST and HTTP Artifact bindings.
- Artifact Resolution Profile: Defines how the Artifact Resolution protocol uses a synchronous binding, for example the SOAP binding.

- Assertion Query/Request Profile: Defines how the SAML query protocols (used for obtaining SAML assertions) use a synchronous binding such as the SOAP binding.
- Name Identifier Mapping Profile: Defines how the Name Identifier Mapping protocol uses a synchronous binding such as the SOAP binding. REST vs. SOAP.

1.4.6 SAML Use Case

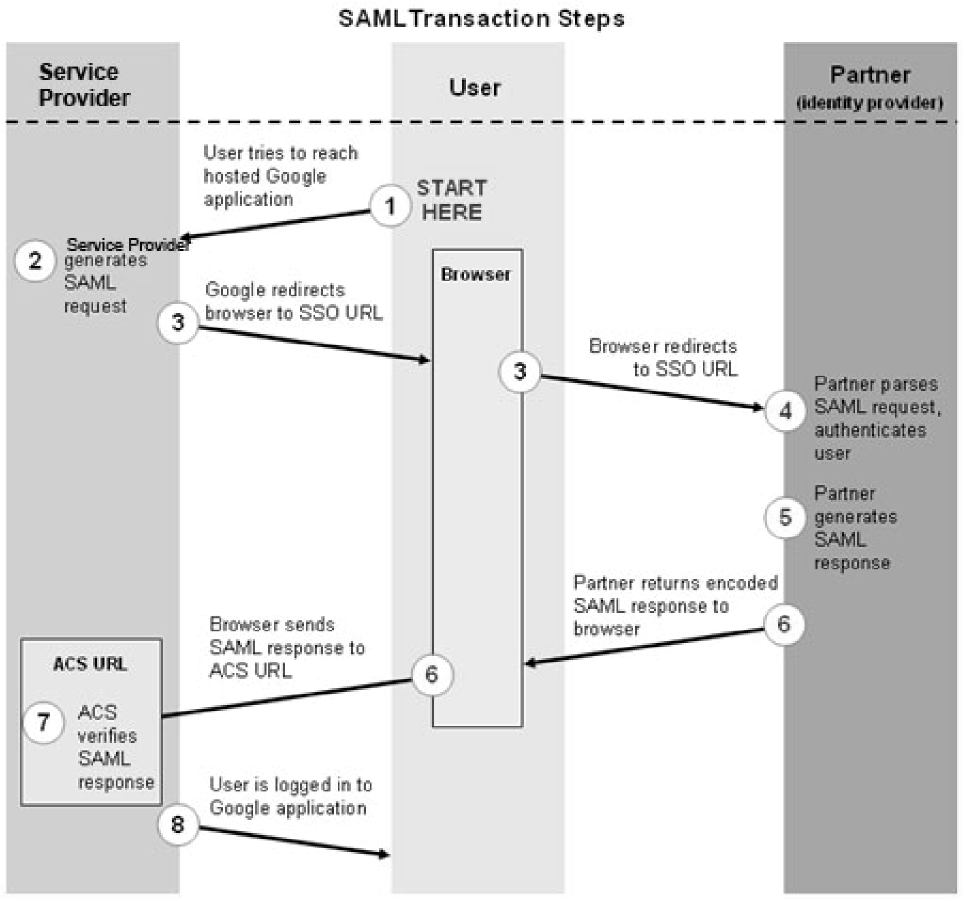

SAMLTransaction Steps
Service Provider
User
Partner (identity provider)
User tries to reach hosted Google application
1 START HERE
2 Service Provider generates SAML request
3 Google redirects browser to SSO URL
Browser
3 Browser redirects to SSO URL
4 Partner parses SAML request, authenticates user
5 Partner generates SAML response
6 Partner returns encoded SAML response to browser
6 Browser sends SAML response to ACS URL
ACS URL
7 ACS verifies SAML response
8 User is logged in to Google application

Figure 5 SAML Transaction Steps

Figure 5 illustrates the following steps.[7]

1. User reaches to hosted application
2. Hosted application generate SAML request and send back to requestor which also contain URL for authentication SSO service provider and client end automatically or by user action will be directed toward desire Identity provider.

[7] http://code.google.com/googleapps/domain/sso/saml_reference_implementation.html

3. Identity provider decodes SAML request and extract desire information lie destination URL, service requestor.
4. Identity provider authenticate user either by valid login credentials or by checking valid session cookies.
5. After successful authentication IdP generates SAML response that contains information that is desire by requested service and by client. For security reason response can be digitally sign by RSA or X.509 certification
6. This SAML response is sent back to client with JavaScript which help it to automatically submit it host from which service was requested.
7. Service Provider verifies response if response was encrypted it will be decrypted with IdP Public key. If the response is successfully verified Server Provider redirects it to destination URL.
8. Now user can access destination URL with login credential.

1.5 REST (Representational State Transfer)

REST is new architecture style which is also seen as alternative to the SOAP. It uses simple HTTP to make interaction between services. Programmers don’t need any type of libraries to implement it. Just simple Browser Address bar could be used to test APIs. REST is stateless Architecture due to which it cannot be used for any type of Authentication and Authorization with extra implementation of technology. Each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server.

Another option for securing REST services is Basic Authentication, which uses a challenge/response model to authorize services. The problem with this method alone is that Passwords are sent in plain text, albeit in Base64 encoding (though Base64 is reversible so is Susceptible to replay attacks and man-in-the-middle attacks).A development of Basic Authentication called Digest Authentication can also be applied to RESTful services. In this case MD5 cryptology is used to secure the password, and is combined with nonce provided by the server to further hide the password from snoopers. This has the added advantage of supporting realm based authentication. It is generally accepted that when used in combination, the three security options mentioned result in secure RESTful services. The approach taken to secure SOAP services is different altogether.

1.5.1 REST Security Features

REST mainly use HTTP Basic It is challenge Response Model, is used by HTTP servers to validate the authentication of Web Browsers. When Client Tries to access the resources protected by Basic Authentication Server give him challenge, after providing correct response Operation will be permitted. Basic Authentication does not provide functionality to deal REALM. it is also vulnerable to Replay Attach due to plain text transmission.

1.5.2 REST vs. SOAP

Using Web Services and SOAP, the request would look something like this:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:body pb="http://www.domain.com/phonebook">
  <pb:GetUserDetails>
   <pb:UserID>12345</pb:UserID>
  </pb:GetUserDetails>
 </soap:Body>
</soap:Envelope>
```

(The details are not important; this is just an example.) The entire message now has to be sent (using an HTTP POST request) to the server. The result is probably an XML file, but it will be embedded, as the "payload", inside a SOAP response envelope.

And with REST? The query will probably look like this:

http://www.domain.com/phonebook/UserDetails/12345.
Principle Comparison between REST and WS-* is shown below
Rest is much faster than SOAP[8], because SOPA requires extra combination with Web Services Protocols. REST also helps in accessing inactive resources where as SOAP clients request operations that is executed on the server. REST can work without any WSDL JavaScript performs the role of WSDL where it is required.REST testing also quiet easy you

[8] http://rudar.ruc.dk/bitstream/1800/2108/1/Web%20services%20-%20SOAP%20%26%20REST.pdf

can test it when service is online. Testing can be performed in same way as testing of any web site could be performed.

1.6 JSON

JSON is light weight data interchange format. It is easy for human to read and write and also easy for machine to parse and generate. It is based on a subset of the JavaScript Programming Language. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.[9]

1.6.1 JSON vs. XML

JSON is Fat-Free Alternative to XML.JSON is much simpler than then XML and it is also not a extensible nut as interoperable as XML is.XML is mainly much complicated for Web Services not also not fit in most of programming languages. JSON is good in representing unordered items. JSON does not provide display capability like XML because it in is not document interchange protocol. When it is matter of representation of entire design of system obviously need to use XML as it is having schema validation semantic etc where as JSON just represent data structure. Right now many organization are shifting towards JSON e.g. Google, Yahoo, PayPal etc.
Table 2 will explain difference in JSON and XML clearly

[9] http://www.json.org/xml.html

JSON	XML

{
"business" :
{
"name" : "One on Wharf",
 "address" :
 {
 "street" : "1 Wharf Road", "town" : "Grantham", "county" : "Lincolnshire",
 },
"phonenumbers" : ["01476 123456", "01476 654321",]
}
}
	<business>
	 <name>One on Wharf</name>
	 <address>
	 <street>1 Wharf Road</street>
	<town>Grantham</town>
	<county>Lincolnshire</county>
	 </address>
	 <phonenumbers>
	 <phone>01476 123456</phone>
	 <phone>01476 654321</phone>
	</phonenumbers>
	 </business>

Table 2 JSON Format vs.XML Format

This table shows clearly how JSON is easy to parse and understand as compare to XML. Brief comparison of JSON and XML is given if following table.

JSON	**XML**
JavaScript Object Notation (JSON) • Textual syntax for serialization of non-recurrent data structures	XML – PO-XML – SOAP (WS-*) – RSS, ATOM
Wire format introduced for AJAX Web applications (Browser-Web Server communication)	Standard textual syntax for semi-structured data
Supported in most languages (not only JavaScript)	Many tools available: XML Schema, DOM, SAX, XPath, XSLT, XQuery
Not extensible (does not need to be)	Everyone can parse it (not necessarily understand it)
“JSON has become the X in Ajax”	Slow and Verbose

Table 3 JSON vs.XML

1.7 Cryptography

Cryptography is becoming essential part of almost every software development especially when it comes to web as the data is transfer from some medium and during transfer it is vulnerable to crack. Many attacks which are known can be encountered by good implementation of encryption. Today almost encryption is everywhere. Passwords are saved in encrypted form. There are three type of cryptographic algorithm involve.

- Secret Key Cryptography
- Public key cryptography
- Hash functions

Figure 6 describers them properly in pictorial form.

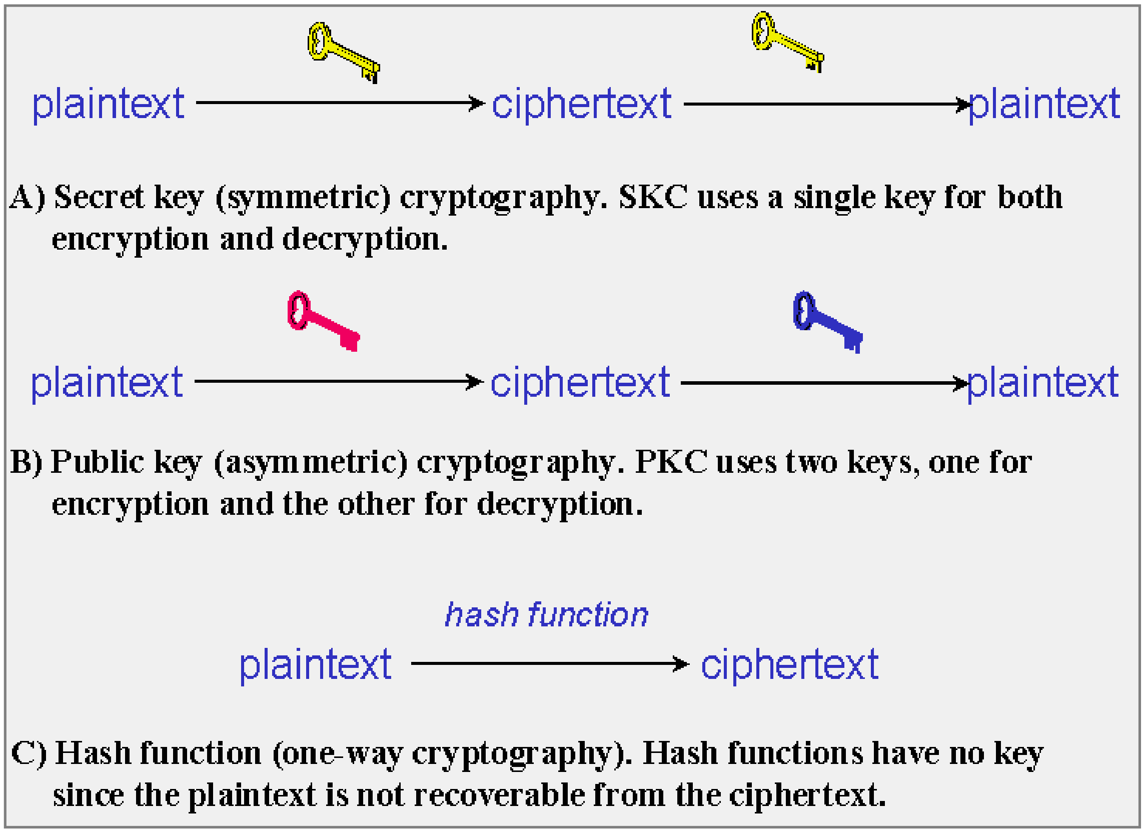

plaintext
ciphertext
plaintext
A) Secret key (symmetric) cryptography. SKC uses a single key for both encryption and decryption.
plaintext
ciphertext
plaintext
B) Public key (asymmetric) cryptography. PKC uses two keys, one for encryption and the other for decryption.
hash function
plaintext
ciphertext
C) Hash function (one-way cryptography). Hash functions have no key since the plaintext is not recoverable from the ciphertext.

Figure 6 Three types of cryptography: secret-key, public key, and hash function[10]

When it comes to the Web Services we have many powerful encryption algorithms. Most of Encryption algorithm requires key which is special for every host or every person if it gets compromised data transfer is vulnerable to be cracked.

Secret key or symmetric key algorithm used same key for encryption and decryption. Whereas public key or asymmetric key cryptography uses key pair one is private and other is public key. Data encrypted with one can only be decrypted by other half of key pair. Every Programming languages support these encryption and decryption algorithm and also help in creating key pairs.

[10] http://www.garykessler.net/library/crypto.html#purpose

1.7.1 PKI[11]

A **public-key infrastructure** (**PKI**) is a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke certificates. In cryptography, a PKI is an arrangement that binds public keys with respective user identities by means of a certificate authority (**CA**). The user identity must be unique within each CA domain. The binding is established through the registration and issuance process, which, depending on the level of assurance the binding has, may be carried out by software at a CA, or under human supervision. The PKI role that assures this binding is called the Registration Authority (**RA**). The RA ensures that the public key is *bound* to the individual to which it is assigned in a way that ensures non-repudiation.

1.7.2 Temporary certificates & single sign-on[12]

This approach involves a server that acts as an online certificate authority within a single sign-on system. A single sign-on server will issue digital certificates into the client system, but never stores them. Users can execute programs, etc. with the temporary certificate. It is common to find this solution variety with x.509-based certificates.

1.7.3 RSA Algorithm[13]

RSA is an algorithm for public-key cryptography that is based on the presumed difficulty of factoring large integers, the factoring problem. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described it in 1978. A user of RSA creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key. The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasibly decode the message. Whether breaking RSA encryption is as hard as factoring is an open question known as the RSA problem.

[11] http://en.wikipedia.org/wiki/Public-key_infrastructure
[12] http://en.wikipedia.org/wiki/Public-key_infrastructure
[13] http://en.wikipedia.org/wiki/RSA_%28algorithm%29

1.8 Thesis Outline

Chapter 1	Describe overview of web services their security protocols and new message format that are being use in web services.
Chapter 2	Is related to literature survey which illustrates work related search area and its limitations.
Chapter 3	Contains problem definition and research objective. It will describe the scope, research and problem domain.
Chapter 4	Defines proposed solution and methodology. It will cover proposed methodology to the problem solution.
Chapter 5	contain results of experiment performed in chapter 4
Chapter 6	Describes conclusion that we had extracted from research and also future extension that is possible from this research.

Chapter 2. Literature Survey

Main purpose of literature survey is to analysis research topic with all current knowledge available and for finding all available solution and their shortcomings. In Literature Review we checked Structure of SAML, vulnerabilities and limitation. We also had compared SOPA with REST which is new Web Service architectural style also had checked Security attacks related to REST and SAML. For making REST fast and efficient we used JSON instead of XML. In Literature Surely we also had studies how JSON is better choice for REST and SAML and how it can help us to achieve our goal.

[2]Has described security model for SOA regardless of which protocol it will be using. it had introduced a Trust Platform Module(TPM) which is embedded hardware which performs all security related functions like encryption etc. This architecture is based on negotiation and validation process. When TPM receive client request it passes from special channel for filtration process. After client passes verification its access level will be received by Token generator, which will leads to creation of token comprising clients' connection information. After which token will be encrypted with provider's public key now whenever client request to any service provider it will attach this token with it. And service provider after decryption of token and validating token grant access according to access level and timestamp which is provided in token. This model also has setup to monitor functionalities of service providers regarding got corrupt with any attack of viruses of Trojan etc. it work like creating one broker which act like client and know what response should client get if it matches with desire result then service is working fine else it is corrupt and it creates alert.

[3]Has technical detail of SAML. It is latest overview document available SAML which is issues by OSAIS. Much of its information is also mentioned in Introduction of SAML as this is basic document for definition of SAML and its architecture. It had mentioned following Bindings HTTP Redirect Binding, HTTP POST Binding, HTTP Artifact Binding, SAML SOAP Binding, Reverse SOAP (PAOS) Binding and *SAML URI Binding*. RESTful Web Service URI Binding will be used as it will help to retrieve data from URI. Profiles Mentioned are *Web Browser SSO Profile*, Enhanced Client and Proxy (ECP) Profile, Identity Provider Discovery Profile, Single Logout Profile, Assertion Query/Request Profile, Artifact Resolution Profile, Name Identifier Management Profile and Name Identifier Mapping Profile. Web Browser SSO Profile uses Authentication Request Protocol and can use any HTTP base Binding.

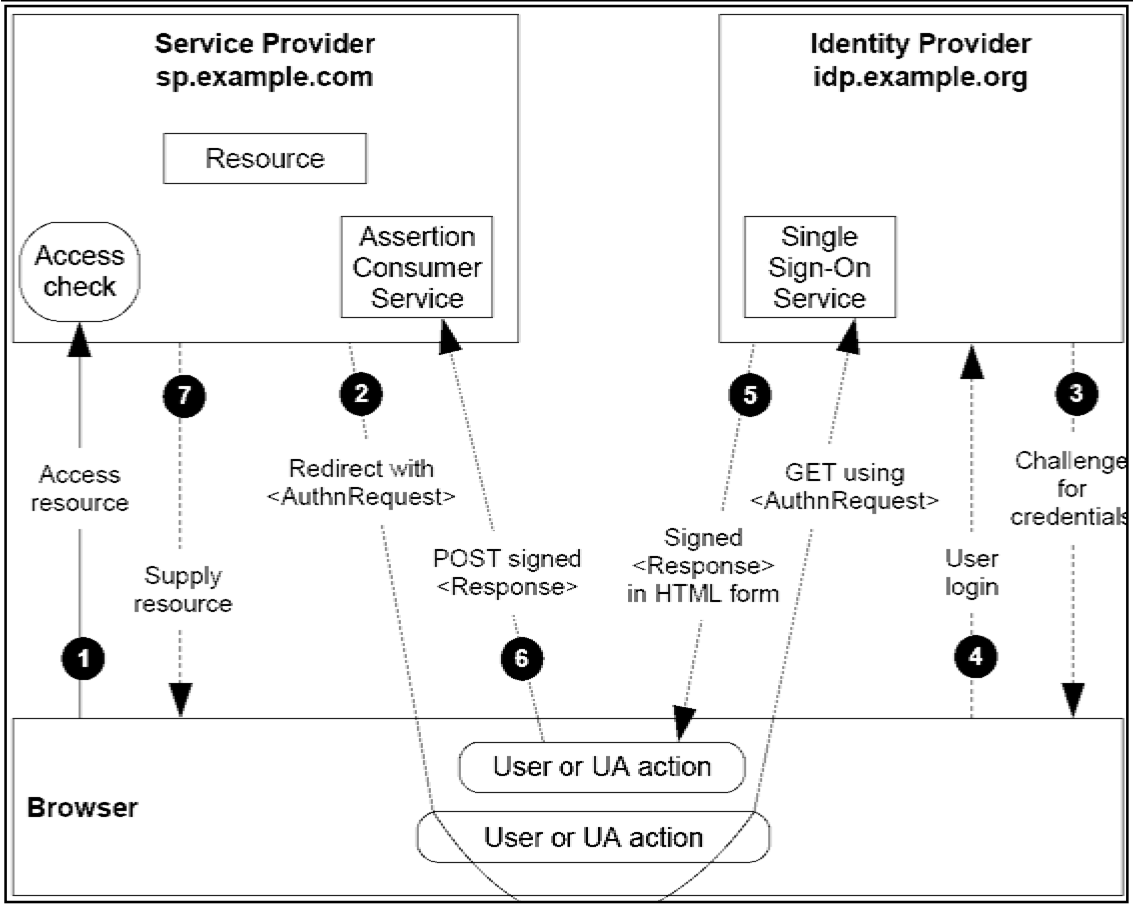


Figure 7 HTTP Redirect Binding and POST Binding

Figure 7 is figure of POST binding that is define in this Overview Document of SAML published by OSAIS.

In [4]Has mention how to use SAML for Single Sign On (SSO).Author had used SAML for Web Authentication with SSO as well. Authors had used Web Browser SSO Profile for its SSO implementation on Web and clarify both scenarios in which Identity Provider Initiate SAML Assertion and also in which Service Provider Initiate SAML Request Author had mention that SAML can be used with HTTP Authentication method which is conventional Authentication method for HTTP and called it Web Browser SSO Profile. This article also had mentioned how this protocol can deal with attack that is related to Web base protocols. Like man-in-middle attacks and spoofing. This article implementation consist of metadata first which helps identity provider to get idea what it is going to get from Service provider or client and what it needs to return back in response to client. For preserving integrity of assertion it could be signed assertion with encryption. Solution for replay attack that this article had mention is that use of attribute named “NotBefore” and “NotOnOrAfter” as shown in Algorithm 1.

```
<samlp:Response xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
Destination=https://mypartner.com/metaAlias/sp ID="ad58514ea9365e51c382218fea" IssueInstant="2009-04-
22T12:33:36Z" Version="2.0">
<saml:Issuer>http://login.mycompany.com/mypartner</saml:Issuer>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
SIGNATURE VALUE, ALGORITHM, ETC.
</ds:Signature>
<samlp:Status>
<samlp:StatusCodeValue="urn:oasis:names:tc:SAML:2.0:status:Success">
</samlp:StatusCode>
</samlp:Status>
<saml:Assertion ID="1234" IssueInstant="2009-04-22T12:33:36Z" Version="2.0">
<saml:Issuer>http://login.mycompany.com/mypartner</saml:Issuer>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
SIGNATURE VALUE, ALGORITHM, ETC.
</ds:Signature>
<saml:Subject>
<saml:NameID>NAMEID FORMAT, INFO, ETC</saml:NameID>
<saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
<saml:SubjectConfirmationData NotOnOrAfter="2009-04-22T12:43:36Z"
Recipient="https://mypartner.com/metaAlias/sp">
</saml:SubjectConfirmationData>
</saml:SubjectConfirmation>
</saml:Subject>
<saml:Conditions
NotBefore="2009-04-22T12:28:36Z"
NotOnOrAfter="2009-04-22T12:33:36Z">
<saml:AudienceRestriction>
<saml:Audience>mypartner.com:saml2.0</saml:Audience>
</saml:AudienceRestriction>
</saml:Conditions>
<saml:AuthnStatement AuthnInstant="2009-04-22T12:33:20Z"
SessionIndex="ccda16bc322adf4f74d556bd">
<saml:SubjectLocality Address="192.168.0.189"
DNSName="myserver.mycompany.com">
</saml:SubjectLocality>
</saml:AuthnStatement>
<saml:AttributeStatement xmlns:xs=SCHEMA INFO>
<saml:Attribute FriendlyName="clientId" Name="clientId" NameFormat="urn:oasis:names:tc:SAML:2.0: attrname-
format:basic">
<saml:AttributeValue>1234</saml:AttributeValue>
</saml:Attribute>
<saml:Attribute FriendlyName="uid" Name="uid" NameFormat="urn:oasis:names:tc:SAML:2.0: attrname-format:basic">
<saml:AttributeValue>the.user@mycompany.com
</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
</saml:Assertion>
</samlp:Response>
```

Figure 8 Sample SAML Assertion

[5] has done analysis of SAML as it is known Web Browser Profile for Single Sign On. Authors of this article have shown different Authentication flaws in this protocol. First and foremost important thing which is mentioned is reuse SSL session which is established for communication from first step till end but it is quiet difficult to achieve due to underlying

technologies limitations. Figure 9 shows the Attack in which “i’ is intruder and showing itself as “sp” to “c”.

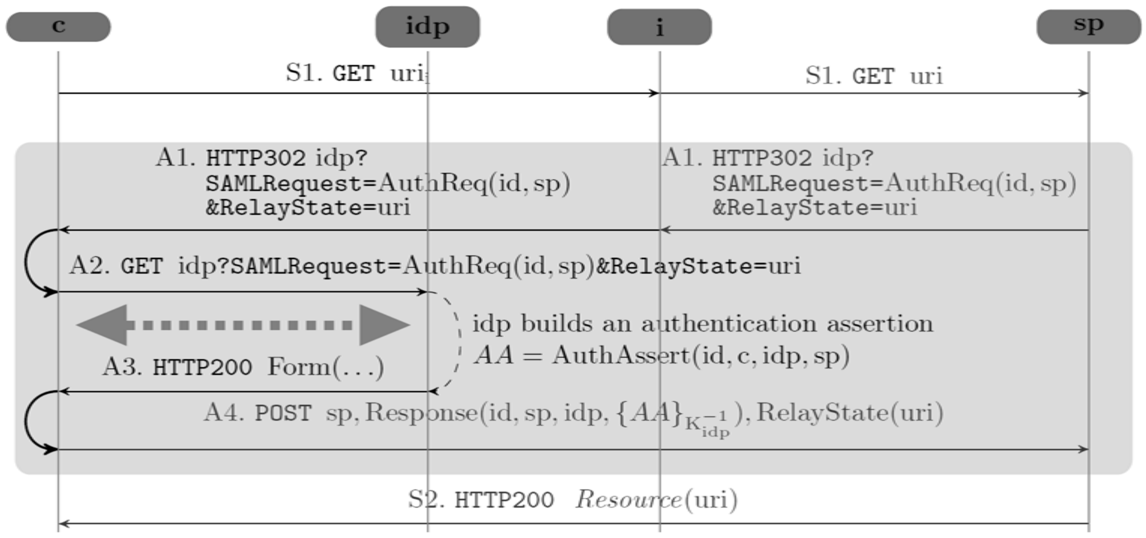

c
idp
i
sp
S1. GET uri
S1. GET uri
A1. HTTP302 idp? SAMLRequest=AuthReq(id, sp) &RelayState=uri
A1. HTTP302 idp? SAMLRequest=AuthReq(id, sp) &RelayState=uri
A2. GET idp?SAMLRequest=AuthReq(id, sp)&RelayState=uri
idp builds an authentication assertion
AA = AuthAssert(id, c, idp, sp)
A3. HTTP200 Form(...)
A4. POST sp, Response(id, sp, idp, {AA}K−1idp), RelayState(uri)
S2. HTTP200 Resource(uri)

Figure 9 Authentication Flaw of the SAML 2.0 Web Browser SSO Pro

As all communication is not on same secure channel also request is generated from same genuine “sp” so request appears with valid digital signature. This flaw in protocol can leads to cross site scripting and cross site forgery attack. Author of this article had deployed these flaws on real life deployment of web Browser SSO especially on Google which is quiet known for SAML implementation. They also had mentioned few ways to encounter this attack one is cookie base in which session cookie is establish but it is not quite difficult to steal cookie. Other solution that they have given is feedback in which IdP inform about what user is being intend to access but these both have drawback other one which is simple and much effective is self signed certification in this process SP generate Authentication Request and its nonce and create its Hash. And whenever client communication it shows same hash value which is only available to SP as secret key is only know to SP.

[6] In this Article Author had comparison REST architecture and WS-*.This comparison was on the base of parameters like Transport Protocol, Payload Format, Service Identification, Service Description, Reliability, Security, Discovery and Implementation Technology.

Conclusion that Author had extracted was for ad hoc integration REST is Good Choice but for Enterprise Development where Cost is not factor in comparison to security and Discovery WS-* is better choice.

Author had mention strengths of REST which are like this

- All major Operating System, Hardware and Programming Languages have compatibility for REST.

- Light weight.
- Less tooling require.
- Registrations to any repository for discovery in not require any more and resource access is base on URIs.
- Stateless but cacheable so large number of requires can be serve.

With all these strengths REST have some shortcomings as well which are given below.

- Limited size of URI which is base of REST although with POST verb this type of problem cannot happen but in case of GET there are chances of this.
- REST is only compatible to HTTP which is common in use but still WS-* have variety of protocols e.g. SMTP,TCP etc

Following table will elaborate comparison done by this author. Two metaphor are used *AD(Architectural decision)* and *AA(Architectural Alternatives).*

Architectural Decision **and AAs REST WS-***	***REST***	***WS-****
Integration Style	*1AA*	*2AAs*
Shared Database		
File Transfer		
Remote Procedure Call	✓	✓
Messaging		✓
Contract Design	*1AA*	*2AAs*
Contact-first		✓
Contract-last		✓
Contract-less	✓	
Resource Identification	***1AA***	***n/a***
Do-it-yourself	✓	
URL Design	***2AA***	***n/a***
“Nice” URI scheme	✓	
No URI scheme	✓	
Resource Identification Semantics	***2AAs***	***n/a***
Lo-REST(POST,GET only)	✓	
Hi-REST(4 verbs)	✓	
Resource Relationships	***1AA***	***n/a***
Do-it-yourself	✓	
Data Representation/Modeling	*1AA*	*1AA*
XML Scheme	✓	✓
Do-it-yourself	✓	

Message Exchange Pattern	***1AA***	***2AAs***
Request-Response	✓	✓
One-way		✓
Service Operation Enumeration	***n/a***	***3AAs***
By function domain		✓
By non-functional properties and QoS		✓
By organizational criterion(versioning)		✓
Total number of Decisions, AAs	***8,10***	***5,≥ 10***

Table 4 Conceptual Comparison between REST and WS-* Summary

Above table has defined Conceptual Comparison of REST and WS-*.Table in Tables has showed that WS-* many choice but have strict conceptual area. But REST has higher number of decisions alternative.

[7] Had performed some experiments on SOAP and REST with different technology and come to conclusion that overall REST with Security is faster than SOAP. Following figure of graph which is taken from this report show how much there is performance difference between SOAP and REST.

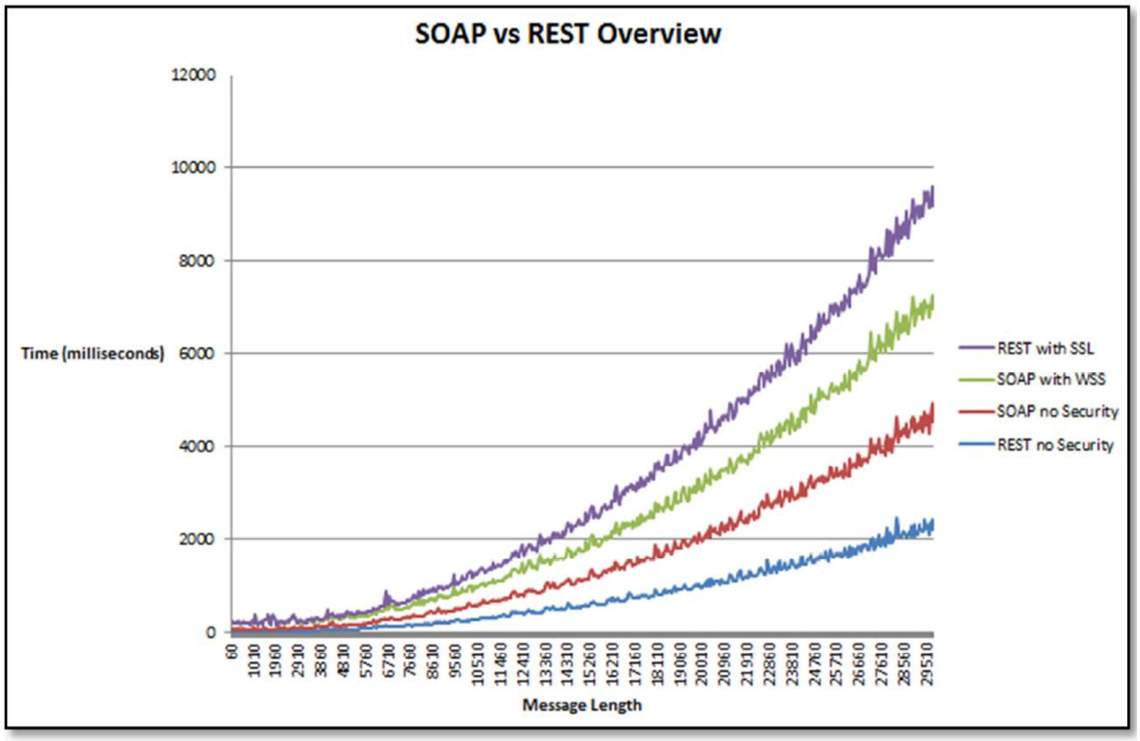


Figure 10 SOAP vs. REST performance overview

[8] has compared two data interchange formats and come to the conclusion that JSON is faster and use less resources that XML and eventfully best practice for smart devices related application and services.

Following picture shows the utilization of resource n case of XML and JSON

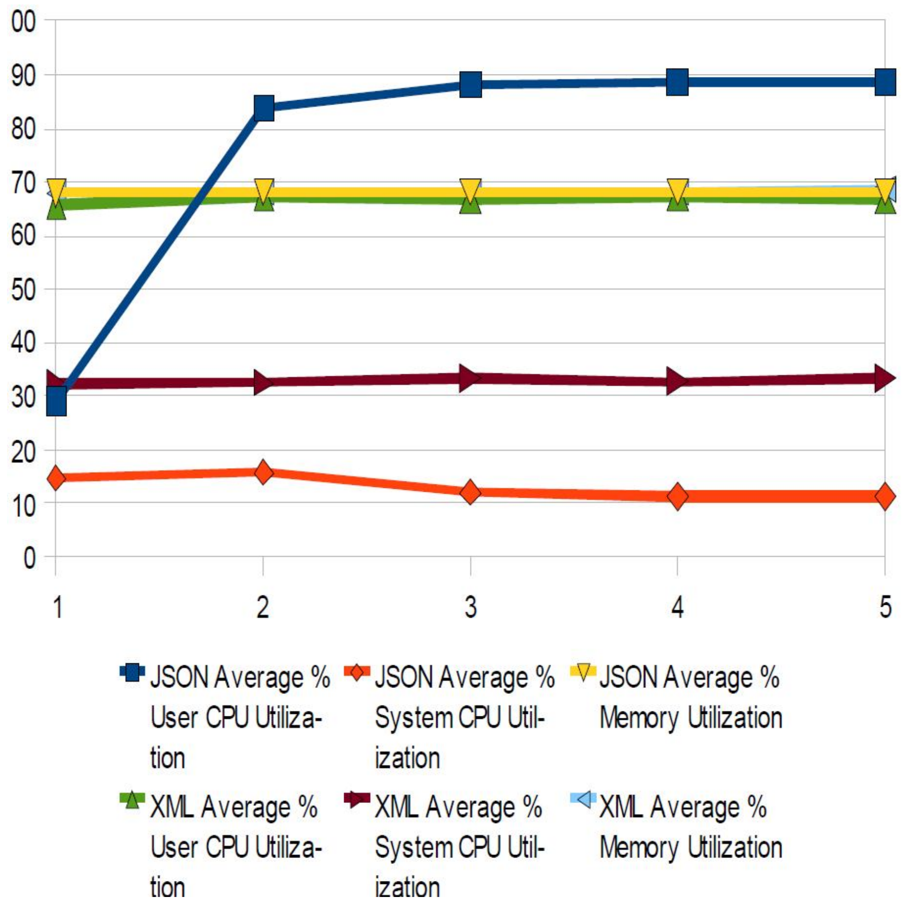


Figure 11 JSON Vs XML Resource Utilizations

For experiment two scenarios were considered one with accurate average measurements because of the high number of encoded object transmission. And Second scenarios with fine grinned observation of impact of fewer transmissions for each measurement.

This table shows the result of both scenarios for JSON and XML

Scenario/Measure	JSON	XML
Scenario 1 Total Time	78.26 seconds	75.77 minutes
Scenario 1 Average Time per Object	0.08ms	4.55 ms
Scenario 1 Average User CPU Utilization	86%	55%
Scenario 1 Average System CPU Utilization	13%	45%
Scenario 1 Average Memory Utilization	27%	29%
Scenario 2 Total Time for 100,000 Objects	7.5 seconds	310 seconds
Scenario 2 Average Time per Object	0.08ms	3.1ms
Scenario 2 Average User CPU Utilization	83-88%	65-67%
Scenario 2 Average System CPU Utilization	11-14%	32-33%
Scenario 2 Average Memory Utilization	68%	68%

Table 5 High-Level Results and Observations for JSON and XML

[9] Has created as Sing Sign On Model for Web Services base on Password Scheme. This model was consisting of Authorization Server, Accounting and Credential Database. It is based on username/password authentication. Client or end user provides credential for authentication and authorization. If user is successfully authenticated service token is generated for user session which will also work for authorization and this token will be used for other web services access also will same token. This model scheme consists of key generation for encryption and decryption due to security purpose. Client and Authentication Server exchange key between with each other. Figure 12 shows Key exchange process in steps.

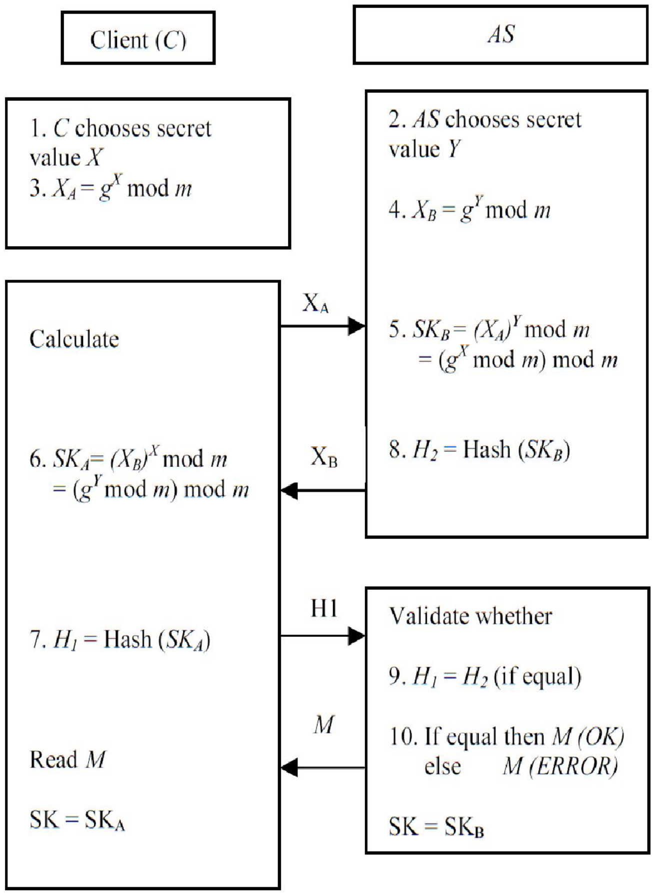

Client (C)
AS
1. C chooses secret value X
3. $X_A = g^X \bmod m$
2. AS chooses secret value Y
4. $X_B = g^Y \bmod m$
X_A
Calculate
5. $SK_B = (X_A)^Y \bmod m = (g^X \bmod m) \bmod m$
X_B
6. $SK_A = (X_B)^X \bmod m = (g^Y \bmod m) \bmod m$
8. $H_2 = \text{Hash}(SK_B)$
H1
7. $H_1 = \text{Hash}(SK_A)$
Validate whether
9. $H_1 = H_2$ (if equal)
M
10. If equal then M (OK) else M (ERROR)
Read M
SK = SK_A
SK = SK_B

Figure 12 Secret Key exchange Process

After successful exchange of key between them, user or client start to send username and password encrypted form. After successful authentication Authorization Server created Token which is base on user’s password and SecID and also involves Hash encryption. Client use it for access to any Web Service when Web Service get Request from Client with Token it send Token of Authorization Server for validation of token as token is Authorization Server Database so can easily verify either it is valid token or not. After receiving verification from Authorization Server, Web Service grants access to Client. This scheme had made Man–in middle attack encounter as process after key exchange is encrypted with key and process of key generation depends on sender so keys will not match. And also due to expiration of token after some specify interval it will also not create any type of replay attack.

[10] has used Username token of WS-Security Model which is being use for SOPA in REST. Author had used header to transplant this username token in REST and also used “secondary password” to enhance security. Algorithm 1 is the Authentication Process.

1) After client sends a request, the server gives the following challenge:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: WSSE realm="User Data",
profile="UsernameToken", nonce=" ecb687b278665c12"
```

2) Client receives the server's challenge and
calculates the WSSE digest:

```
# Information from the challenge
REALM="User Data"
NONCE="ecb687b278665c12"
# Information of HTTP method and URI from client request
Method="GET"
URI="/personal.html"
# Information of client
USER="Lcroy"
PASSWORD="hello world"
CNONCE="0ce167a8e0c1c5c6"
CREATED="2008-03-11T08:00:00Z"
NC="0001"
# UsernameToken's digest R1 calculated by client
R1=Base64(SHA1(NONCE,CREATED,PASSWORD))
# Secondary password R2 calculated by client
H1=MD5 (USER, REALM, PASSWORD)
R2=MD5 (H1, NONCE, NC, CNONCE, METHOD, URI)
```

3) Client sends response to the server:

```
GET /personal.html HTTP/1.1
HOST: www.test.com
Authorization: WSSE profile="UsernameToken"
X-WSSE: UsernameToken Username="lcroy",
PasswordDigest=R1,
Cnonce=" 0ce167a8e0c1c5c6",
Created=" 2008-03-11T08:00:00Z",
SecondaryPassword=R2,
Realm="User Data",
Nonce="ecb687b278665c12",
Uri=" /personal.html",
NC="0001"
```

Algorithm 1 Username Token Authentication Process

Figure 13 shows the flow diagram on this Username Token schema.

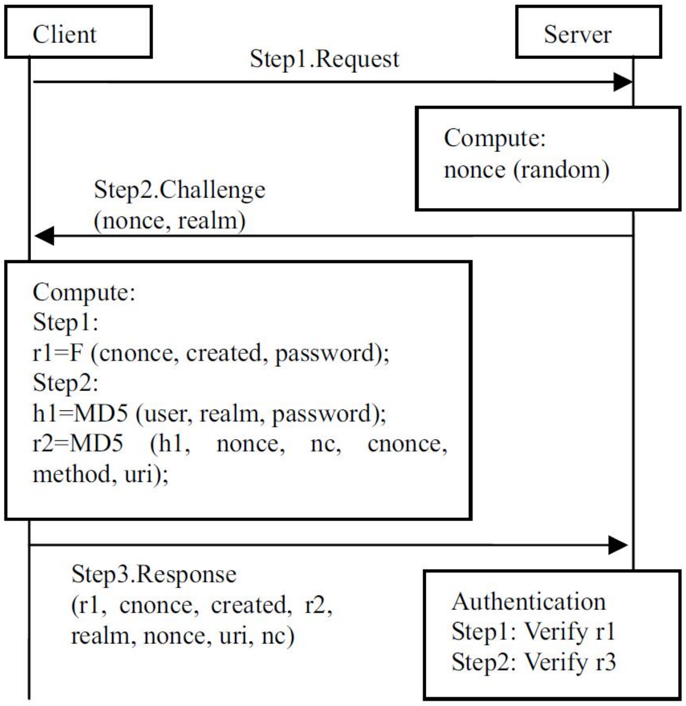

Client
Server
Step1.Request
Compute:
nonce (random)
Step2.Challenge
(nonce, realm)
Compute:
Step1:
r1=F (cnonce, created, password);
Step2:
h1=MD5 (user, realm, password);
r2=MD5 (h1, nonce, nc, cnonce, method, uri);
Step3.Response
(r1, cnonce, created, r2, realm, nonce, uri, nc)
Authentication
Step1: Verify r1
Step2: Verify r3

Figure 13 Authentication Process Flow

First server save secondary password values, Client sends request to Server, Server give challenge to client with two parameters nonce and realm. Releam is use to indentify group or resource that client wants to access. Client knows that Server is using Username token scheme so it will generate cnonce and create encrypted value of password with current data and other parameters. Client will also generate secondary password value which is generated from releam password and other parameters as shown is figures “nc” is number sequence which shows which response sequence is this and same number shows this response is duplicate. After server receives response authentication process starts which consist of comparing password and secondary password, after successful match resource will be granted to requesting client.

[11] Has proposed a central security system which can handle authentication and authorization for Restful services. Authentication and authorization is token base which is generated by security service provider. Whenever any client which could any web browser or any other web application access web service and access is without token client is redirect to security service and after authentication setup token is provided to client which can be used for multiple calls. Now in every call to any web service token will be added with other requesting parameters for authorization purpose roles groups etc are define at security center service which can identity what is accessible to this client. Whenever web service receives, client request with token. Web service sent token to security server for validation purpose and

after successful validation resource will be granted. For encounter of eavesdropping secure channel of communication is used. Figure 14 is shows authentication procedure that is defined in this report with RESTful architecture. Figure 15 Authorization with same setup and base on role base system

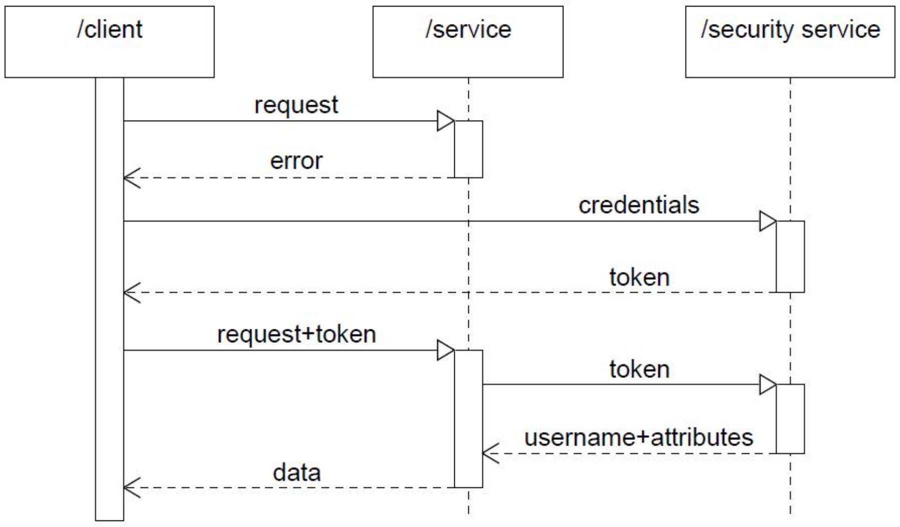


Figure 14 Authentication Request Process

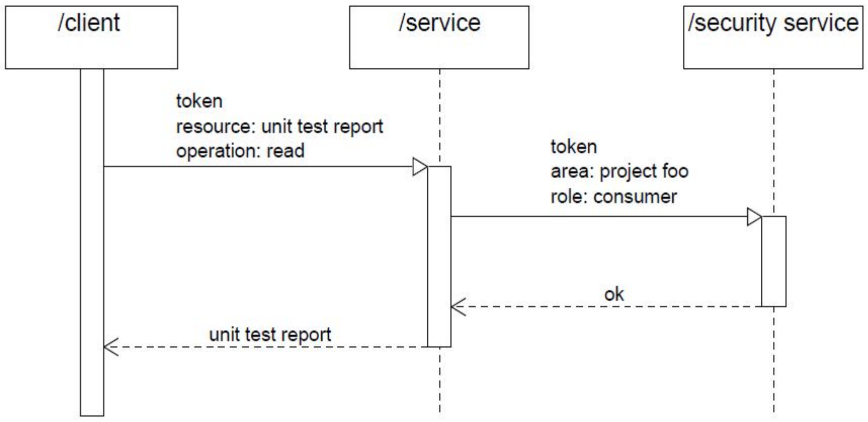


Figure 15 Authorization Process

[12] Has mentioned model for SSO with JSON and SOAP. This made this architecture light weight. This Article had mention the case for B2B setup in which novice user is not getting involved and interface is not public access interface. Both Parties i.e. Alice and Bob have Key pair sharing with each other for communication.

Whole Scenario is shown in Figure 16

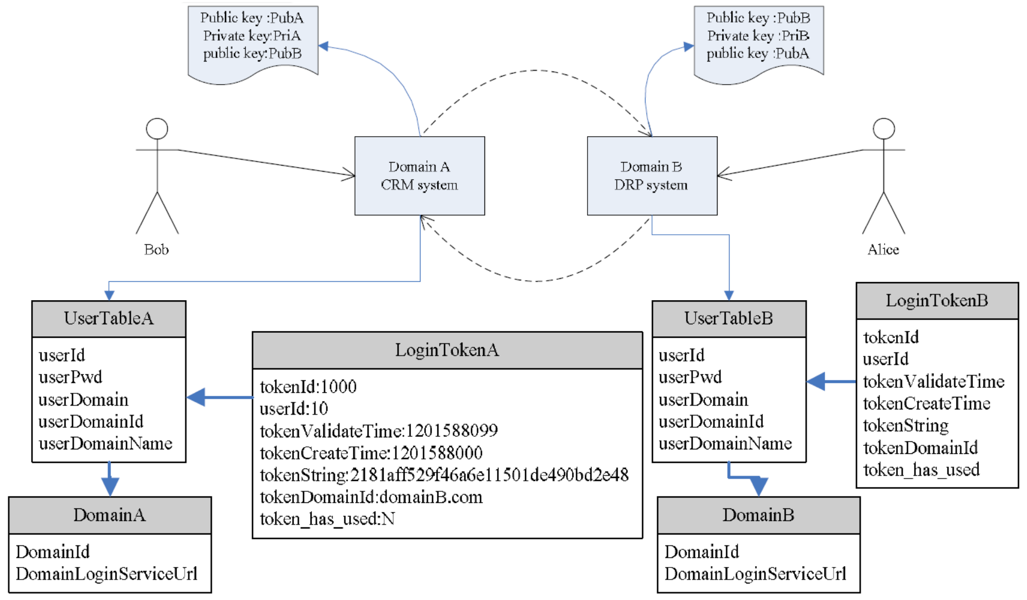


Figure 16 B2B SSO Scenario

Both Alice and Bob are managing Public and Private Key for each other and sending data to each other with their corresponding Key encryption which can be called as Both are Service Provider as well as Identity Provider.

[13] Also has shown Architecture for SSO with cryptography. This architecture consist of two phases first is called as “Profile Creation” in which User signup to SSO server and creates it profile and authenticate by given login and password which are encrypted and decrypted by public and private key of SSO server. After this user will input information about applications which are also encrypted. During runt time phase which is second phase of this architecture user authenticated to SSO server, user can select application where he/she previously have authentication and authorization data, after local authentication user get connected to application. They have use Elliptic Curve Cryptography (ECC). As this architecture does not have any key exchange process so man-in-middle attack is not possible. As encryption and decryption are done by taking URLs as public and private key so which also made system easy to use enhanced the privacy. Figure 17 shows define architecture.

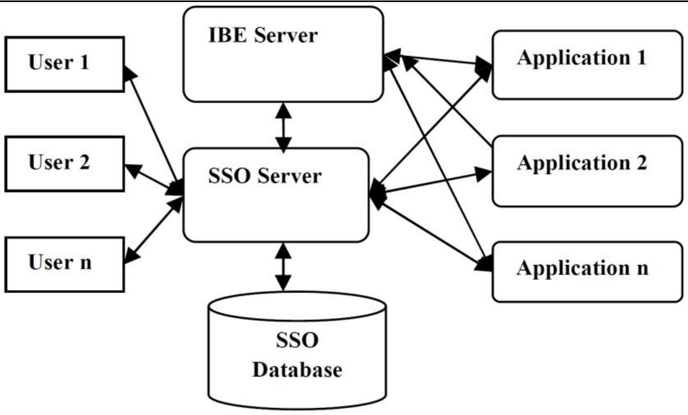


Figure 17 Proposed SSO System Architecture

This architecture is testing with “HTTPAttack” a testing tool and check for availability and response time

[14]Has SSO solution for clouds network. The flow of Cloud based SSO is elaborated in Figure 18, in which user authenticates to the central authentication server, and this authentication server itself supplies the user credential (e.g., username and password) to the appropriate server whenever the user requests to use an application on another server. This module is developed using PHP language which enables user to get them register with this server and store their credentials. This authentication proxy server uses a database to maintain all the credentials for the registered users. Procedure of authenticating users is through cURL and SSL which makes itself robust by creating a secure channel over an insecure network. This ensures reasonable protection from eavesdroppers and man-in-the-middle attacks, provided that adequate cipher suites are used and that the server certificate is verified and trusted. When users are creating SSO agents with server, information is passes through http Requests and uses RSA algorithm to encrypt and store.

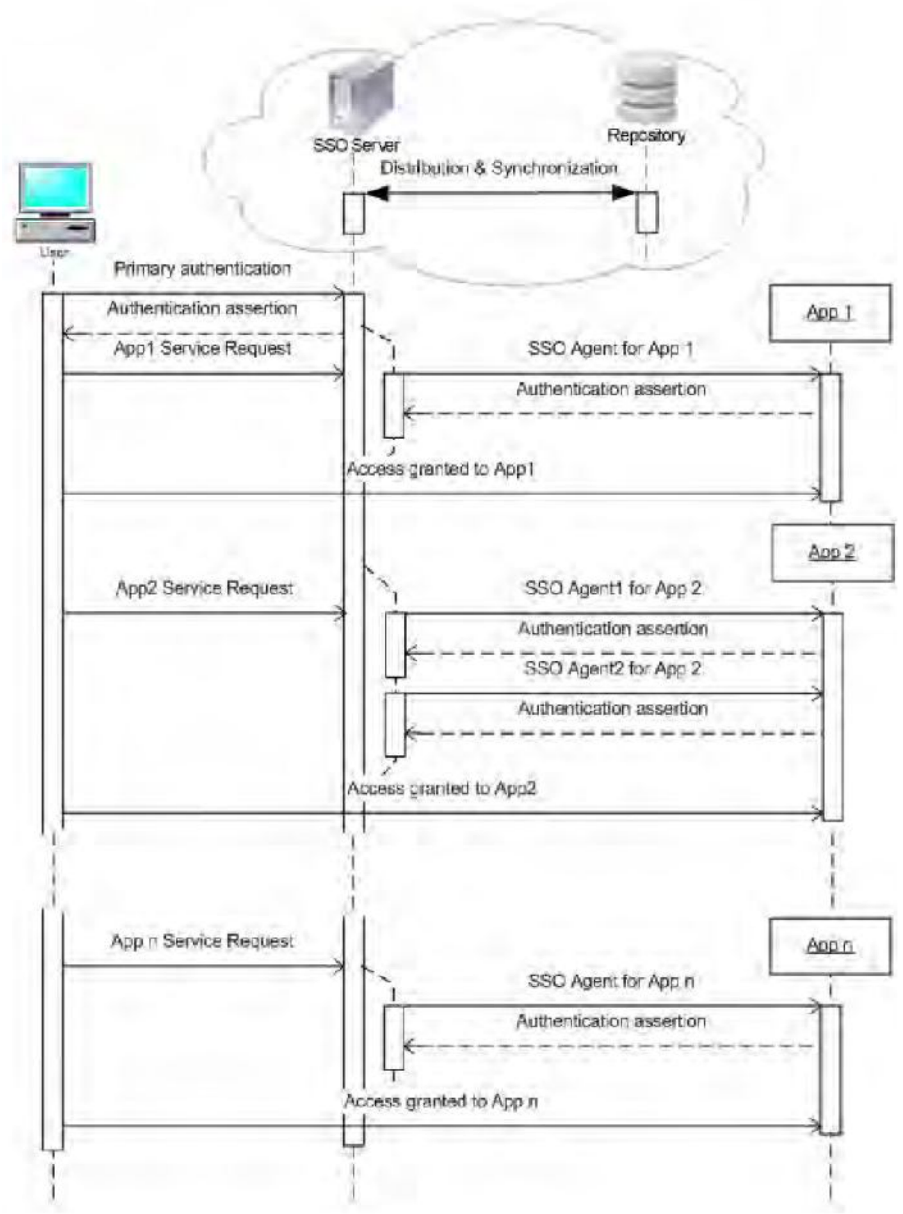


Figure 18 SSO for Cloud

2.1 Limitations of Literature Survey

In this all literature we came to know about how much SSO is beneficial and effective for any web development and why SAML is better choice than other available solutions of SSO. We also had checked REST and SOAP comparison and learned that REST is becoming popular due to is easy to deployment and development and many other factor it is most used solution in case of Rapid development [6]. With Rest although we can use XML very easily but XML also have its cost and have many extra features which are not required in many case for web services new technology which comes in use is JSON which is light weight and easy to use as well[9]. In our studies we also came to know how much Single Sign On is now important with distributive environment and also from client's ease of usability has checked how many SSO protocols are available for Web service and their status according to usability, popularity and also availability for research and devolvement.

[3]SAML is consider choice for our research which is SOAP base and also its last version which is V.2.0 came in 2006 after which many new technologies are invented and are in use of programming of web service and other web applications. This reference document shows Assertion and implementation with SOAP and XML technology and protocol.

[4] Shows how SAML can be use for Web and also mention how it can be protected from difference attacks like Replay attack man-in-middle attack but still it were SOAP base and with XML base assertion also it could be good solution for large and complex system where we expect divers type of response and request types in cases where we are not expect much diversity it could be costly.

[9] shows us SSO deployment with cryptography which is somehow now essential is data transfer and it become more important in case when data is send only or in header that is how REST works. every time when client access any service it needs to add token in its parameter list which later will be verified by Identity provider through service which is time taking and also add extra load on identity provider also it is not specified authentication process for REST.

[10] Has developed authentication process for REST with MD5 as encryption process although is quiet efficient than [9] as does not involve any key exchange mechanism but it is only for Service provider and client provider setup and did not given any guide line for its extension to SSO setup where authentication would be done only by central identity provider.

[11] is SSO solution with token like [9] but this token which is being generated by security service this scheme also have one more benefit which it can manage access control by implementing or adding one more parameter called as role. As shown is Figure 14 and Figure 15. We can see in Figure 14 client takes Token from security service and when send request to service it again validate this token from security service this is time consuming as sending request to security service and then waiting for its response also creates extra load don security service.

[14] is SSO solution for cloud but it also have same issue of extra load on SSO server. [13] is SSO solution with ECC as algorithm for cryptography.SSO server perform cryptography but this cryptography consist of client and servers URLs as public and private key which made him more secure from man in middle attack and phishing. [12] is solution of SSO with

JSON and also for E-commerce. Figure 16 is pictorial representation of system It shows both domain used public and private key of each other to send data to each other they are working as web service and identity provider to one and other this setup is quiet fine for B2B setup but is not applicable for B2C type of applications and also in case where more than one application are join together to create on large system in this we cannot make every as Identity provider. [2] is token generated process of authentication and authorization or SOA but it is hardware which so any machine needs special chip to word are authentication machine.

2.2 Conclusion from Literature

REST is getting popular in programming and also in place of XML JSON is getting in use [6] but also it is lacking in protocol which can support it especially in case of security. REST become more efficient with use of JSON with it so we can change the term like this “We need a security protocol which should be as effective as SOAP protocols are but efficient, fast and easy to use for which REST is getting its fame”. [9] [11] [14] [13] Had given SSO solution which are some effective but shows extra overload on Identity provider we need security model which can reduce overload and extra weight for client. [12] Is IdP solution but not for large scale and not for B2C environment. So we come to conclusion we need Single Sign On security protocol for REST, which should effective as SOAP base SAML. But easy to apply fast and efficient as REST, Can deals with all major security threats of web service mention in [1].

Chapter 3. Requirements Analysis

In this chapter we are going to high light what we extracted from our literature and how we are going to develop it with the help of available resource and technologies. Currently Web Service is becoming part of every development due to its flexibility and affordability. As rest is having structure like web so it inherited all the vulnerabilities of Web Application.

3.1 Open Issues

SOAP or REST in case of either Communication method we don’t have any Security Structure build with it. It is left to Programmers to Handle these thing by usage of some other security standard. Many Security Standard are introduced like WS-Security, XACML, SAML, WS-Trust and Many more but all these Security Standards are XML base so Work only with XML.REST which is emerging are new communication method for Web Services cannot use them without modification of their implementation Methods.

Common threats Categories are Spoofing, Information Disclosure, Tampering, Denial of Service and Privileges Alleviations these Categories include all type of Attack like SQL injection, Replay, offline password attacks in case of REST, Cross Site Scripting Attack with AJAX or any other way.

Following is Brief Introduction of few them [1]

- SQL Injection due to expose of Request/Response which is in form XML/JSON Attacker can easily find way to inject its query
- Replay Attack as REST is stateless so same request can be sent any time which can engage Service Provider for long time and genuine user will wait.
- Offline Password Attack attacker intercept Server’s Challenge and response value and guess Password as HTTP Authentication is Base64 encoded so highly vulnerable to offline password guessing attacks.
- Cross Site Scripting Attack by XML or AJAX can execute any JavaScript base code which is located are remote site which easily can be deployed any type of malware at machine

For both SOAP and REST, SSL helps to deals with Information Disclosure by Encryption. REST has only Basic HTTP Authentication for it which only base64 encoding and easily can be decrypted.

With this Security Threat still REST is becoming popular because it is easy to implement, less costly and also can support JSON which can also be used are request/response method in place of XML.

3.2 Project Scope

As define earlier REST doesn’t have any security standard associated with so we need a Security Standard which can encounter Replay Attack, Spoofing, Authorization issues and Tampering. SOAP does not have any issue of size of its packet but in case of REST as it HTTP base we have only two ways to send information first URI we can’t have large data in URI which can leads to Crashing of Service and exposing it internal information. Second are HTTP headers we can use any HTTP Header associated, theoretically it could also be unlimited but same reason of URL applies to header. So vendor in most of cases applies Maximum Length at Header and URL in which Application will work efficiently and does not catch in any type of dead lock. As after studying REST we know it have all vulnerabilities that Web can have plus due to exposure of resource in URL it makes application more under attacks. REST needs a security model which should be simple to implement as REST and JSON and also effective. Existing Security Standard works well with SOAP but they are not for REST because REST is Architecture Style not a Communication protocol which can have packet or tagged information inside it.

3.3 Problem Definition

After formulation of literature survey now we are going to define the problem which we are going to solve. Many security protocols have been introduced for REST and SOA in general as well but they all have some limitation in terms of cost, load balancing application. Single Sign On is now becoming essential need of any system that is created with many different web services which are located at different web hosting and developed in different web technologies. We need to check how we can make Single Sign On Setup for REST, SAML is considered to be optimal solution to be transform for REST as it is in practical use by many different famous enterprises but with XML an SOAP, now we need to see how we can make such setup with REST ad JSON as JSON will be best to apply in place of XML to make REST more fast.SSL which provider security at transport level we also need to check how can make our model security at message level.

With other benefits of SAML another one is it is loose Coupling of directions with platform independency so best choice as security standard with REST as REST is also Stateless. SAML protocol which is a security standard is being used in SOAP which is dealing SSO with SOAP already, REST web services are based on HTTP protocol we can use the HTTP Redirect Binding which is one of profile of SAML[15]to send the Unsolicited Responses. There is not any problem to add the necessary query parameters to any HTTP method, the

HTTP Redirect Binding with Unsolicited Responses covers the same scenario we have with SOAP without defining the additional standards[14]. SAML profile that will be used could be Web Browser SSO Profile as it can be initiated from Service Provider or Identity Provider as per request Case. As like WS-Security [4] SAML assertion is also not encrypted which we needs to encrypt so attacker cannot guess it and use it.URL length is also issue, theoretically it could be infinity but many vendors limit the url length for maintain efficiency and robustness. So there are chances of truncation of SAML response.

Load balancing is also important factor to be considering which is quiet ignored in [14] [13].so need system in which service provider can validated token by its self rather than again sending it for verification to identity provider and creates load on identity provider service.

Out focus is also on system which can work for client which are intended to interact with e-commerce application and our module would be able to support them it should not be only for B2B applications [12].

When we are developing our application we also need to consider that our web service would be hosted on ordinary hosting like shared hosting ion which all resource of web server are not accessible to them for some apprehension of hoisting providing companies although Identity provider is high computing server and can be deploy on some dedicated high level machine.

[14] http://saml.xml.org/news/how-to-use-saml-with-rest-web-services

Chapter 4. Proposed Solution and Methodology

This is third chapter of our research report in this chapter we will discuss how we can remove these shortcomings that we came to now from literature survey. After design of model which we have implemented this model with available technology.

Graphical representation of it will exact like define in [3] for HTTP Post Redirect Binding Only Difference will be in place of SOAP message information will be sent through HTTP header.

Steps for Authentication Process with SAML in REST

- Client Sends Request to Service Provider.
- Service Provider will redirect it to Identify Provider(IdP) with Authentication Request and SSO
- IdP replies Client (Browser or User) with challenge for Authentication.
- Client Sends response to IdP back by answering that Challenge.
- IdP Receive this response and check clients value and own generated value if match will reply with authentication access for particular Service Provider.

4.1 Proposed Assertion and Binding

These are steps followed for achieving SAML assertion in REST

Step1: Client or Browser Sent Request to Service Provider

Step2: Service Provider Redirect by HTP to Identity Provider with Assertion value it may or may not involve User interaction it can also be done Auto buy taking information from Client Machine.

Step3: IdP Given Challenge to Client which can be of any type depends on per system implementation and security needs e.g. it contain some random String or Data with Resource Identification and User Identification or it can also be some simple Login Authentication Setup to Verify User genuinely

Step4: Client gives response to Challenge On base which it will be decided to give access

Step6: User Redirect with Containing Access Flag which was given by IdP in its POST Request. Service Provider will confirm Assertion at its end for correct and valid assertion

Step7: After Validation of Assertion Service Provider will give access to Resources

Following Diagram Shows SAML Binding with REST in place of SOAP

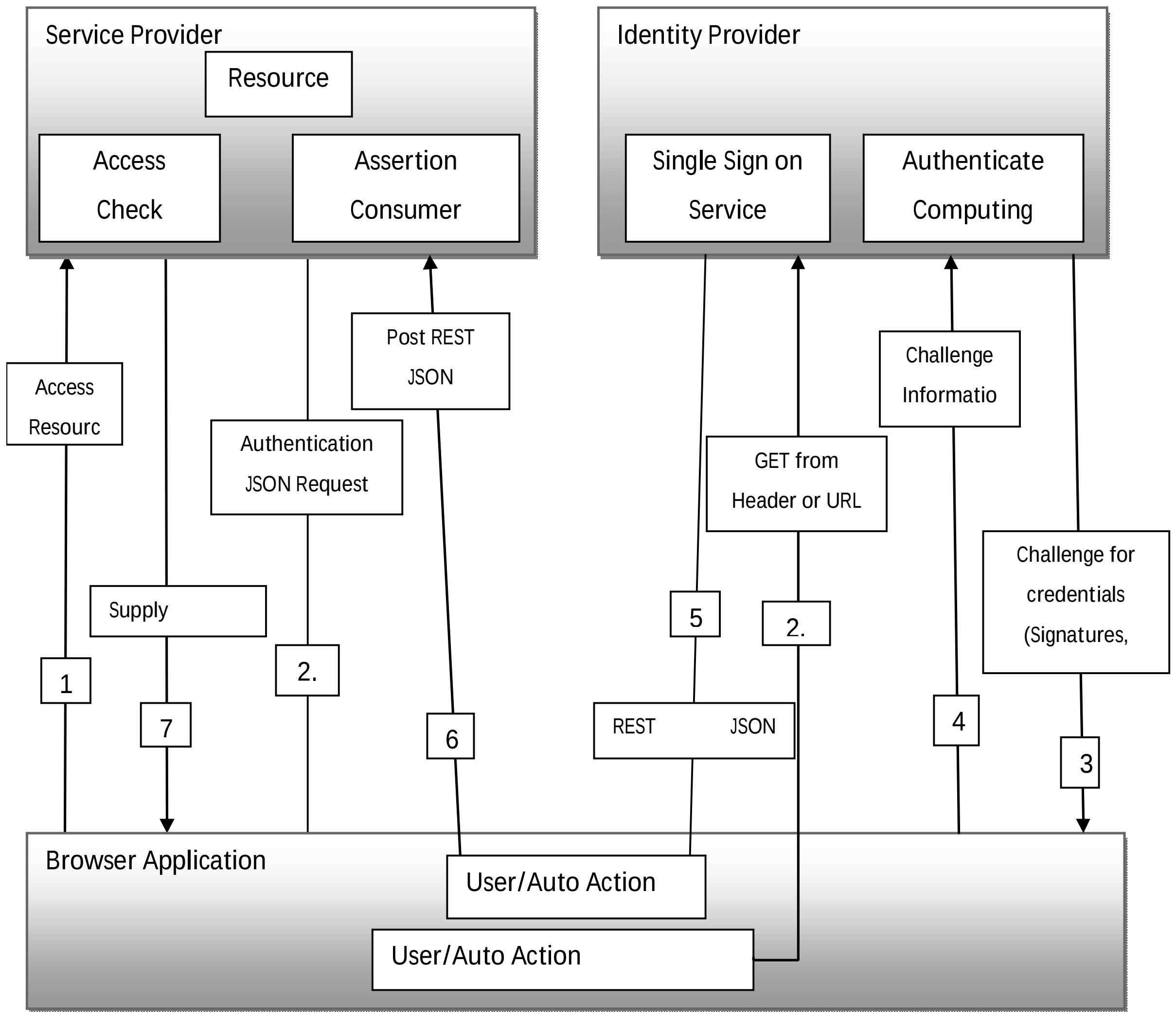


Figure 19 SAML HTTP Post and Redirect Binding with REST

4.2 Implementation

This Model is implemented in PHP. Two web services in REST style are created with on Identity Provider Server. One Web Application which will access both web service. Encryption is performed to secure data. IdP which is having all information about Web Services Subscribed with it. First of All IdP Created Private Public Key Pair which are distributed to Web Services which are Subscribed to it. How they are transferred is not dealt in this research and also it is out of question for it. When Web Application Access Web

Service it will check it have SAML response in its Parameters. If it have then it will decrypted with public key of IdP and check for Valid response from IdP. Decrypted String it JSON Array which can be decoded on. This JSON array have name of Web Application for which this Response was created and Code of Response and any Identification of user in our case it was Email Address. In case Web Service found not any parameter of SAML Response or Fail to decrypt Response Or Response Code is not success or Response created was not the Referring Application then Web Service not give access to any required resource. When it is found Request of Resource is not Authenticated then Web Service will generate a Request for Authentication which is encrypted with key of Particular web Service and this authentication request contain requesting application address required response format and instant value. Idp at receiving of Authentication request check decrypt with key of particular Web Service and check for referring Application and Requesting application value in JSON array. After successful decryption and necessary checks Idp show any challenge like login form after successful login Idp Return response for Web Application in Parameter name SAMLResponce this Response is encrypted with key of Idp. Consumer of this Response can use it for access to any other Web Service without need of re-login as Every Web Service Subscribed of same Idp can decrypt and use this SAMLResponce.

Abstract view of Model is show in following figure.

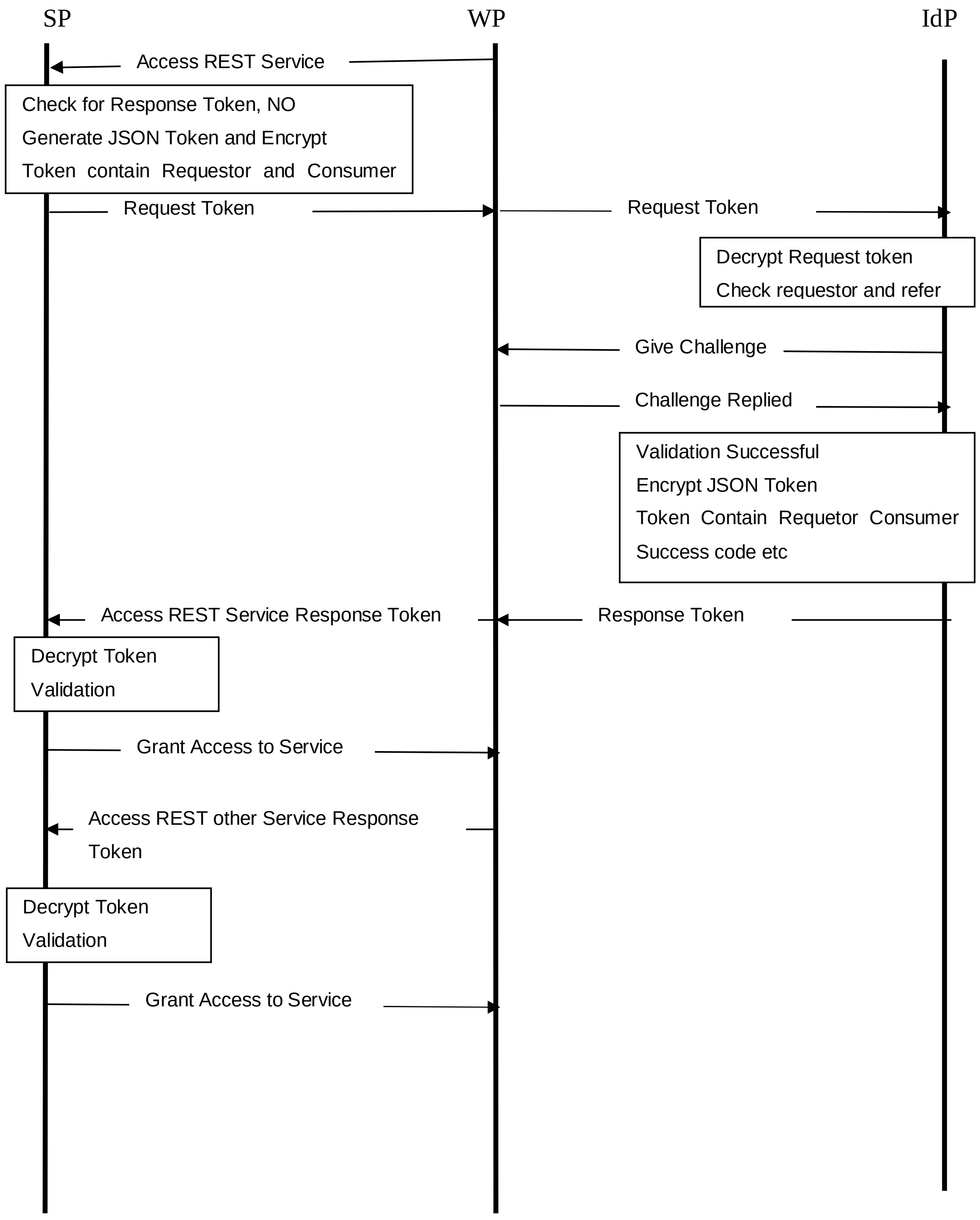


Figure 20 Flow diagram of Model

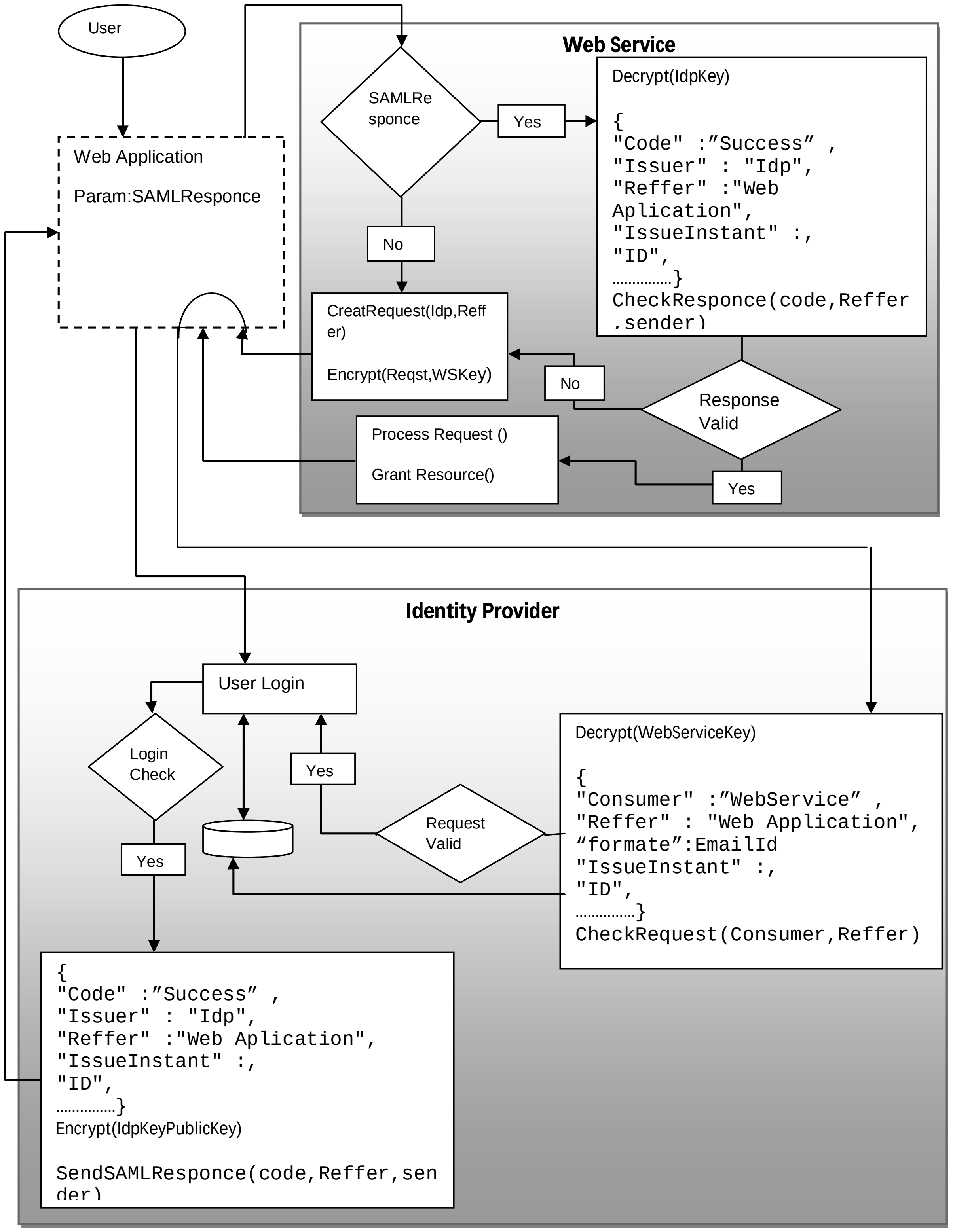


Figure 21 Proposed Implementation Model

4.3 Pseudo code of Implementation

Following is flow of Implemented Model.

This implementation is Service Provider Initiated.

- Keys are already exchange between Web Services.
- Web Application access desire Resource from Web Service.
- Web Service Perform following Checks.
- **Security Check at Web Service**

```
IF SAMLResponce
    IF Decrypt SAMLResponce with IdP Pubic Key
        JSON Response after Decryption
        IF Check JSON{'consumer': "WebApplication"} == HTTP Reffer
            Check JSON{'Expire':"TimeStamp"} //option if required to
        avoid replay attack
        Email= JSON{'EmailID':"EmailID value from Idp"}
        END IF
        ELSE
        Flag "Response is not generated for requesting Application or
        consumer"
        END ELSE
    END IF
    ELSE
    Flag "Response Value is tempered"
    END ELSE
Give access to desire Resiurce
END IF
ELSE
// Create Assertion Request for Idp
JSONRequest
{
'Requester':"Web Service"
'Consumer':"Web Application"
'ID':"uniqe number"
'IssueInstant':"time stamp when requested is generated"
'formate':"Email ID"// in which response is required
```

```
    }
    Encrypt  JSONRequest with WebServiceKey
    Unrlencode()
    Send back Request to Web Application or consumer with Resquest WebService name
    END ELSE
```

- **Check and Flow at Identity Provider**

```
IF SAMLRequest in POST vars
    IF Decrypt SAMLRequest with Key of WebService
        IF JSONRequest{‘consumer’:”Web Application”}== HTTPReffer
        Make User Authentication //Login,
        Successful Login
        JSONResponce
        {
        ‘ID’:”uniqe number”,
        ‘Issue Instant’:”time stamp when response is created”,
        ‘EmailID’:”Email value from Login”,
        ‘consumer’:”Web Application”,
        ‘Expire’:”TimeStamp”,//optional if need to avoid replay attack
        ‘issuer’:”IdP name”
        ‘Code’:”Success”
        }
        Encrypt SAMLResponce with IdpKey
        Urlencode SAMLResponce
        Return SAMResponce to Consuming Web Application
        END IF
        ELSE
        Flag “ Request is not created for using application”
        END ELSE
    END IF
    ELSE
    Flag “Data is tempered or Request is not from Mentioned Web Service”
    END ELSE
END IF
ELSE
Flag “cant not get authentication “
END                                                              ELSE
```

Algorithm 2 Pseudo Code

4.4 Tool of Development PHP

For the development of web services PHP is used with MySql as database. For encryption purpose PHP library phpseclib[15] is being used. In our development we also had used Ajax. CURL which is library of PHP is used for sending request as REST client from Web Application side.

PHP is one of most popular web application technology as free available and support every new development trend. Many famous web service and web application ar4e developed in PHP like Facebook.PHP also does not create any problem for its installation for development[16] or for live commercial purpose.PHP by default works with MySql as database server which is also freely available with MySql PHP support other database servers as well. PHP is platform independent and

4.5 Web Service

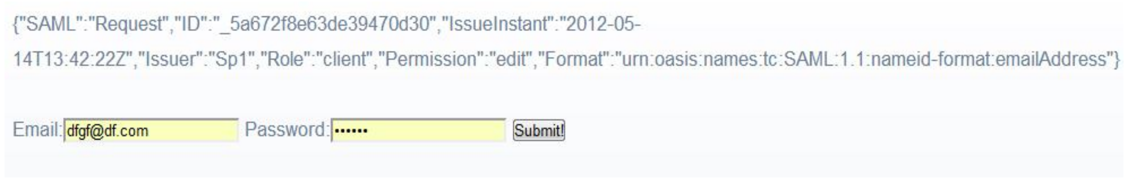
Two web services are created for testing of single sign on functionality. Identity provider generated private and public key for web services and every web service is facilitated with its own private key and pubic key of Identity provider.

Web services checks two conditions if Web Application sent ***‘SAMLResponce’*** then start its decryption with pubic key of identity provider and extract json decoded string. For verification of token, check for issuer, referrer, and expiration and generation time. Other condition can also be added in it as per requirement of system. After successful validation of token requested service or query result will be provided.

In case if SAMLResponce is not sent in request then it creates a request for authentication encrypted with its private key. This request contains information like requestor time

[15] http://phpseclib.sourceforge.net/
[16] http://www.easyphp.org/

format.Following figure shows request that is sent to identity provider from Web Service through Web Application which is working as Consumer of Web Services.



```
{"SAML":"Request","ID":"_5a672f8e63de39470d30","IssueInstant":"2012-05-14T13:42:22Z","Issuer":"Sp1","Role":"client","Permission":"edit","Format":"urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"}
```

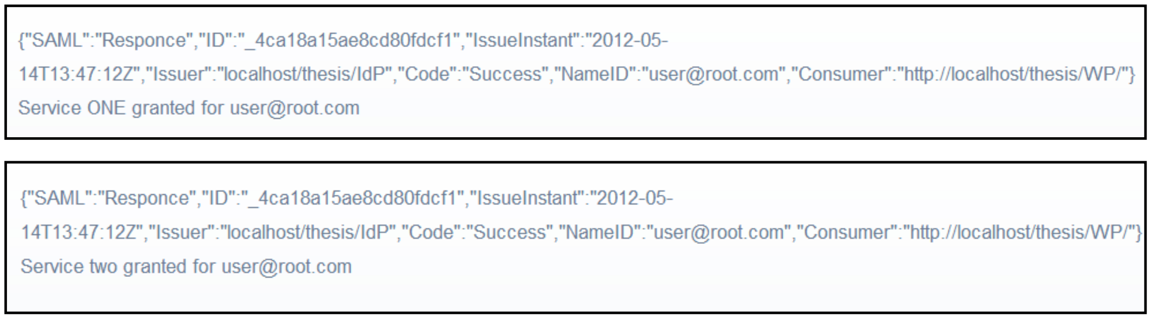
Email: dfgf@df.com Password: ••••••

Figure 22 SAML Request from Web Service for Identity Provider

4.6 Identity Provider

Identity provider when receives request. Of authentication it gives challenge to user for its validation authentication. In our case it is simple login form with user name and password. After successful validation Identity provider created SAML response which is encrypted with private key of identity provider and send it back to Service consumer which can further use it for access to services.

Following figures shows how response looks like.



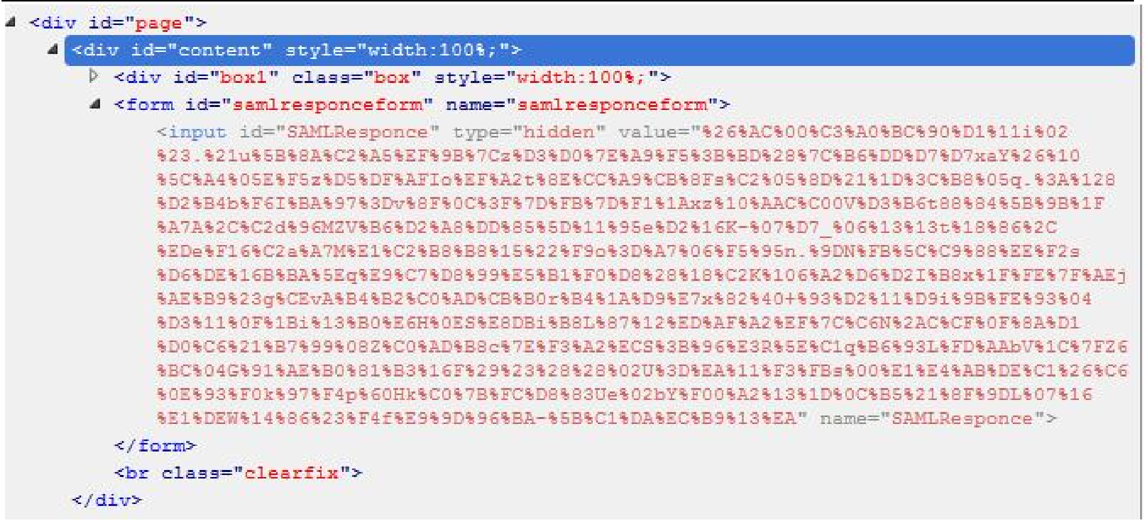
```
{"SAML":"Responce","ID":"_4ca18a15ae8cd80fdcf1","IssueInstant":"2012-05-14T13:47:12Z","Issuer":"localhost/thesis/IdP","Code":"Success","NameID":"user@root.com","Consumer":"http://localhost/thesis/WP/"}
Service ONE granted for user@root.com
```

```
{"SAML":"Responce","ID":"_4ca18a15ae8cd80fdcf1","IssueInstant":"2012-05-14T13:47:12Z","Issuer":"localhost/thesis/IdP","Code":"Success","NameID":"user@root.com","Consumer":"http://localhost/thesis/WP/"}
Service two granted for user@root.com
```

Figure 23 SAML Response generated by Identity provider

4.7 Requesting Web Application

Web Application is basically a consumer of services it could be any other application android application, iphone/ipad. Web application whenever access service it will send SAML response token to service to show it is valid access, else in other case it needs to get authenticated. As you can see in figure SAML response value is some string which is not meaningless apparently, so in case of token is stolen it is useless unless have keys of web services. For checking of token is consumed by application for which it is issues Consumer name variables is also added to it which make sure same application is using this token.



```
<div id="page">
  <div id="content" style="width:100%;">
    <div id="box1" class="box" style="width:100%;">
    <form id="samlresponceform" name="samlresponceform">
        <input id="SAMLResponce" type="hidden" value="%26%AC%00%C3%A0%BC%90%D1%11i%02
        %23.%21u%5B%8A%C2%A5%EF%9B%7Cz%D3%D0%7E%A9%F5%3B%BD%28%7C%B6%DD%D7%D7xaY%26%10
        %5C%A4%05E%F5z%D5%DF%AFIo%EF%A2t%8E%CC%A9%CB%8Fs%C2%05%8D%21%1D%3C%B8%05q.%3A%128
        %D2%B4b%F6I%BA%97%3Dv%8F%0C%3F%7D%FB%7D%F1%1Axz%10%AAC%C00V%D3%B6t88%84%5B%9B%1F
        %A7A%2C%C2d%96MZV%B6%D2%A8%DD%85%5D%11%95e%D2%16K-%07%D7_%06%13%13t%18%86%2C
        %EDe%F16%C2a%A7M%E1%C2%B8%B8%15%22%F9o%3D%A7%06%F5%95n.%9DN%FB%5C%C9%88%EE%F2s
        %D6%DE%16B%BA%5Eq%E9%C7%D8%99%E5%B1%F0%D8%28%18%C2K%106%A2%D6%D2I%B8x%1F%FE%7F%AEj
        %AE%B9%23g%CEvA%B4%B2%C0%AD%CB%B0r%B4%1A%D9%E7x%82%40+%93%D2%11%D9i%9B%FE%93%04
        %D3%11%0F%1Bi%13%B0%E6H%0ES%E8DBi%B8L%87%12%ED%AF%A2%EF%7C%C6N%2AC%CF%0F%8A%D1
        %D0%C6%21%B7%99%08Z%C0%AD%B8c%7E%F3%A2%ECS%3B%96%E3R%5E%C1q%B6%93L%FD%AAbV%1C%7FZ6
        %BC%04G%91%AE%B0%81%B3%16F%29%23%28%28%02U%3D%EA%11%F3%FBs%00%E1%E4%AB%DE%C1%26%C6
        %0E%93%F0k%97%F4p%60Hk%C0%7B%FC%D8%83Ue%02bY%F00%A2%13%1D%0C%B5%21%8F%9DL%07%16
        %E1%DEW%14%86%23%F4f%E9%9D%96%BA-%5B%C1%DA%EC%B9%13%EA" name="SAMLResponce">
    </form>
    <br class="clearfix">
  </div>
```

Figure 24 SAML Token for Service Consumer

Screenshot of full flow of this model application with coding high lights is added in Appendix.

Chapter 5. Results

In this chapter we are going to discuss the result that we had extracted from our implementation and bases of these results we will evaluate our result success. After implementation we had noted down results. For testing parameter and tools Authors of [15] has given few guide lines as well as tools and their comparison which helped us in selecting tools and parameter for our work. Testing web service is quiet difficult moreover when you need to check more than one services collectively.

Our testing consist of three parts

- Defining cases of vulnerabilities.
- Comparison with literature
- Load test

5.1 Testing Vulnerabilities to different attacks

Web application and web services are vulnerable to much attack but here we are high lighting only those, which could occur or seems can occur due to this model.

5.1.1 Token Stealing

In case if someone tries to use token that is generated for WP. Honest Web Service that is SP1 and SP2 after decryption check for the token referrer this information comes in header with all other information should match to the issuer parameter of JSON array that is extracted after decryption. This would evenly also solve the problem that is mention is [16] a case in which some web service become dishonest and work as intruder. As in our case Identity provider will issue token only for referrer and service provider will check token for which it is issued is also refereeing it.

5.1.2 Session hijacking

In this whole model flow we had showed we did not used any session as Web Services are REST base so already stateless and token will be kept in WP until it is open as soon as it get close “SAMLResponce” value will also get discarded although it consumer and developer of WP thinks that is not harmful for them to keep this value then cane make its cookie at their end, but still for SP1 and SP2 it is still stateless.

5.2 Replay attack

To make sure token it not used again and again to Identity Provider can add parameter of "Not After" which will define time limit after which it will be consider expire. Also SP1 or Service Provider would be able to compile it.

5.3 Cross Site Scripting (XSS)

Although cross site scripting attack is not part of this model as it is problem that is be deal by programmer in coding and it is programmer style which implements such filter those can disable effectiveness of cross site script or unwanted scripts.

In out model implementation there are two post variables which are vulnerable to XSS one is SAMLResponce and other is SAMLRequest so mainly any implementation in any technology need to secure these two variables from being exploited in Web Application (WP)

5.4 Comparison with Survey discussed Techniques

Now we are going to compare our proposed technique with the existing technique that is mention in literature section. One Main thing that we added more is Access control facility in out model which is not basically part of Single Sign On but it is also need of new Development in SOA it will make things easy rather than using separate protocols for authentication and authorization one can work for both situation with adding extra load.

First of all we are going to compare it with [12] i.e. "JSON Based Decentralized SSO Security Architecture in E-Commerce"

JSON Base Decentralize SSO	**Our Proposed**
For B2B	B2C
More than one IdP	One IdP
Works for limited number of Services as everyone will be IdP it would be like Mesh sort of topology.	Works for large number of Service as it is like star topology every Service is connected to only one IdP.
Hard to expand for smart device implementation due to decentralize Identity providing system	Can easily be adopted for smart devices.

Table 6 Comparison of JSON Decentralized SSO [12] with Our Technique

Now we are going to compare our Proposed model with [11] “Simplified Authentication and Authorization for RESTful Services in Trusted Environments”

Simplified Authentication and Authorization for RESTfull Services in Truest Environments	**Our Propsed**
SSL is required to encounter eavesdropping	Encryption is performed in model without involving any extra library from server side.
Service Provider validate token from IdP .	Service Provider can validate token by itself.

Next Comparison is with [14] “Securing user authentication using single sign-on in Cloud Computing”

Securing using SSO in cloud computing	**Our Proposed**
Load on Security Server for validation of token	No load on IdP as Service Provider will validate token of Responce.

Table 7 Comparison between Securing using SSO in cloud computing [14] with our proposed model

This comparison is with [13] “Secure Web Based Single Sign-On (SSO) framework using Identity Based Encryption System”.

Secure SSO framework Identity Based Encryption	**Our Proposed**
ECC cryptography which is less CPU time consuming but complex mathematics [17]	RSA which is high CPU time consuming but established
Profile Creation cost in every session	No need of profile creation once service provider are subscribed with Identity provider they can work in authentication.
User interact to application through SSO server which add load to SSO server can creat bottle neck	User interfact to service provider directly after getting authentication from Identity Provider.

Table 8Comparsiion Secure SSO framework Identity Based Encryption [13] with our proposed model

5.5 Quantitative Analysis

Quantitative analysis is not performed on this model there are many reason for it. We can see in literature [16] [5] [4] [12] [17] [14] [13] and others model related to SOA or web service does not mention any sort of quantitative analysis.” Secure Web Based Single Sign-On (SSO) framework using Identity Based Encryption System” [13] had performed some tests related to load and performance testing using “HTTPAttack” name tool which is not available right

[17]http://crypto.stackexchange.com/questions/1190/why-is-elliptic-curve-cryptography-not-widely-used-compared-to-rsa

now[18]. Beside this fact there are some reason due to which it is difficult to perform such test in such things which are related to some model of web service as every implementation technology have its own performance constraint JAVA is slower than PHP. [7]
Testing web services is more challenging than testing traditional software due to the complexity of Web service technologies and the limitations that are imposed by the SOA environment. Limited controllability and observation render most existing software testing approaches either inapplicable or inactive. Some of the technological benefits of SOA such as late-binding and dynamic service selection also increase the level of testing challenge [18].

5.6 Load Testing

Load and stress handling depends on many factors like technology of development, server machine power, database server, web server technology so result that are showing in our research may vary according to any changes in these factors. Like as we had mention we had used PHP if instead of PHP, JSP is being used result would be same for model by it would be slower so number of transaction per second might be less then shown. Similarly we had test our work on PC machine localhost but for live server which are commercial available it will show different result as they are able to handle high load and can handle high number of transaction per second. Figure below shows load test that is performed in SOAPUI[19] trial version is used to perform this test. This figure is Excel representation of data to make it elaborative. We can see as the number of threads increases number of transaction per second decreases. Bytes per seconds are almost consistent in recoded time. Next Figure is SOAPUI graph screenshot.

[18]https://www.google.com.pk/#hl=en&sclient=psy-ab&q=%22HTTPAttack%22&oq=%22HTTPAttack%22&aq=f&aqi=g-s4&aql=1&gs_l=hp.3..0i10l4.9697696.9701341.3.9701560.12.10.0.0.0.0.1056.2754.6-2j1.3.0.cish.1.0.0.INlsLJrYc0A&pbx=1&bav=on.2,or.r_gc.r_pw.r_cp.r_qf.,cf.osb&fp=2a099ec05bf823bb&biw=1190&bih=636
[19] http://www.soapui.org/

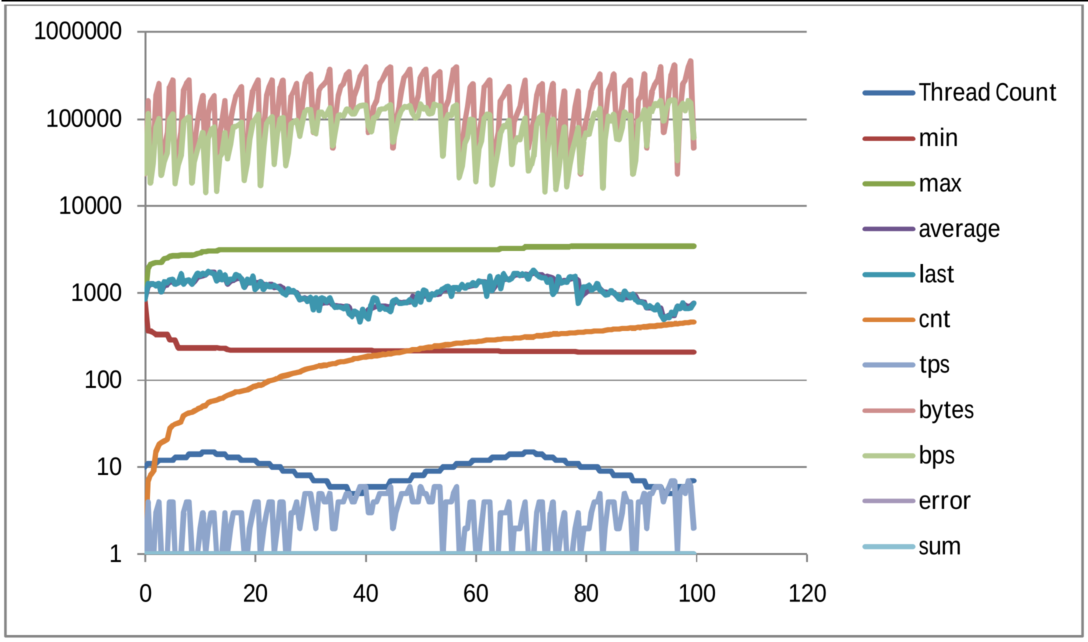
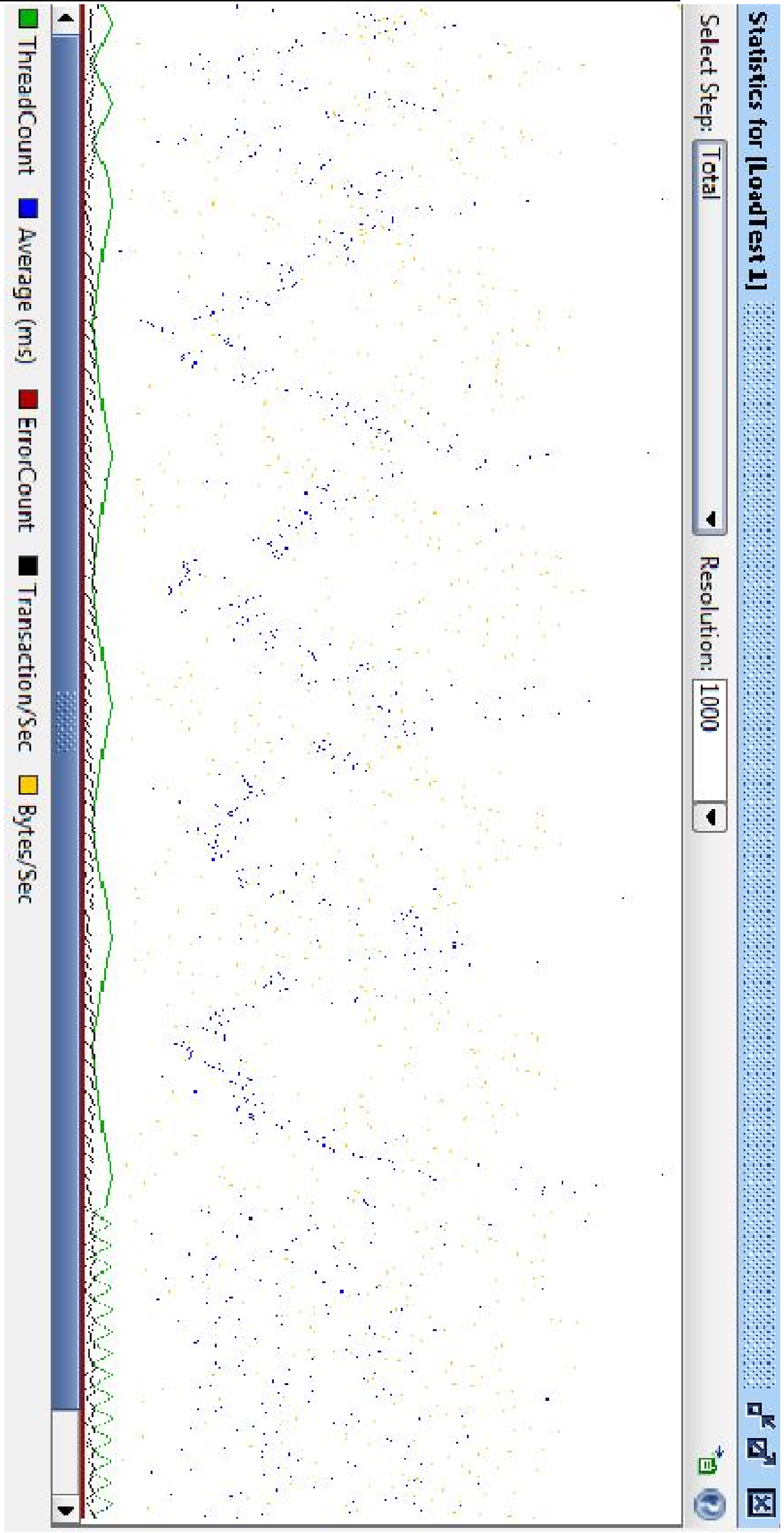


Figure 25 Load Test graph with Excel


Statistics for [LoadTest 1]

Select Step: Total

Resolution: 1000

Figure 26 Load Test in SOAPUI

Chapter 6. Conclusion & Future Work

This is last chapter of this documentation in this chapter we are going to summarize our whole research work and its result and effect on future researcher in this filed.

6.1 Conclusion

REST with its popularity needs all security protocol or standard which are previously only for SOAP should be transformed according to needs of REST. Also federated Identity management which is also most used Authentication Management system Single Sign On which is also one of property of Federated Identity Management. SAML which is loose coupled and free source Standard of security and Federated Identify with SSO. This Research focused on Web Services which are REST base so are stateless and can't have session management system at their end. This Research had given security setup for REST Web Service this setup is base on Federated Indentify Setup and provide Single Sign On facility. Encryption is also applied to handle any type of security issue related to phishing and other attacks related to stealing. During this research Web Services and Identity Provider Server setup both are REST base so it had given a way to have Identity Provider in REST style. For Load balancing on Identity Provider Server Response which is generated have ID of IdP also all Web Services which are subscribed with same IdP are provided with public key of IdP so if they will be able to extract and decrypt JSON data from it give it hind data or Response Parameter is not tempered and coming from same Consumer for which IdP had issued Response. Web Application can maintain sessions at its end for further refer use or to access other Web Services from same IdP. Where are if want to avoid *ReplayAttack* in JSON array IdP can add two more parameter these can be **NotAfter** and **UseBefore.** This work had showed with common web base technologies and protocols secure Web Services can be built.

6.2 Future Work

Web Service in becoming basic component of many development structure now many user needs Android Iphone Ipad apps for their application it is obvious if single sing on is available for Web Application it will be required in Android Iphone app as well. Our next step will how we can make it compatible with these development technologies. These developments also used Web Service for their bases so Service Provider and Identity provider does not need any sort of changes in it. Change related to posting from this application and posting to these applications needs to get study.

This model also has some vulnerability to DoS in case WP get dishonest and try to send again and again SAMLResponce fake value to make Service Provider busy this issue needs to be dealt. Also now with emerging technology and computing structure new concept of mobile hoisting is introduced, we also can make it available for mobile hosting setup.

In future work this same technique can be extended with IdP discovery in which Web Services is not provided with authentication server or can be authenticated from more than one server so the selection of server with less load and efficiency. As this technology base on encryption so in IdP discovery it will also be dealt how keys can be exchanged or how encryption will be performed.

References

[1] Noureddine, Adam A., "Security in web 2.0 application development," in *08th International Conference on Information Integration and Web-based Applications & Services(iiWAS2008)*, New York, 2008.

[2] Mohammad Ghafari, Mortaza Saleh, Dr. Nasser Modiri, "A Model to Increase the Trust in Service Oriented Architecture," in *2011 Fifth Asia Modelling Symposium*, 2011.

[3] OASIS, "Security Assertion Markup Language (SAML) V 2.0 Technical Overview," 25 March 2008. [Online]. Available: http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf. [Accessed 21 April 2012].

[4] Kelly D. LEWIS, James E. LEWIS, "Web Single Sign-On Authentication using SAML," *IJCSI International Journal of Computer Science Issues,* vol. 2, pp. 41-48, 2009.

[5] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, Giancarlo Pellegrino, Alessandro Sorniotti, "From Multiple Credentials to Browser-based Single Sign-On: Are We More Secure?," in *26th IFIP TC-11 International Information Security Conference (SEC 2011)*, Switzerland, 2011.

[6] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann, "RESTful Web Services vs. “Big” Web Services: Making the Right Architectural Decision," in *17th International Worl Wide Web Conference(WWW2008)*, Bejing,China, April 2008.

[7] J. Nahon, "www.comp.leeds.ac.u," August 2011. [Online]. Available: www.comp.leeds.ac.uk/mscproj/reports/1011/nahon.pdf.gz. [Accessed 15 May 2012].

[8] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, Clemente Izurieta, "Comparison of JSON and XML Data Interchange Formats: A Case Study," in *Computers and Their Applications in Industry and Engineering*, Montana State University, USA, 2009.

[9] Lata Kiran, Sandeep Sood, Dr,Khuldip Singh, "A Signle Sing-On Model for Web Services based on Password Scheme," in *CICSYN '09 Proceedings of the 2009 First International Conference on Computational Intelligence, Communication Systems and Networks*, Washington DC, 2009.

[10] Dunlu Peng, Chen Li, Huan Huo, "An extended UsernameToken-based approach for REST-style Web Service Security Authentication," in *2nd IEEE International Conference on Computer Science and Information Technology*, 2009.

[11] Eric Brachmann, Gero Dittmann and Klaus-Dieter Schubert, "Simplified Authentication and Authorization for RESTful Services in Trusted Environments," IBM, 2011.

[12] Ye Jun, Li Zhishu, Ma Yanyan, "JSON Based Decentralized SSO Security Architecture in E-Commerce," in *In Fei Yu, Qi Luo, Yongjun Chen, Zhigang Chen Proceedings of The International Symposium on Electronic Commerce and Security, ISECS 2008*, Guangzhou, China, 2008.

[13] Rajesh Kumar Singh, Alwyn R Pais, "Secure Web Based Single Sign-On (SSO) framework using Identity Based Encryption System," in *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, 2009.

[14] Revar, A.G,Bhavsar, M.D, "Securing user authentication using single sign-on in Cloud Computing," in *Engineering (NUiCONE), 2011 Nirma University International Conference on*, Ahmedabad, Gujarat, December 2011.

[15] Sana Azzam, Mohammed Naji Al-Kabi, Izzat Alsmadi, "Web Services Testing Challenges and Approches," in *International Conference on Computing and InformationTechnology - Information Systems*, Taibha, KSA, 2012.

[16] Armando, A., Carbone, R., Compagna, L., Cu_ellar, J., Tobarra, L., "Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps.," in *6th ACM workshop on Formal Methods in Security Engineering*, 2008.

[17] Zhang Wenchao, Li Yafen(Senior- Engineer), "Federation Access Control Model Based on Web-Services," in *2010 International Conference on E-Business and E-Government*, 2010.

[18] Mustafa Bozkurt, Mark Harman and Youssef Hassoun, *Testing & Verification In Service-Oriented Architecture: A Survey*, London: online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/000, 2009.

[19] Sebastian Graf, Vyacheslav Zholudev,Lukas Lewandowski, "Hecate,Managing Authorization with RESTful XML," in *second international workshop on RESTful Design*, New York, 2011.

[20] Nawwar Kabbani, Scott Tilley, Lewis Pearson, "Towards an Evaluation Framework for SOA Security Testing Tools," in *IEEE International Systems Conference*, San Diago, April 2010.

Appendix

A. Nomenclature

DoS	Denial of service
IdP	Identity provider
JSON	JavaScript Object Notation
MD5	Message digest 5
PHP	Hypertext Preprocessor
REST	Representational State Transfer
SA	Security Association
SAML	Security Assertion Markup Language. XML-based emerging Oasis standard for exchange of “assertions” enabling distributed authorization services.
SHA	Secure hash algorithm
SOA	Service Orient Architecture
SOAP	Simple Object Access Protocol
WADL	Web Application Description Language
WS	Web Service
WSDL	Web Service Descriptive Language
UDDI	Universal Description and Discover Integration
XSS	Cross Site Scripting