Journal of Computer Science 9 (8): 998-1007, 2013
ISSN: 1549-3636
© 2013 Science Publications
doi:10.3844/jcssp.2013.998.1007 Published Online 9 (8) 2013 (http://www.thescipub.com/jcs.toc)

THE DYNAMIC NONCE BASED AUTHENTICATION SCHEME TO DEFEND INTERMEDIARY ATTACK FOR WEB SERVICES

Elangovan Uma and Arputharaj Kannan

Department of Information Science and Technolgy, Anna University, Chennai-25, India

Received 2013-04-21, Revised 2013-06-26; Accepted 2013-07-04

ABSTRACT

Nowadays, Web services have used drastically for various online applications like banking, e-bill processing. All online services need robust security architecture for handling sensitive data like user name, password. But, The Web service has security problems that need to be solved. The existing security scheme lacks to defend the attacks from replay and password guessing attacks. In this study we proposed a new scheme for a secure authentication procedure for the web service to enhance the security of the existing schemes. The proposed system has been implemented with the Dynamic Nonce for validating the user with username and password which is embedded with WS-Security. The Dynamic Nonce has been implemented with the user's mouse movement by satisfying the condition given in the proposed scheme. It has changed for every session because it is generated from user's mouse movements. The system has analyzed with possible attacks. The proposed dynamic nonce based authentication scheme is suitable for lower.

Keywords: Web Services, Authentication, DynamicNonce, WS-Security, Security Token, Time Stamp

1. INTRODUCTION

The XML web services need client authentication to prevent attacks. The system needs to share data of user information such as username, password in encrypted format which was implemented using WS-Security framework to provide message level security. WS-Security protects message contents, while transport service intermediaries and gives authentication and authorization control, which protects service provider from malicious requesters (Chang and Lee, 2012). WS-Security does not define any authentication ticket mechanism; instead, it defines how to use plain user name/password, Kerberos and X.509 tickets within the context of a SOAP header.

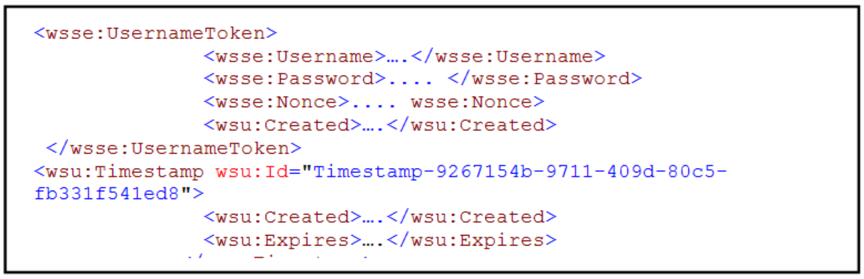
WS-Trust, WS-SecureConversation and WS-Federation define the protocols that help establish agreements between requester and provider about the kinds of security in use. WS-SecurityPolicy is used to declare a provider's requirements for security like strong authentication (Xu *et al*., 2012).

WS-Security header can be added to SOAP messages before sending to the service provider. The header should include authentication, authorization, encryption and signature. The provider can validate the credentials of the requester before executing the service. Invalid credentials, typically result in the return of an error message to the requester. Invalid credentials, will be returned to the client by an error message.

A WS-Security Username Token enables theend-user identity to be passed over multiple hops before reaching the destination Web service. The user identity is inserted into the message and is available for processing at each hop on its path. The client user name and password are encapsulated in a WS-Security <wsse:UsernameToken>. When the Enterprise Gateway receives this token, it performs the following tasks, depending on the requirements as shown in **Fig. 1**:

- Ensure that the timestamp on the token is still valid
- Authenticate the user name against a repository
- Authenticate the user name and password against a repository

Corresponding Author: Elangovan Uma, Department of Information Science and Technolgy, Anna University, Chennai-25, India



```
<wsse:UsernameToken>
              <wsse:Username>….</wsse:Username>
              <wsse:Password>.... </wsse:Password>
              <wsse:Nonce>.... wsse:Nonce>
              <wsu:Created>….</wsu:Created>
 </wsse:UsernameToken>
<wsu:Timestamp wsu:Id="Timestamp-9267154b-9711-409d-80c5-
fb331f541ed8">
              <wsu:Created>….</wsu:Created>
              <wsu:Expires>….</wsu:Expires>
```

Fig. 1. WS-security specification for authentication

The WS-Security specification, describes how to secure Web services at the message level, rather than at the transport protocol level or wire level. Existing transport-level solutions such as SSL/TLS provide solid point-to-point data encryption and authentication but have limitations if a message needs to be processed or examined by an intermediate service. For example, many organizations deploy an application-layer-filtering firewall to examine traffic before it is passed along to an internal network (Bertino *et al*., 2010).

1.1. Reviews of Related Schemes

There are many schemes have been proposed in literature, some of the schemes are reviewed in this study.

1.1.1. Review of Yang *et al*. (2005) Scheme

Yang *et al*. (2005) proposed two password schemes: Time stamp based and nonce based password authentication schemes. This authentication scheme is shown in **Fig. 2**, which is based on Diffie-Hellman Key Exchange and improves the security of the SIP authentication scheme.

1.1.2. Off-line Password-Guessing Attack on Yang's Scheme

The client sends the messages to the server, an eavesdropping adversary can get the $t_1 \oplus F$ (pw) in step1. Also, in step2, the adversary can get the $t_2 \oplus F$ (pw). Now, the adversary can compute $t_1 \oplus t_2$ from the obtained messages:

$$t_1 \oplus t_2 = (t_1 \oplus F(pw)) \oplus (t_2 \oplus F(pw))$$

The adversary can generate a password f (pw') from the following equation and derive the corresponding $t'_1 \oplus t'_2$ then verify it by checking:

$$t'_1 \oplus t'_2 = (f(pw') \oplus (t_1 \oplus f(pw))) \oplus (f(pw') \oplus (t2 \oplus f(pw))).\ t_1 \oplus t_2 \overset{?}{=} t'_1 \oplus t'_2$$

1.1.3. Review of Kim *et al*. (2005) Scheme

NONCE based scheme and Time stamp based scheme have been proposed by Kim *et al*. (2005). Each scheme has to be combined to with stand time stamp and Nonce usage in cryptanalytic attack. The Nonce based scheme needs time stamp to avoid off-line password guessing attack. The time stamp based scheme needs Nonce values to thwart the attacker from brute force attack.

1.1.4. Review of Wu and Weaver (2007) Scheme

Wu and Weaver (2007) used intermediary-based, query-based and hybrid approaches to resolve the issues for different types of information in security tokens and proposes three exchange models accordingly. Authors also provided a comprehensive framework using Web services to exchange security tokens across security domains with suitable approaches and exchange models.

1.1.5. Review of Lee *et al*. (2005) Scheme

Lee *et al*. (2005) created the sip authentication by using nonce based scheme. The scheme contains the registration phase and login phase. In registration phase, a client uses his identity (id), password (pw) and media address (ma), to construct token1 = (id||pw||ma) and h (token1).After receiving the message from user, the server genereates a random nonce, rs and computes h (rs). In the login and authentication phase of Lee scheme, there is no information to inform that the received data is fresh or replayed data which leads to replay attack.

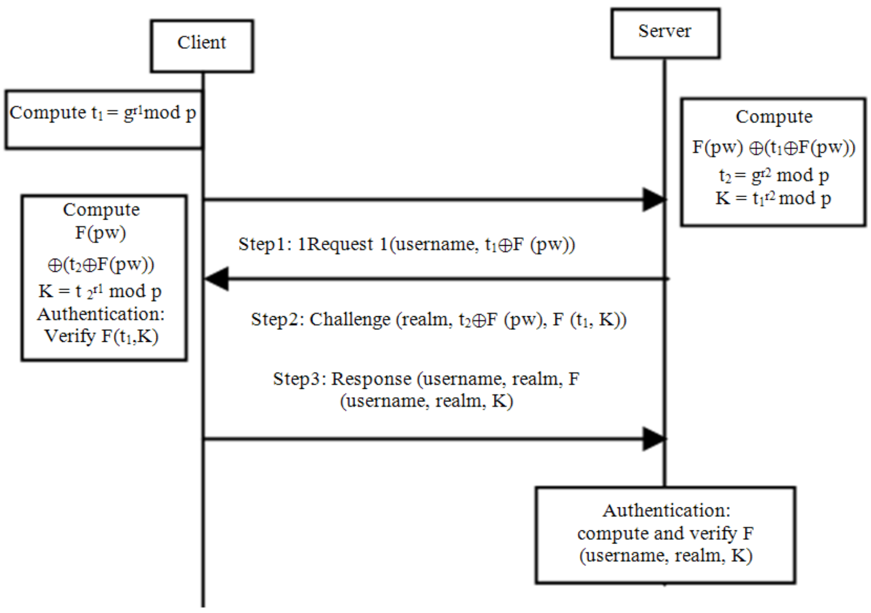


Fig. 2. Authentication phase of Yang *et al*. (2005) scheme

1.1.6. Review of Shi Scheme

The author Shi and Yoo (2006) proposed site-authenticated scheme. The system has implemented with exclusive OR operations. The server uses MAC address to protect data before sending which is not secure. The LAN card identifies the destination address in transmitting packet. In the OSI model, the network layer has to use its address to verify the device at destination. So, this scheme is not secure.

2. MATERIALS AND METHODS

The existing systems provide the static Nonce for user identification. The static Nonce can be easily hackable by brute force attack. The proposed system has been implemented with dynamic Nonce with the use of client's mouse movements.

The proposed system has been developed with one way hash function with Nonce and Time stamp. This scheme is an improved scheme from Yang *et al*. (2005) schemes. The proposed system provides secure authentication to prevent unauthorized access and to identify users for its session information, but the information returned by existing web services does not signed and encrypted. This system encodes a string that includes the password and a timestamp using the SHA-1 hashing algorithm. Including a timestamp to the password before sending the message will prevent replay attack.

1.2.1. Dynamic Nonce Generation

The Nonce has generated dynamically from the mouse movement of client system. The system calculates the different mouse movements and stores it into database for each session of communication. It collects fresh Nonce for all session. The first value of any X coordinate is treated as initial value and the Y value is considered to be a total number of iteration:

- Get X_0 coordinates to set the initial value
- Get Y_0 coordinates to set the total number of iterations
- Calculate random number R using intermediate coordinates for Y_0 number of times

$$R = X_1^{y1} + X_2^{y2} + ... + X_n^{yn} \leq$$

Dynamic Nonce $N_1 = R^y n \bmod p$

The dynamic nonce has generated using above equation P which is a large prime number. This authentication is based on Diffie-Hellman Key Exchange and improves the security of the original Web service authentication scheme.

1.2.2. Registration and Login Phase

The new user has to register the username and password to become a legitimate user of remote server. The user name and password are stored in the database of remote server. The registration and login phase of the proposed scheme is shown in **Fig. 3 and 4**.

1.2.3. Registration Phase

A new Client C_i sends the ID_C, a hashed password F (PW) and T_c to the service provider via secure channel. Then the service provider sends g and p to the respective client. Thus the C_i client registers user Id and password with the service provider.

1.2.4. Login Phase

If the Client C_i wants get the resources of service provider P, P must authenticate the user U. To accomplish this C and P must perform following steps.

The step by step procedure for an user authentication is given below.

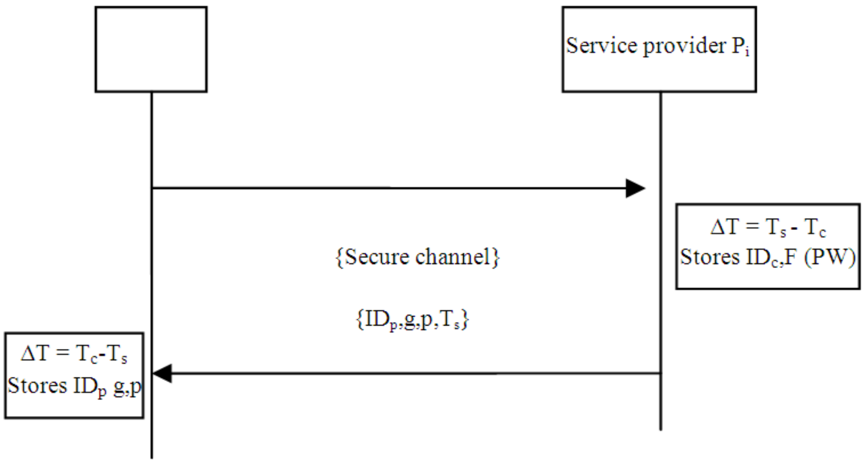


Fig. 3. Registration phase of the proposed scheme

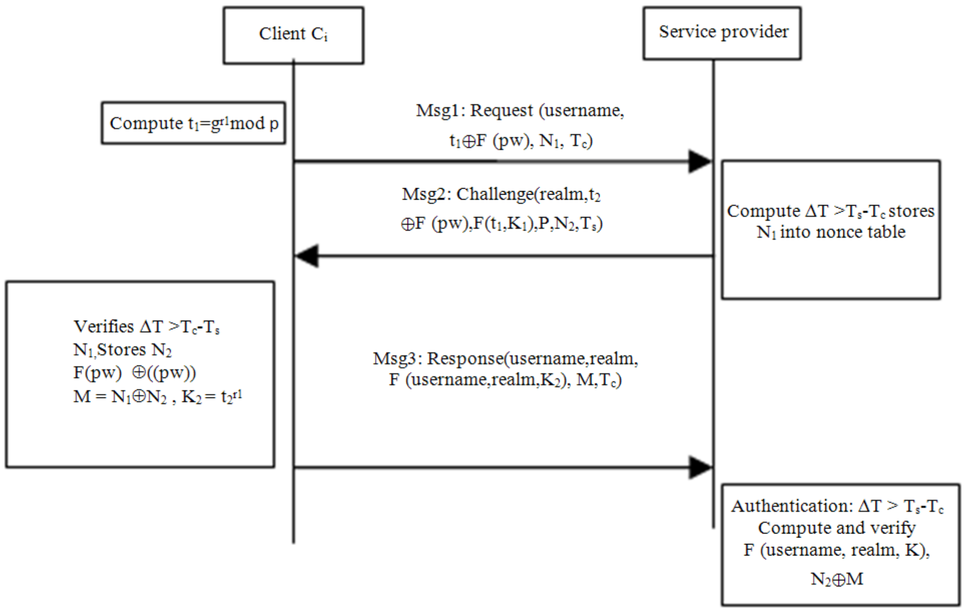

Client C_i
Service provider
Compute $t_1 = g^{r1} \bmod p$
Msg1: Request (username, $t_1 \oplus F$ (pw), N_1, T_c)
Msg2: Challenge(realm,$t_2 \oplus F$ (pw),F(t_1,K_1),P,N_2,T_s)
Compute $\Delta T > T_s - T_c$ stores N_1 into nonce table
Verifies $\Delta T > T_c - T_s$
N_1,Stores N_2
F(pw) $\oplus$((pw))
$M = N_1 \oplus N_2$, $K_2 = t_2^{r1}$
Msg3: Response(username,realm, F (username,realm,K_2), M,T_c)
Authentication: $\Delta T > T_s - T_c$
Compute and verify
F (username, realm, K),
$N_2 \oplus M$

Fig. 4. Proposed scheme for an user authentication

- C selects private random integer r1 < p and calculates $t_1 = g^{r1} \bmod p$. The value t_1 is public
- C sends the service request with a dynamic Nonce N_1, username, $t_1 \oplus F(pw)$ and T_c
- Upon receiving message from C, P checks timestamp $\Delta T > T_s - T_c$ is true and stores N_1 into Nonce **Table 1** and gets the value t_1 value by xoring. Provider P changes its id for every client from client's id and Nonce value of client
- P selects private random integer r2<p and computes $t_2 = g^{r2} \bmod p$. The value t_2 is public. Then Provider P calculates $K_1 = t_1^{r2} \bmod p$
- The provider P sends challenge message to C which holds the data of Challenge(realm,$t_2 \oplus F(pw)$,$F(t_1,K_1)$,P,N_2,T_s)
- After receiving message2 from provider, the C_i verifies ΔT, N_1 of incoming message and stores N_2.Then it calculates M by XOR the value of N_1,N_2 and $K_2 = t_2^{r1} \bmod p$
- Client C finds the t_2 value from challenge message by XOR F(pw) with $(t_2 \oplus F(pw))$. Then the client verifies $F(t_1,K)$ to authenticate the provider
- After authenticating the service provider P, it sends the responds message to Provider P Msg3: Response(username,realm,F(username,realm,K_2), (M,T_c)
- Service provider receives the response message 3 from Client C. It verifies ΔT and N_2 by XOR operation with M to get N_1 to authenticate the client
- Now Client gets the available services from services from service provider

1.2.5. Implementation

The secured system is implemented with C Sharp and WSE3.0 in .net environment to provide role based security. The proposed system supports role-based authorization of SOAP messages by constructing a security token within the SOAP message. The server and client find the incoming message and compare the password digest to a stored digest of the correct password. The timestamp must be recent otherwise the server will reject the user's login. The proposed system use public-key signatures to sign their messages so the server can be sure the contents of the message have not been viewed using diffie-helman key exchange algorithm and dynamic nonce.

3. RESULTS AND DISCUSSION

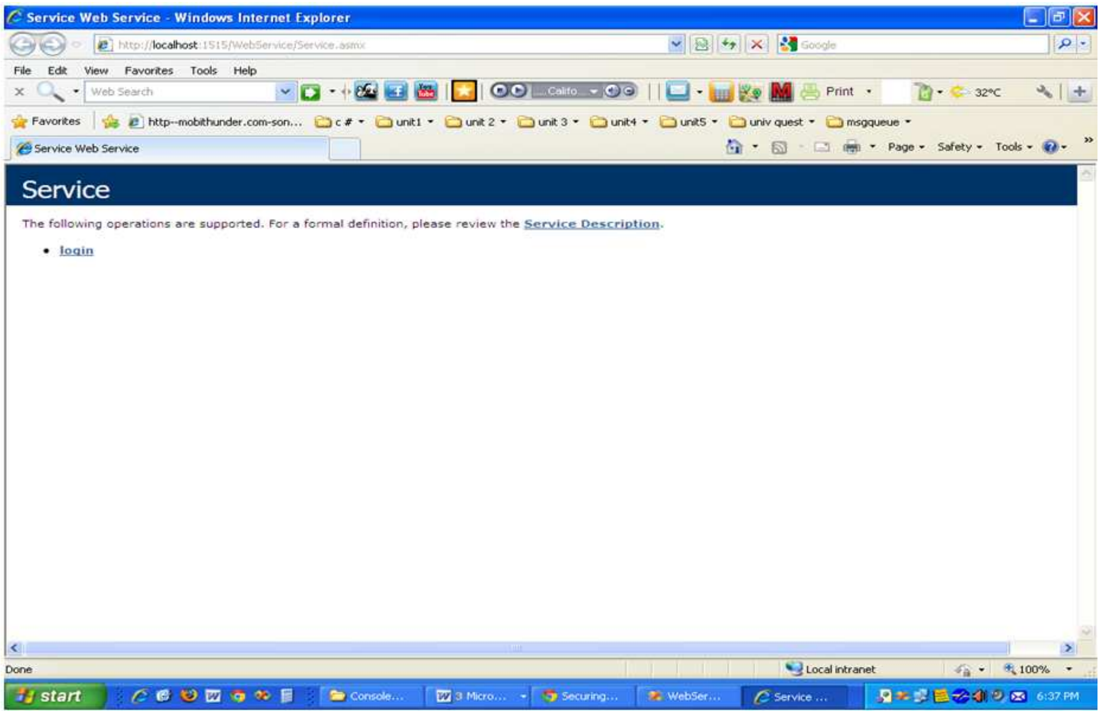
The Service provider publishes the server page to consume the service from the client side is shown in **Fig. 5**. The server maintains all incoming and outgoing message through web.config using WS-Security as shown in **Fig. 6 and 7**.

The encryption method has been implemented in username token which is implemented in authenticate token. Then, it is accessed through token checker libraries. The TokenCheckerlibrary (dll file) is registered with the web service. Through by adding TokenCheckerLibrary and UserNameToken in add tag which is under SecurityTokenManager tag of Web.Config file in **Fig. 8**. The add tag consists the details about UserNameToken like type, namespace and local name. The TokenChecker method has been called by service provider to retrieve the password of token Username to get a password from database for the given username. The service provider checks the password returned by TokenChecker and matches with the password in the SOAP header. If the password in SOAP header does not match with server's data, then server sends an exception will to the client.

<wsse:UsernameToken> element provides a countermeasure for replay attacks: <wsse:Nonce> and <wsu:Created>. A dynamic nonce is a random value which is created by sender to include in each UsernameToken that it sends. Although using a nonce is an effective countermeasure against replay attacks, it requires a server to maintain a cache of used nonces and consumes the server resources. Combining a nonce with a created timestamp has the advantage of allowing a server to limit the cache of nonces to a "freshness" time period, establishing an upper bound on resource requirements.

A first approach to prevent this could be to specify a timeout value for the token, thus a request with an expired timestamp will not be accepted by the server. If the sender sets a timestamp of 60 seconds and the server receives the message later then 60 seconds after the given <Created>value, it simply rejects the whole request. This is easy to implement, but can also have some problems, like expired messages being accepted due to clock synchronization issues on the server.

Regarding time synchronisation issues, WS-Security provides the <Timestamp> header and for it uses <MouseMove> headers for random number generation. These can be very useful for message creation, receipt and processing. The schema outline for the <Timestamp> and <wsu:MouseMove> element has displayed in **Fig. 9**.


Service

The following operations are supported. For a formal definition, please review the Service Description.

- login

Fig. 5. login service

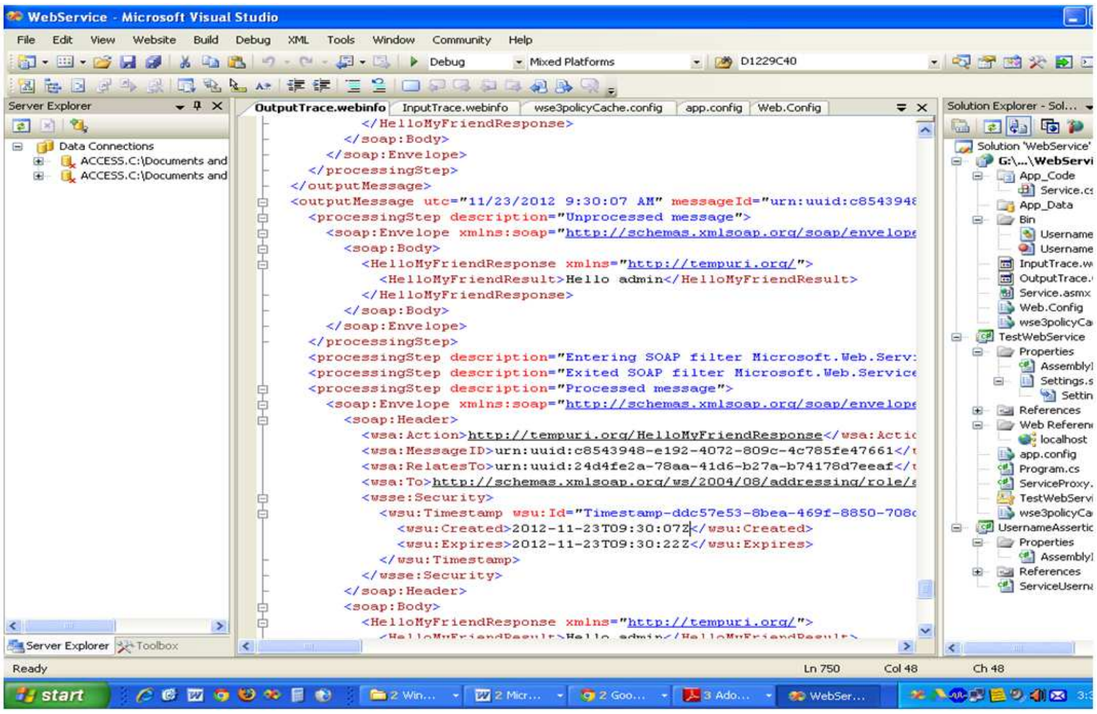
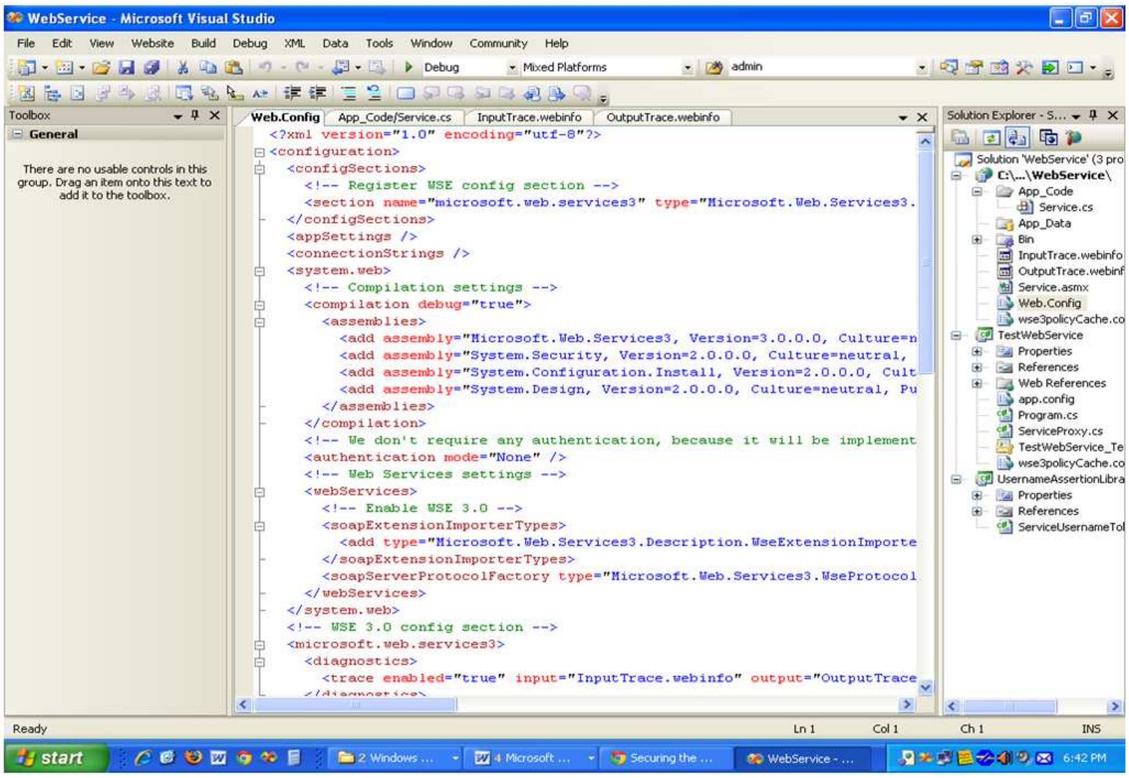

```
            </HelloMyFriendResponse>
          </soap:Body>
        </soap:Envelope>
      </processingStep>
    </outputMessage>
    <outputMessage utc="11/23/2012 9:30:07 AM" messageId="urn:uuid:c8543948
      <processingStep description="Unprocessed message">
        <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope
          <soap:Body>
            <HelloMyFriendResponse xmlns="http://tempuri.org/">
              <HelloMyFriendResult>Hello admin</HelloMyFriendResult>
            </HelloMyFriendResponse>
          </soap:Body>
        </soap:Envelope>
      </processingStep>
      <processingStep description="Entering SOAP filter Microsoft.Web.Serv:
      <processingStep description="Exited SOAP filter Microsoft.Web.Service
      <processingStep description="Processed message">
        <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope
          <soap:Header>
            <wsa:Action>http://tempuri.org/HelloMyFriendResponse</wsa:Actic
            <wsa:MessageID>urn:uuid:c8543948-e192-4072-809c-4c785fe47661</t
            <wsa:RelatesTo>urn:uuid:24d4fe2a-78aa-41d6-b27a-b74178d7eeaf</t
            <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/a
            <wsse:Security>
              <wsu:Timestamp wsu:Id="Timestamp-ddc57e53-8bea-469f-8850-708c
                <wsu:Created>2012-11-23T09:30:07Z</wsu:Created>
                <wsu:Expires>2012-11-23T09:30:22Z</wsu:Expires>
              </wsu:Timestamp>
            </wsse:Security>
          </soap:Header>
          <soap:Body>
            <HelloMyFriendResponse xmlns="http://tempuri.org/">
```

Fig. 6. Web service configuration



```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- Register WSE config section -->
    <section name="microsoft.web.services3" type="Microsoft.Web.Services3.
  </configSections>
  <appSettings />
  <connectionStrings />
  <system.web>
    <!-- Compilation settings -->
    <compilation debug="true">
      <assemblies>
        <add assembly="Microsoft.Web.Services3, Version=3.0.0.0, Culture=n
        <add assembly="System.Security, Version=2.0.0.0, Culture=neutral,
        <add assembly="System.Configuration.Install, Version=2.0.0.0, Cult
        <add assembly="System.Design, Version=2.0.0.0, Culture=neutral, Pu
      </assemblies>
    </compilation>
    <!-- We don't require any authentication, because it will be implement
    <authentication mode="None" />
    <!-- Web Services settings -->
    <webServices>
      <!-- Enable WSE 3.0 -->
      <soapExtensionImporterTypes>
        <add type="Microsoft.Web.Services3.Description.WseExtensionImporte
      </soapExtensionImporterTypes>
      <soapServerProtocolFactory type="Microsoft.Web.Services3.WseProtocol
    </webServices>
  </system.web>
  <!-- WSE 3.0 config section -->
  <microsoft.web.services3>
    <diagnostics>
      <trace enabled="true" input="InputTrace.webinfo" output="OutputTrace
```

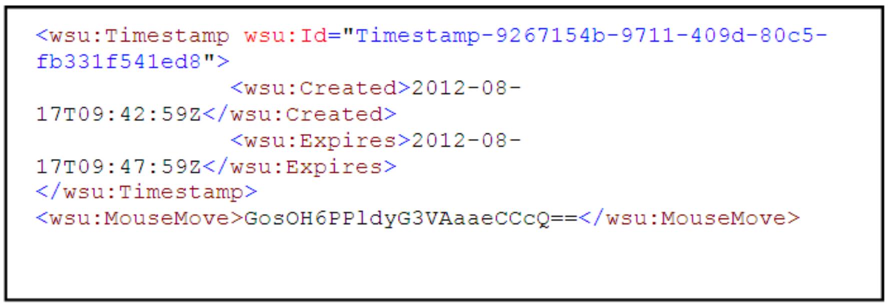
Fig. 7. Trace of WS-Security of communication flow between Client/Server

```
<wsse:UsernameToken>
<wsse:Username>admin</wsse:Username>
                <wsse:Password  Type="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-username-token-
profile-
1.0#PasswordDigest">RKFQi+jYBQpXRGnHogrRzpcmpVM=</wsse:
Password>

<wsse:Nonce>tdusE7RWcPV19ZrQqWJEJg==</wsse:Nonce>
                <wsu:Created>2012-08-
21T11:45:32Z</wsu:Created>
<wsse:Username>admin</wsse:Username>
                <wsse:Password
Type="wsse:PasswordDigest">
184D2A12DRG8D9F0C6HH82C89B091EG5C8A872F94DC188</wsse:Pa
ssword>
<wsse:Password>
lc/H00ZNNj1PbCHD4jjBH0E2vLM=</wsse:Password>

<wsse:Nonce>xj2b0r4U8maf4bvAexyFiw==</wsse:Nonce>
                <wsu:Created>2012-08-
17T09:42:59Z</wsu:Created>
 </wsse:UsernameToken>
```

Fig. 8. Hashed password and Dynamic Nonce for Authentication schemes



```
<wsu:Timestamp wsu:Id="Timestamp-9267154b-9711-409d-80c5-
fb331f541ed8">
              <wsu:Created>2012-08-
17T09:42:59Z</wsu:Created>
              <wsu:Expires>2012-08-
17T09:47:59Z</wsu:Expires>
</wsu:Timestamp>
<wsu:MouseMove>GosOH6PPldyG3VAaaeCCcQ==</wsu:MouseMove>
```

Fig. 9. Dynamic random number generation with Time stamp and mouse movement

Table 1. Some notations

Notation	Measuring
F (.)	One way hash function
PW	Password
p	Prime number
g	G<p and g is a primitive root of p
C, P	The user and the service provider
ID_x	The identity of the entity X
ΔT	Expected time interval
T_c, T_s	Time stamp of client and service provider
r_x	Private key r_x<p for entity x
T_x	Public key for entity x
K_x	Secret key of client and provider
$\oplus$	The XOR operator
	Nonce value of client and provider

Table 2. Comparison of security on resisting attacks

	Replay attack	Offline password attack	Server spoofing guessing attack	Man-in-the middle attac
Yang *et al*. (2005)	No	Yes	Yes	Yes
Wu and Weaver (2007)	No	No	Yes	Yes
Lee *et al*. (2005)	No	Yes	No	No
Shi and Yoo (2006)	No	Yes	Yes	Yes
Proposed scheme	yes	yes	yes	yes

Table 3. Performance comparison

Authentication schemes	Dynamic nonce	Time stamp
Yang *et al*. (2005)	No	No
Shi and Yoo (2006)	No	No
Proposed Scheme	yes	yes

Table 4. Comparison based on methodologies of various schemes

	Encryption	Verification table	Use MAC address	Mutual authentication
Lee *et al*. (2005)	yes	no	yes	no
Shi and Yoo (2006)	yes	yes	yes	yes
Proposed scheme	yes	yes	yes	yes

3.1. Security Analysis

The security robustness of proposed scheme has been analyzed. The comparison with the related reviewed schemes on security properties of proposed scheme is summarized in **Table 2** and its performance comparison has been listed in **Table 3 and 4**.

3.2. Replay Attack

The proposed system is a timestamp-based password authentication scheme, the replay attack is prevented by checking the freshness of the message. The expected time interval can be set by service provider. The service provider and client maintain the Nonce table to check the freshness of the random values. If the random value exceeds the expected time interval, that is T_x (N2')>ΔT, then the message of the attacker will be treated as old. Then the server or client discards the message.

3.3. Password Guessing Attack

An attacker tries to give different passwords by brute force or dictionary method from the legitimate user name in the Msg1: Request (username, t_1'⊕F(pw)', N_1', T_c'). The server will not respond the attacker's message. It is because attacker needs that exact value of t_1.

3.4. Server Spoofing Attack

In this scheme, the client and service provider pre-shares the id and password. The message passed between the client and service provider needs id of sender for authentication. Even though the attacker knows the server id, the client rejects the message, because, the client analyses all incoming message for secret values. The attacker sends the message to client Msg2: Challenge (realm', t_2'⊕F(pw)', F(t_1,K_1)', P', N_2', T_s') the received message has checked for expected validity time, freshness of N_1, F(pw) ⊕(t_2'⊕F(pw)') will return value t_2' and K_2= t_2'^{r1} mod p'is checked with value of K_1 is found false. The client will interpret that message sent by the server is a spoofed server.

3.5. Man-in-the middle Attack

An attacker cannot get the password or key value of client by using the Msg2: Challenge (realm', t_2' ⊕F(pw)', F(t_1,K_1)', P', N_2', T_s'). Because, the client checks the id of the service provider for each incoming message. Also, the client checks the validity of P', N_2', Ts'.

4. CONCLUSION

In a Web Service context, a dynamic business process may involve many applications and services from different organizations and security realms which are combined at runtime and collaborate a peer-to-peer way. Web services are a hurriedly growing technology, especially in the service oriented enterprise. Web services have a lot to offer when it comes to creating enterprise-based applications for selling things over the internet. We have developed a new authentication scheme for web services applications in low cost effective method. It is well suitable for all types of systems with lower configurations. The proposed scheme has analyzed for various types of attacks in web services environment.

4.1. Future Work

We have planned to implement it using Kerberos token for multiple web services in the federated environment.

5. ACKNOWLEDGEMENT

The researchers would like to acknowledge the financial support provided by University Grant Commission, Govt of India, New Delhi. (UGC F.No. 41-1363/2012(SR).

6. REFERENCES

Bertino, E., L. Martino, F. Paci and A. Squicciarini, 2010. Security for Web Services and Service-Oriented Architectures. 1st Edn., Springer, ISBN-10: 3540877894.

Chang, C.C. and C.Y. Lee, 2012. A secure single sign-on mechanism for distributed computer networks. IEEE Trans. Indus. Electr., 59: 629-637. DOI: 10.1109/TIE.2011.2130500

Kim, K.W., J.C. Jeon and K.Y. Yoo, 2005. An improvement on Yang et al.'s password authentication schemes. Applied Math. Comput., 170: 207-215. DOI: 10.1016/j.amc.2004.11.040

Lee, S.W., H.S. Kim and K.Y. Yoo, 2005. Efficient nonce-based remote user authentication scheme using smart cards. Applied Math. Comput., 167: 355-361. DOI: 10.1016/j.amc.2004.06.111

Shi, W. and H.S. Yoo, 2006. Efficient nonce-based authentication scheme using token-update. Comput. Sci. Appli., 3982: 213-221. DOI: 10.1007/11751595_24

Wu, Z. and A.C. Weaver, 2007. Using web services to exchange security tokens for federated trust management. Proceedings of the IEEE International Conference on Web Services, Jul. 9-13, IEEE Xplore, Salt Lake City, UT., pp: 1176-1178. DOI: 10.1109/ICWS.2007.185

Xu, J., D. Zhang, L. Liu and X. Li, 2012. Dynamic authentication for cross-realm SOA-based business processes. IEEE Trans. Services Comput., 5: 20-32. DOI: 10.1109/TSC.2010.33

Yang, C.C., R.C. Wang and W.T. Liu, 2005. Secure authentication scheme for session initiation protocol. Comput. Security, 24: 381-386. DOI: 10.1016/j.cose.2004.10.007