Implementation Strategies for Multifactor Authentication for E-Governance Applications through Restful Webservices

V. Nirmalrani
Information Technology
Sathyabama University
Chennai, Tamil Nadu

P. Sakthivel, Ph.D
Electronics and Communication Engineering
Anna University
Chennai, Tamil Nadu

ABSTRACT
Governance means the exercise of political, economic and administrative authority in the management of a country's affairs, including citizen's interests and exercise of their legal rights and obligations. E-governance may be understood as the performance of this governance through the electronic medium in order to facilitate an efficient, speedy and transparent process of disseminating the required information to the public, and other agencies to perform the government administration activities. Authentication is the key to secure e-Governance applications and services. User name and password credentials are used for authenticating and authorizing, which is not sufficient. As Internet is more vulnerable nowadays, this one factor authentication is not secure and it is vulnerable for hacking. Even, in case of RESTful Web services, the current system doesn't provide any security measures except user name and password credentials, even which are hard coded in the invoking applications. This paper proposed a novel and sufficient solution that addresses the authentication in more secure and complex way. The proposed work uses the multi-factor authentication for e-Governance Applications through RESTful web services. Multi-factor Authentication includes One Time Password (OTP), Digital Signatures, extended Token based authentication for web services. Solutions to be delivered as Web services (Component based architecture) with certain access control which serves the following two purposes. First, it secures the application and services, and latter it provides a reusable component for authentication.

General Terms
Security, Access Control.

Keywords
Authentication, E-Governance, Access Control, Authorization, Security.

1. INTRODUCTION
Security is a critical part of any web application. Web applications by definition allow users to access a central resource - the Web server - and through it, to others such as files, database servers, etc., by understanding and implementing proper security measures, the resources are guarded against unauthorized access and also provide a secure environment in which users are comfortable working with application. Web Application Security is a science of information security relating to the World Wide Web, HTTP and web application software. It is also known as "Web Security".

2. RESEARCH BACKGROUND

2.1 E-Governance
Governance refers to the exercise of political, economic and administrative authority in the management of a country's affairs, including citizens' articulation of their interests and exercise of their legal rights and obligations. E-governance may be understood as the performance of this governance via the electronic medium in order to facilitate an efficient, speedy and transparent process of disseminating information to the public, and other agencies, and for performing government administration activities. E-governance can bring forth new concepts of citizenship, both in terms of citizen needs and responsibilities. Its objective is to engage, enable and empower the citizen.

2.2 Webservices
A Web service is a method of communication between two electronic devices over the web (Internet). The W3C defines a "Web service" as "a software system designed to support inters operable machine-to-machine interaction over a network". It has an interface described in a machine-process able format (specifically Web Services Description Language, known by the acronym WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

2.3 Styles of Webservices
Web services are a set of tools that can be used in a number of ways. The three most common styles of use are RPC, SOA and REST.

2.3.1 Remote Procedure Call (RPC)
RPC Web services present a distributed function (or method) call interface that is familiar to many developers. Typically, the basic unit of RPC Web services is the WSDL operation.

2.3.2 Service Oriented Architecture (SOA)
Web services can also be used to implement architecture according to service-oriented architecture (SOA) concepts, where the basic unit of communication is a message, rather than an operation. This is often referred to as "message-oriented" services. SOA Web services are supported by most major software vendors and industry analysts. Unlike RPC

Web services, loose coupling is more likely, because the focus is on the "contract" that WSDL provides, rather than the underlying implementation details.

2.3.3 REpresentational State Transfer (REST)
REST attempts to describe architectures that use HTTP or similar protocols by constraining the interface to a set of well-known, standard operations (like GET, POST, PUT, DELETE for HTTP). Here, the focus is on interacting with stateless resources, rather than messages or operations. Clean URLs are tightly associated with the REST concept. An architecture based on REST (one that is 'RESTful') can use WSDL to describe SOAP messaging over HTTP, can be implemented as an abstraction purely on top of SOAP (e.g., WS – Transfer), or can be created without using SOAP at all.

2.4 Authentication
Authentication is a process of verifying the identity or location of a user, a service or an application. Authentication is performed using at least one of three mechanisms: “something you have”, “something you know” or “something you are”. The authenticating application may provide different services based on the location, access method, time of day, etc. It confirms that users are who they say they are [1]. For example, a user must provide a user name and password that are checked against an authority (a database, for example, or a domain server). Nearly all websites that maintain user-specific accounts employ passwords to verify that a user attempting to access an account is, in fact, the account holder.

2.5 Authorization
The determination of what resources a user, service or application has permission to access. Accessible resources can be URLs, files, directories, servlet, databases, execution etc. In other words, it is the process of granting or denying access to resources for specific users. In any sort of network, the user logs in, after doing that successfully, the user can access specific files, folders, printers, and other resources - that is, the user have authorization to get access to resources based on login credentials.

2.6 Need for E-Governance
The purpose of implementing e-governance is to enhance good governance. Good governance is generally characterized by participation, transparency and accountability. The recent advances in communication technologies and the Internet provide opportunities to transform the relationship between governments and citizens in a new way, thus contributing to the achievement of good governance goals. The use of information technology can increase the broad involvement of citizens in the process of governance at all levels by providing the possibility of on-line discussion groups and by enhancing the rapid development and effectiveness of pressure groups. Advantages for the government involve that the government may provide better service in terms of time, making governance more efficient and more effective. In addition, the transaction costs can be lowered and government services become more accessible.

3. MOTIVATION
The aim of this paper is to formulate the strategies for securing e-governance applications and its corresponding web services [4]. As a part of security, this paper examines the following:

Multi-factor Authentication for e-governance applications

- One Time Password (OTP)
- Mobile OTP
- HMAC OTP
- Digital Signatures
- Implementing Security as reusable components (Web Services)
- Implementing WS-Security Model for RESTful Web services
- Implementing Qualifier based Access to RESTful Web services

3.1 Existing System
The existing system is using very simple authentication methods. Most of the applications are using text based user name, password combination to authenticate users into the system. Applications have implemented application level roles to differentiate users.

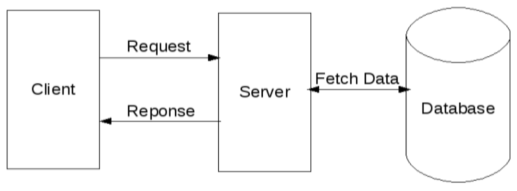


Fig 1: Existing System

This authentication mechanism is used in all e-governance applications and each having different way of implementation. Web services are used to communicate between e-governance applications. Authentication & Authorization are only at web server level. Most of the e-governance services are using Digest Authentication and roles for authorization.

3.2 Issues in Existing System

3.2.1 Weak Authentication
The existing system is using weak authentication mechanism. It simply prompts for a user name and password credentials from the end user. These credentials are transferred over HTTP to the server. Some of the e-governance applications encode the credentials using certain hashing methods. Some transfer the data over SSL. This mechanism fails when the password is leaked. Someone who came to know the password either by sniffing or by some other mechanisms, it is very easy for him to tamper the system. Also, in this mechanism, no one can claim that an action was done actually by the same user.

Web services are using simple Digest authentication which is again simple text based passwords. This also can be passed to some other persons and the services can be accessed. When accessing from client e-governance applications the user name credentials has to be hard coded in the code and it will be transferred as plain text.

3.2.2 Insufficient Privileges
In the existing system, the web services implements security by simply locking IP segments in the network/firewall level. No specific mechanisms were implemented in application level. This may create a problem when the more applications are hosted in the allowed IP address and no mechanism to find which application is invoking the service URL. Since the user name credentials were passed over HTTP as clear text, sniffing is also possible.

3.2.3 Too many mechanisms
Security implementation codes are mostly developed for each e-governance application. This makes variety of implementation for the same purpose with different pros and cons. Logics are not shared among the applications. So every application's authentication and authorization behaves differently. They differ by having different strategies for developmental activities and implementation.

The following or the other issues in existing system

- Only Single factor Authentication
- Access level is very low

3.3 Proposed System
Proposed System seems to implement the following mechanisms to enhance the existing system:

- Multi-factor Authentication
- Qualifier based Authorization
- WS-Security Based Authentication for RESTful Web Services
- Security as reusable Components

3.3.1 Multi-factor Authentication
Multi-factor authentication requires the use of elements from two or more categories. Supplying a user name ("something the user knows") and password (more of "something the user knows") is still single factor authentication, despite the use of multiple pieces of distinct information. An example of true multi-factor authentication is requiring that the user also utilize a hardware token or virtual token, a smart card or USB dongle, ("something the user has"), or a thumb print or iris scanner ("something the user is").

At the same time they are validating the identity of the user, many on-line sites also attempt to confirm the validity of the site to the user (called "mutual authentication"). The weakest form of mutual authentication generally displays an image and/or phrase previously selected by the user. More advanced forms of mutual authentication exchange a one-time key with the user's device.

3.3.2 Qualifier Based Authorization
It is suggested to use Server level Roles for authorization as level 1 and one more authorization should be made at Application level. This enables the application level authorization and solves the problem of same IP different application access. Each client should be registered against the access to web services [5].

3.3.3 WS-Security for REST services
REST does not have predefined security methods so developers define their own, and often, developers in a hurry to just get their web services deployed don't treat them with the same level of diligence as they treat web applications.

These conditions lead to web services with serious vulnerabilities [3]. For instance, most APIs handle authentication using a key but no secret, essentially requiring a user name but no password. Another problem is using HTTP basic authentication (with no SSL) and letting the user name and password cross the wire with no encryption.

REST APIs typically have the same attack vectors as standard web applications, including injection attacks, cross-site scripting (XSS), broken authentication and cross-site request forgery (CSRF).

So, the proposed system tries to implement a WS-SECURITY based Authentication for REST. WS-SECURITY is a security standard used in SOAP services.

4. AUTHENTICATION AND AUTHORIZATION
4.1 Multi-Factor Authentication

4.1.1 MOTP
Using static passwords for authentication, as it is commonly done, has quite a few security drawbacks: passwords can be guessed, forgotten, written down and stolen, eavesdropped or deliberately being told to other people. A better, more secure way of authentication is so called "two-factor" or "strong authentication" based on one time passwords. Instead of authenticating with a simple password, each user carries a device ("token") to generate passwords that are valid only one time. To generate a onetime password, the user has to enter his personal PIN into the device. So the authentication is based on two factors: the token device and a PIN. This is obviously more secure than just a password, as an attacker needs to get hold of both the PIN as well as the token device. In addition, eavesdropping on a password that is valid only one time is of no use to the attacker.

On the other hand, the drawback of strong authentication is that every user has to be provided with a token device. This can be quite expensive. Fortunately mobile phones that are capable of running Java applets are becoming more and more widely spread. It stands to reason to use your mobile phone as an authentication token.

Mobile-OTP is a free "strong authentication" solution for Java capable mobile devices like phones or PDAs. The solution is based on time synchronous one time passwords [2]. It consists of a client component (a J2ME MIDlet) and a server component (a UNIX shell script).

4.1.1.1 Algorithm
The MIDlet generates OTPs by hashing the following data with MD5:

- The current epoch-time in a 10 second granularity
- The 4-digit PIN that a user enters
- A 16-hex-digit secret that has been created when the device was initialized.
- When entering a PIN, the MIDlet displays the first 6 digits of the MD5-hash.

4.1.1.2 Applications
MOTP can be used to authenticate a user in a system via an authentication server. Also, if some more steps are carried out (the server calculated and user calculated values are checked against each other), the user can also authenticate the validation server.

4.1.1.3 MOTP API Work flow Architecture
This is the one time password. The password can be verified by the server, as the server also knows the current time, Init-Secret and PIN of the user. To compensate time differences, the server will accept passwords from 3 minutes in the past to 3 minutes in the future. In addition, different time offsets can be specified for each user on the token and/or the server. Each password will be accepted only once. After 8 successive failed authentication attempts a user gets locked out. Authentication is based on two factors: a PIN known by the user and the Init-Secret stored on the mobile device.

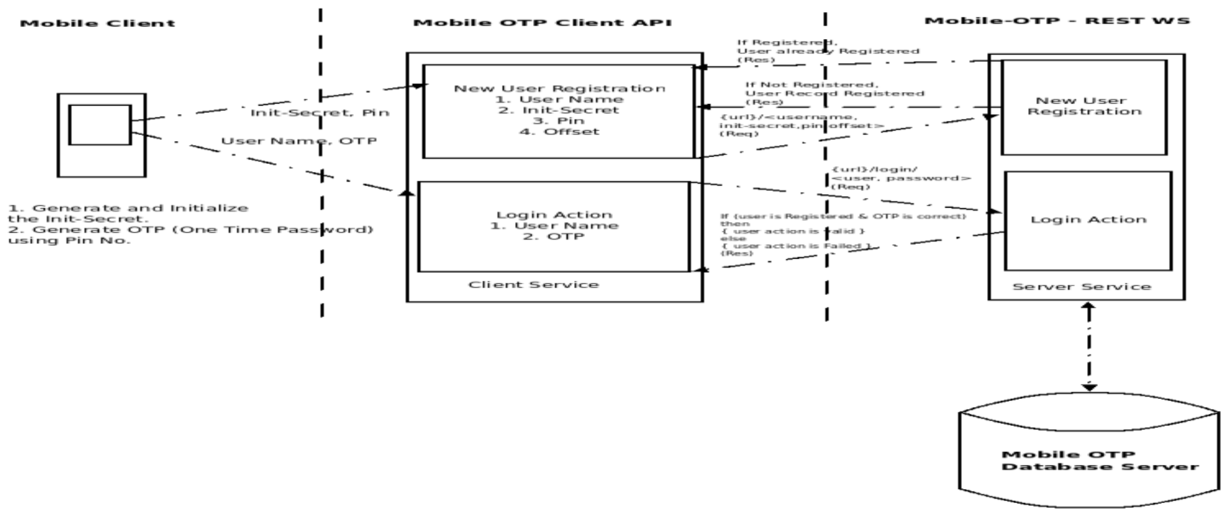

Mobile Client
Mobile OTP Client API
Mobile-OTP - REST WS
Init-Secret, Pin
User Name, OTP
1. Generate and Initialize the Init-Secret.
2. Generate OTP (One Time Password) using Pin No.
New User Registration
1. User Name
2. Init-Secret
3. Pin
4. Offset
Login Action
1. User Name
2. OTP
Client Service
If Registered, User already Registered (Res)
If Not Registered, User Record Registered (Res)
{url}/<username, init-secret,pin,offset> (Req)
{url}/login/ <user, password> (Req)
If (user is Registered & OTP is correct) then { user action is Valid } else { user action is Failed } (Res)
New User Registration
Login Action
Server Service
Mobile OTP Database Server

Fig 2: MOTP Work flow

4.1.2 HOTP
HOTP is an HMAC-based One Time Password algorithm. It is a cornerstone of Initiative for Open Authentication (OATH). HOTP was published as an informational IETF RFC 4226 in December 2005, documenting the algorithm along with a Java implementation. Since then, the algorithms were adopted by many companies worldwide (see below) and became the world's leading standard for event-based OTP authentication [2]. The HOTP algorithm is a freely available open standard.

4.1.2.1 Algorithm
Let, K is a secret key, C is a counter, HMAC (K, C) = SHA1 (K 0x5c5c… SHA1 (K 0x3636… C)) be an HMAC calculated with the SHA-1 cryptographic hash algorithm and truncate be a function that selects 4 bytes from the result of h in a defined manner.

Then HOTP (K, C) is mathematically defined by, HOTP (K, C) = Truncate (HMAC (K, C)) & 0x7FFFFFFF. The mask is to disregard the most significant bit to provide better interoperability between processors. For HOTP to be useful for an individual to input to a system, the result must be converted into a HOTP value, a 6–8 digits number that is implementation dependent.

HOTP-Value = HOTP (K, C) mod 10d, where d is the desired number of digits.

4.1.2.2 Applications
HOTP can be used to authenticate a user in a system via an authentication server. Also, if some more steps are carried out (the server calculates subsequent OTP value and sends/displays it to the user who checks it against subsequent OTP value calculated by his token), the user can also authenticate the validation server.

4.1.2.3 Tokens
Both hardware and software tokens are available from various vendors, for some of them see references below. Hardware tokens implementing OATH HOTP tend to be significantly cheaper than their competitors based on proprietary algorithms. As of 2010, OATH HOTP hardware tokens can be purchased for a marginal price.

Software tokens are available for (nearly) all major mobile/smart phone platforms (J2ME, Android, iPhone, BlackBerry, Maemo, and Windows Mobile).

4.1.2.4 HOTP API Work flow Architecture
This work flow architecture shows the overview of HOTP API and Integration of HOTP API with applications.

4.1.3 E-Token & Digital Signatures
E Token is a hardware mechanism used for password authentication via identity management technique and provides hacking problem solution to the user. It looks similar to pen drive and fixes in the USB port of the computer. This device is beneficial for government and defense organizations. Besides this it is also very useful where there security is the most thing weather personal computer.

E-Token is a smart-card chip with USB interface that attaches in the USB Port [3]. As it fixes in the USB port, it begins to work. After plug in into the router e Token asks for login to users, it encrypted the user via asking private keys, digital certification and password. Due to this user identification it allows the user to access any file or document. The user can then change the default setting of the system such as PIN (Personal Identification Number) and others. Then it store the changes and allows the user to access the system with these changes, if another user use the system and unable to match the information what eToken asks, it doesn't allow the user to access the system. EToken is not only useful for the personal computer but also the user can also use this device in the cyber café. After matching the correct identification user can access any protected file or documents. If this device is detached from the computer, the PC becomes ordinary system.

A digital signature or digital signature scheme is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit. Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery and tampering.

4.1.3.1 PKI
Public Key Infrastructure (PKI) is a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates. In cryptography, a PKI is an arrangement that binds public keys with respective user identities by means of a certificate authority (CA).

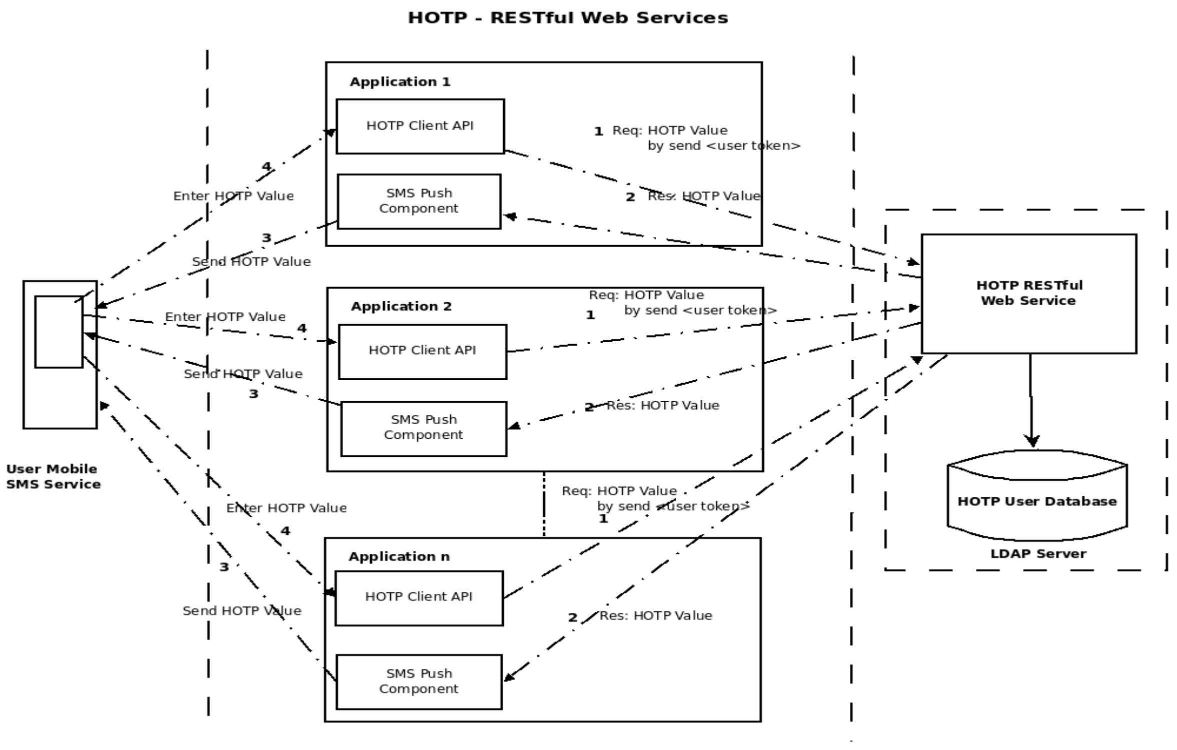


Fig 3: HOTP Work flow

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). For each user, the user identity, the public key, their binding, validity conditions and other attributes are made unforgettable in public key certificates issued by the CA.

The term trusted third party (TTP) may also be used for certificate authority (CA). The term PKI is sometimes erroneously used to denote public key algorithms, which do not require the use of a CA.

4.1.3.2 Public-key cryptography

Public-key cryptography is a cryptographic approach which involves the use of asymmetric key algorithms instead of or in addition to symmetric key algorithms. Unlike symmetric key algorithms, it does not require a secure initial exchange of one or more secret keys to both sender and receiver. The asymmetric key algorithms are used to create a mathematically related key pair: a secret private key and a published public key. Use of these keys allows protection of the authenticity of a message by creating a digital signature of a message using the private key, which can be verified using the public key. It also allows protection of the confidentiality and integrity of a message, by public key encryption, encrypting the message using the public key, which can only be decrypted using the private key.

Public key cryptography is a fundamental and widely used technology around the world. It is the approach which is employed by many cryptographic algorithms and cryptosystems. It underlies such Internet standards as Transport Layer Security (TLS) (successor to SSL), PGP, and GPG.

4.1.3.3 PKCS Standards

In cryptography, PKCS refers to a group of public-key cryptography standards devised and published by RSA Security.

Table 1. PKCS Standards

PKCS Standards	Version	Name
PKCS #1	2.1	RSA Cryptography Standard
PKCS #2	-	Withdrawn
PKCS #3	1.4	Diffie-Hellman Key Agreement Standard
PKCS #4	-	Withdrawn
PKCS #5	2.0	Password-based Encryption Standard
PKCS #6	1.5	Extended-Certificate Syntax Standard
PKCS #7	1.5	Cryptographic Message Syntax Standard
PKCS #8	1.2	Private-Key Information Syntax Standard.
PKCS #9	2.0	Selected Attribute Types
PKCS #10	1.7	Certification Request Standard

PKCS Standards	Version	Name
PKCS #11	2.20	Cryptographic Token Interface (Cryptoki)
PKCS #12	1.0	Personal Information Exchange Syntax Standard
PKCS #13	–	Elliptic Curve Cryptography Standard
PKCS #14	–	Pseudo-random Number Generation
PKCS #15	1.1	Cryptographic Token Information Format Standard

A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an accidental or intentional change to the data will change the hash value. The data to be encoded is often called the "message", and the hash value is sometimes called the message digest or simply digests. The ideal cryptographic hash function has four main or significant properties:

- It is easy to compute the hash value for any given message,
- It is infeasible to find a message that has a given hash,
- It is infeasible to modify a message without changing its hash,
- It is infeasible to find two different messages with the same hash.

4.1.3.4 PKI Standards for X.509

- PKCS#7 (Cryptographic Message Syntax Standard - public keys with proof of identity for signed and/or encrypted message for PKI)
- Secure Sockets Layer (SSL) - cryptographic protocols for Internet secure communications
- On-line Certificate Status Protocol (OCSP) / Certificate Revocation List (CRL) - this is for validating proof of identity
- PKCS#12 (Personal Information Exchange Syntax Standard) - used to store a private key with the appropriate public key certificate

4.1.3.5 On line Certificate Status Protocol (OCSP)

The On line Certificate Status Protocol (OCSP) is an Internet protocol used for obtaining the revocation status of an X.509 digital certificate. It is described in RFC 2560 and is on the Internet standards track. It was created as an alternative to certificate revocation lists (CRL), specifically addressing certain problems associated with using CRLs in a public key

- Install eToken Driver to sign the data.

infrastructure (PKI). Messages communicated via OCSP are encoded in ASN.1 and are usually communicated over HTTP. The "request/response" nature of these messages leads to OCSP servers being termed OCSP responders.

4.1.3.6 Certification Path Validation Algorithm

The certification path validation algorithm is the algorithm which verifies that a given certificate path is valid under a given public key infrastructure (PKI). A path starts with the Subject certificate and proceeds through a number of intermediate certificates up to a trusted root certificate, typically issued by a trusted Certification Authority (CA).

CRL Fields: The X.509 v2 CRL syntax is as follows. For signature calculation, the data that is to be signed is ASN.1 DER encoded. ASN.1 DER encoding is a tag, length, value encoding system for each element.

Certificate List::= SEQUENCE {

TbsCertList TbsCertList,

Signature Algorithm Algorithm Identifier,

Signature Value BIT STRING}

TbsCertList::= SEQUENCE {

Version Version OPTIONAL, -- if present, MUST be v2

Signature Algorithm Identifier,

Issuer Name,

This Update Time,

Next Update Time OPTIONAL,

Revoked Certificates SEQUENCE OF SEQUENCE {

User Certificate CertificateSerialNumber,

Revocation Date Time,

CrlEntryExtensions Extensions OPTIONAL -- if present, MUST be v2,

CrlExtensions [0] EXPLICIT Extensions OPTIONAL -- if present, MUST be v2

- Version, Time, CertificateSerialNumber, and Extensions are defined in the ASN.1
- Algorithm Identifier is defined.

4.1.3.7 DSC Work flow

- Create an Applet – to sign the form data.
- Create a JSP
    - o JSP Form to submit data.
    - o Sign the submitted data by calling the applet using JavaScript.
    - o Return the Signed value and Display in alert message.

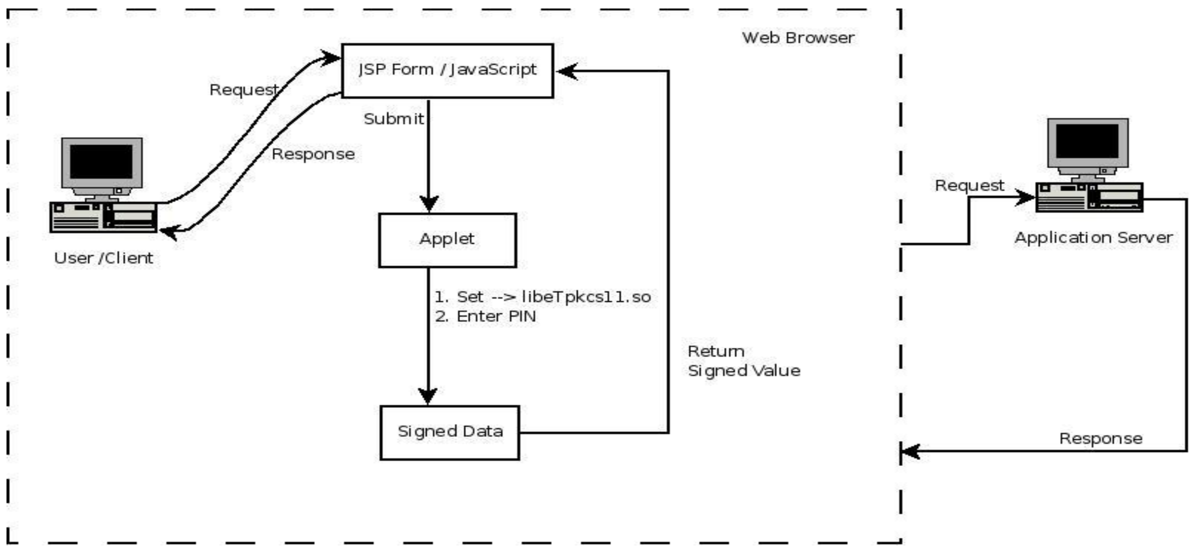


Fig 4: DSC Work flow

4.2 Extended Usernametoken Based Access to Restful Webservices

WS-Security is based on the SOAP not REST. This paper will extract the User name token from Extended User name Token and secondary password into the HTTP WS-Security, and transplant it to the HTTP header, and add header.

REST security is usually using transmission security protocol SSL / TLS, Basic and Digest Authentication. These are REST's own security features, and describe the scheme and shortages of Basic Authentication and Digest Authentication as follows:

4.2.1 Basic Authentication

HTTP Basic Authentication, which 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, its identity will be challenged by the server. Operations will be permitted only if the client provides correct certificate, otherwise correct certificate will be required and the request should be resend by client.

HTTP basic authentication provides a simple way for user authentication, but it cannot provide realm (required Resources) authentication function. Although it uses the Base64 encoding, it is not encrypted and transmits password in plain text directly. This makes it is vulnerable to replay attacks. So this authentication is only suitable for less demand on the security of systems or devices.

4.2.2 Digest Authentication

Digest authentication is a challenge-response mechanism:

- The browser sends an HTTP request (e.g. a GET) to a web server
- The server sees the URL being accessed has been configured to require Digest authentication and replies with a 401 "Authentication required" status plus a "nonce": a unique hash of several data items, one of which is a secret key known only to the server.
- The browser pops up a dialog box requesting user name and password. Once the user enters their information, an MD5 hash of the user name, password, nonce and URL are computed and the browser re-sends the original request along with the hash.
- The web server compares that hash with its own computation of the same values. If they match, the original HTTP request is allowed to complete.

4.2.3 WS-Security

This paper proposes to implement this WS-Security for RESTful web services. User name Token is an important security technology in WS-Security. The main idea is web service is assigned a unique user name and password for each user and whenever accessing the web service, this user name and password should be passed over HTTP headers.

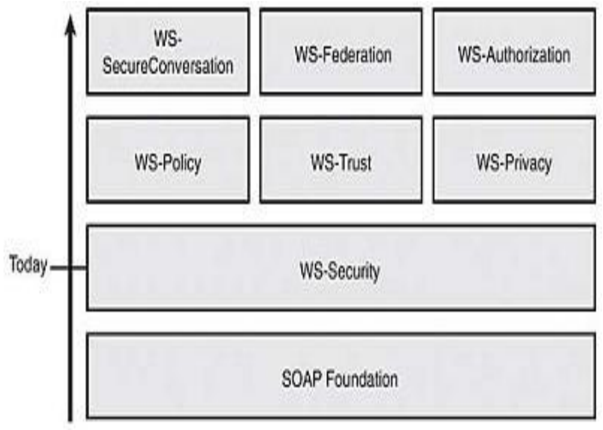


Fig 6: Web Security

This has its own limitations. In client side, the user name password has to be mentioned as plain text while calling the service. This will lead to password leakage. So, this paper proposes to use encoded user name password combination.

5. IMPLEMENTAION OF MULTIFACTOR AUTHENTICATION

The proposed system has been implemented with the following hardware and software requirements.

It requires special hardware hand held devices such as E-tokens, Java enabled Mobile. This system has been implemented in Ubuntu 10.04 Operating System with e-token drivers. This has been using J2EE with JAX RS Language with Database Libraries. This has been implemented in Apache Tomcat 6.X application server. The database has been created using PostgreSQL 8.X. NetBeans 6.9 and Eclipse Helios have been used as IDE.

5.1 MOTP Based Authentication

Fig. 7 shows the generation of init secret key from the mobile (MOTP Client). From the client application, type 0000 to generate the init secret key. Type the Random 25 digit as key. This will generate a 16 digit init secret key. These 16 digits should be keyed in during user registration.

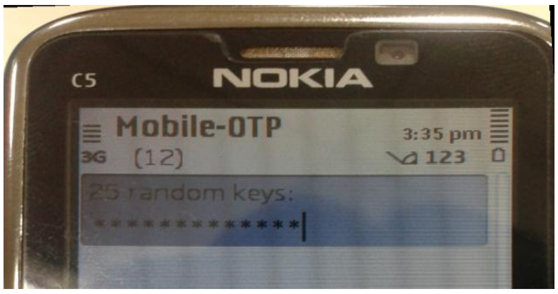


Fig 7: MOTP Init Secret Key Generation

The registration used for user creation. It collects a unique user name, IMEI number and mobile number for sending OTP. IMEI number is used as the secret key in HOTP computation. It makes the computation unique since IMEI number is unique for each user. So, the secret key won't be repeated for other users. Offset parameter is used to adjust the time according to the user's Time Zone. If set to 0, no addition or subtraction. Any other value has manipulations.

Login page just prompts for the registered unique user name and the password generated from mobile.

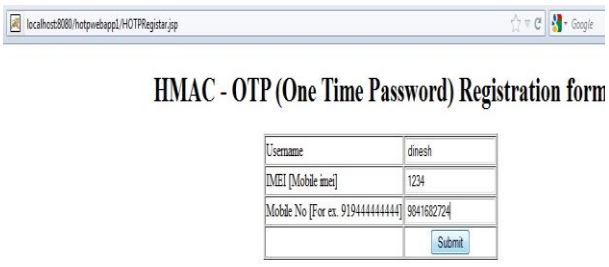
The user will be redirected to the login success page when the user has provided a correct password. Else the user will be redirected to Login Failure page. This indicates the authentication has failed.

5.2 HOTP Based Authentication

Fig. 8 shows the registration screen used for user creation. It collects a unique user name, IMEI number and mobile number for sending OTP. IMEI number is used as the secret key in HOTP computation. It makes the computation unique since IMEI number is unique for each user. So, the secret key won't be repeated for other users.

The HOTP based Login page just prompts for the registered unique user name and the user has to click Submit. This Click will POST the login page.

The user will be redirected to Login Success page when the user has provided a correct password. The user will be redirected to HOTP Login Failure page when the user has provided a wrong password. This indicates the authentication has failed.


localhost:8080/hotpwebapp1/HOTPRegistar.jsp

HMAC - OTP (One Time Password) Registration form

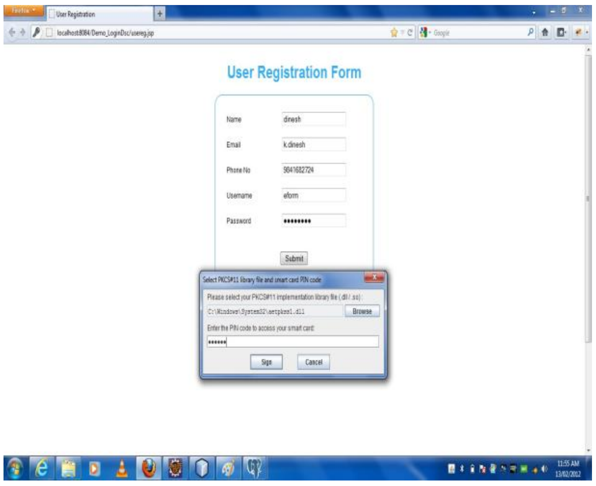
Username	dinesh
IMEI [Mobile imei]	1234
Mobile No [For ex. 919444444444]	9841682724
	Submit

Fig 8: HOTP User Registration

5.3 E-Token Based Authentication

Fig. 9 shows the registration screen used for user creation. It collects a unique user name and mobile number and E-Token. Entered details are stored in the database. For E-Token, the following details are stored:

- Certifying Authority
- Issuing Authority
- Validity of Token
- Hash value of signed data (private key + user name + password)


localhost:8084/Demo_LoginDsc/usereg.jsp

User Registration Form

Name	dinesh
Email	k.dinesh
Phone No	9841682724
Username	eform
Password	•••••••••

Select PKCS#11 library file and smart card PIN code

Please select your PKCS#11 implementation library file (.dll / .so):
C:\Windows\System32\aetpkss1.dll

Enter the PIN code to access your smart card:
••••••

Fig 9: E-Token User Registeration

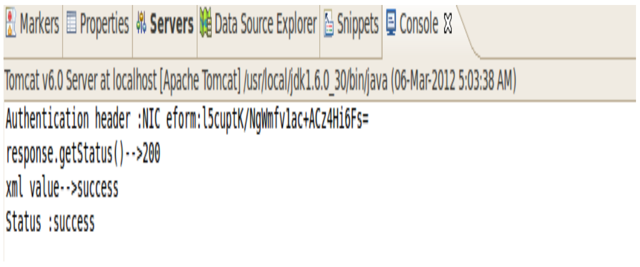
DSC based Login page just prompts for the registered unique user name & password credentials. Then, click Login, which prompts the password to validate E-TOKEN.

Once the user provides user name, password for application and password of E-TOKEN, the application generates the hash of these three parameters. It is checked with the database values. If both these values are same, then the user is redirected to login success page. Else the user is redirected to login failure page.

5.4 WS-Security for Rest Web Services

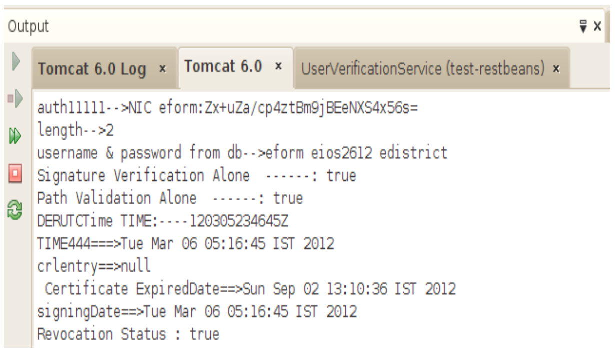
Fig. 10 shows the implementation of WS-Security for REST web services in client side. This shows the HTTP Header of the Service Request from a client. This Header contains the user name in plain text and password in encoded. Even if the HTTP request is sniffed, the sniffer doesn't see the actual password.

Fig. 11 shows the implementation of WS-Security for REST web services in server side. This shows the HTTP Header of the Service Request from a client. This Header contains the user name in plain text and password in encoded. Even if the HTTP request is sniffed, the sniffer doesn't see the actual password.



```
Tomcat v6.0 Server at localhost [Apache Tomcat] /usr/local/jdk1.6.0_30/bin/java (06-Mar-2012 5:03:38 AM)
Authentication header :NIC eform:l5cuptK/NgWmfv1ac+ACz4Hi6Fs=
response.getStatus()-->200
xml value-->success
Status :success
```

Fig 10: WS-Security for REST Web Services - Client



```
auth11111-->NIC eform:Zx+uZa/cp4ztBm9jBEeNXS4x56s=
length-->2
username & password from db-->eform eios2612 edistrict
Signature Verification Alone  ------: true
Path Validation Alone  ------: true
DERUTCTime TIME:----120305234645Z
TIME444===>Tue Mar 06 05:16:45 IST 2012
crlentry==>null
 Certificate ExpiredDate==>Sun Sep 02 13:10:36 IST 2012
signingDate==>Tue Mar 06 05:16:45 IST 2012
Revocation Status : true
```

Fig 11: WS-Security for REST Web Services - Server

5.5 Qualifier Based Access Control

Fig. 12 shows the Success page of Qualifier based access control. Whenever accessing a component, the client will pass a user name and password binded in the HTTP Header. Then the server reads the user name from the header and retrieves the Role of the specified user name from the Database stored during the registration. This role is double checked with the component's allowed Roles list. If these two matches, accessing the component is permitted. If these two doesn't matches, accessing the component is prohibited.

Login Success!

Fig 12: Qualifier Based Access Control Success Page

This paper addresses the strategies to be followed in implementing multi-factor authentication for e-governance applications through REST services. The main target of this paper is to secure both applications and the reusable shared components. Through this paper, the drawbacks of one-factor authentication and open access methodology of services were clearly mentioned.

When comparing to existing system, the proposed system fills most of the security holes. In the existing system, the security in transport layer is very less, whereas in proposed system, the data are transmitted as encoded over HTTP. In the existing system, the application data is not stored in secured manner. Hacker who gets access to database server can access to the actual data. The proposed work stores data in encrypted and encoded manner. Even though, a hacker accesses the database, the data is not understandable.

Through this paper, it is suggested to use multi-factor authentication for e-governance applications and for securing services, usage of WS-SECURITY model is recommended. Also for securing password, Base64 encoded value is transferred over HTTP header. The experimental results show the security level of both existing and proposed systems.

The following tasks can be taken as an enhancement for the proposed work.

- Extended User name Token should not be a plain text in the client side coding
- Improve the e-token based authentication for all Certifying authorities and Certificate types.
- OTP process should be enhanced to provide more Time zone specific functions.

6. REFERENCES

[1] Reeder, R. W. Schechter, "When the Password Doesn't Work: Secondary Authentication for Websites", IEEE Transaction on Security & Privacy, Vol. 9, Issue 2, pp 43 – 49, April 2011.

[2] Xu Chengqiang , Zhang Zhenli , "An integrated One-Time-Password and access control authentication scheme", 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT 10), Vol. 2, pp. 252 – 254, November 2010.

[3] Dunlu Peng, Chen Li, Huan Huo, "An Extended UsernameToken-based Approach for REST-style Web Service Security Authentication", 2nd IEEE International Conference on Computer Science and Information Technology, pp 582 – 586, August 2009.

[4] Li Liangzhi, "Research on the E-Government Scheme based on Multi- Technologies and Bi-directional Authentication", International Conference on Management of e-Commerce and e-Government (ICMECG '08), pp 124 – 127, October 2008.

[5] Kaleem Iqbal Siddiqui , Raja Iqbal , Tauseef Ahmad Rana , "Qualifier based access to web-services for portal to portal Communication", proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC '05), Vol. 1, pp 138 -144, 2005

[6] http://docs.amazonwebservices.com/AmazonDevPay/latest/DevPayDeveloperGuide/LSAPI_Auth_REST.html

[7] http://www.bouncycastle.org/csharp/resources.html

[8] http://www.akadia.com/services/ssh_test_certificate.html