International Journal of Database Theory and Application
Vol.7, No.4 (2014), pp. 169-188
http://dx.doi.org/10.14257/ijdta.2014.7.4.13

Investigation on Security Implementation and Performance Aspects of MedWS: a Hierarchical SOAP based Web Service

Abhijit Bora and Tulshi Bezboruah

Department of Electronics and Communication Technology, Gauhati University,
Guwahati-781014, Assam, India
Tel: +91-361-2671262(O); Fax: +91-361-2700311(O)
abhijit.bora0099@gmail.com, zbt_gu@yahoo.co.in

Abstract

Security implementation in hierarchical SOAP based web services is essential from the perspectives of efficiency and reliability of the service. We present here a study on the use of encryption, signature and signature followed by encryption security standards in a hierarchical web service. The web service with the security policies is evaluated through the development of the application, testing of the service and evaluation of the performance metrics. The performance latencies are observed and compared to study the response of hierarchical web service against each security policies. Here we present in details the architecture of the service, software aspects, its testing procedure and the performance aspects of implemented security policy along with the outcomes of statistical analysis based on the observed performance metrics.

Keywords: *Hierarchical web service, security, encryption, signature, Java API, RDBMS*

1. Introduction

Web services (WS) have been widely accepted as platform independent services. It is a service provider for specific business logic (BL). It is similar to the activity of remote procedure call using client server application. Professionally, a WS is designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format specifically Web Service Description Language (WSDL). Other systems interact with the WS in a manner prescribed by its description using Simple Object Access Protocol (SOAP) [1].

Under the concept of Service Oriented Architecture (SOA), WS are in constant communication with other services. Their clients make requests for services through of a communication channel such as the Internet, sending and receiving information simultaneously. Another benefit is the possibility to develop WS in different languages and platforms. This technology transmits their information using two protocols, Extensible Markup Language (XML) and Hypertext Transfer Protocol Secure (HTTPS) [2].

In order to make the WS communications more secure, the SOAP message structure itself is not enough to ensure the transmission of message between sender and receiver. The inclusion of Web Service Security (WSS) specification provides security at message level throughout the communications in between clients and WS. The integrity, authenticity and confidentiality of message can be obtained through the implementation of WSS specifications. While WSS strengthen the security of WS, the performance overheads is an important factor that can come from: (a) extra CPU times to process WSS-related elements/operations at both client and services ends; (b) longer networking times to transport larger SOAP messages due to additional WSS contents [3]. Hence, the

Copyright ⓒ 2014 SERSC

performance analysis for the inclusion of security policy in WS is a major concern while developing a system with security aspects.

1.1. Related Work

In the year 2007, Shipping Chen, John Zic, Kezhe Tang, David Levy [4] presented a testing result for monitoring the performance overheads of WS due to WSS. The work includes a client that sends a request to WS for a list of data objects as customer records. It uses different message sizes and WSS security policies for performance monitoring. The WS does not execute any database related BL operation. The benchmark prototype is developed using C# and Microsoft .NET Framework 2.0. The work concluded that the cost of WSS Signing is much higher than WSS Encryption.

In the year 2011, Douglas Rodrigues, Julio C. Estrella, Kalinka R.L.J.C. Branco [5] presents a study of impact factor of security policies in system performance. This includes performance overheads of security policies in WS that calculates addition of two numbers. The WS does not perform any database related operations. The test environment includes client and WS that are implemented using Java programming language, Apache Axis2 with Apache Tomcat and Rampart 1.4 module. The monitoring expresses that response time of Signing security policy is much less than Encryption security policy.

In the year 2011, Xinfeng Ye, Lei Zhong [6] illustrated a scheme that ensures the security of the data stored on the service providers. This associates access control policy for each data item in the database.

In the year 2010, Thomas Bleier, Arndt Bonitz, Christian Wagner [7] illustrated a model driven architecture design that implements a wrapper around the existing services for security requirements in eGovernment business processes.

In the year 2012, Ji Hongbin, Zhao Fengyu, Xu Tao [8] analyzes the security mechanism of WS communication in Apache Axis2 and .Net platforms and monitors the result of these heterogeneous platforms.

2. The Objective and Methodology

The main objective of the proposed work is to implement security policies in a prototype hierarchical WS communications and test the service considering a pharmacological dataset [9] to study various attributes, like response of the service and its reliability. The WS has been developed and implemented by using Java Application Programming Interfaces (APIs), apache tomcat and MySQL database server. It has three different services, namely: (a) the parent WS, (b) the client WS and (c) the child WS. The data size of the WS is 15000. The WS has been deployed on Mercury LoadRunner for performance test. The data mapping in between diseases, medicines, components, manufacturer and clinical remarks is prepared. The flowchart of the proposed work is presented elsewhere [10]. The architecture is developed for the WS. The testing is carried out using Mercury LoadRunner testing tool for a sample of 8 virtual users. The study with 8 virtual users in our testing is justified by the fact that the server stress and connection refusal increases with the higher number of virtual users due to garbage collected heap because of improper release of memory in time. This causes the decrease of response time at server end [11-13]. Hence, the testing is performed for the scope of the work and the outcomes of the different security testing are recorded accordingly. Statistical analysis on the recorded performance metrics has been performed to analyze different aspects of the WS.

3. Security in WS

The security of information is a necessity with WS, since their business flows, processes and internal architecture are exposed. To ensure security Organization for the

Copyright ⓒ 2014 SERSC

Advancement of Structured Information Standards (OASIS) has developed WSS specification to provide message-level protection between clients and WS through message integrity, message confidentiality and message authentications. WSS makes use of SOAP's composable and extendable architecture by embedding security-related information (security token, signatures etc.) in the SOAP header without affecting the data stored in the SOAP's body (but may be encrypted/signed). This design allows WSS to integrate with SOAP as a plug-in and still retain SOAP's composability and extensibility for other purposes [4]. By applying message protection at the SOAP encoding layer we can have an access to a more flexible range of protection policies. In particular, because the SOAP layer is aware of the message structure, we can apply protection at a finer level of granularity-for example, by encrypting and signing only those headers that actually require protection. This feature enables us to support more sophisticated multi-tier architectures [14].

The WSS Policy provides a general framework for applying policies that modify the semantics of connections and communications at runtime in a WS application. It is built on top of WS–Policy framework and defines a set of policy assertions that can be used in defining individual security requirements or constraints. Those individual policy assertions can be combined using policy operators defined in the WS–Policy framework to create security policies that can be used to secure messages exchanged between a WS and a client. WS-Policy Attachment specification provides a set of policy subjects that can be configured to apply security policies. WSS Policy specification also defines how security assertions defined in the specification can be attached to policy subjects and possible policy subjects for each of those assertions [15]. The qualities of protection that can be applied to part or all of a message are: (i) Encryption, (ii) Signing and (iii) Signing followed by encryption. The binding types that can be used to protect SOAP messages in the WSS Policy specification are:

(i) sp:TransportBinding: The transport binding refers to message protection provided at the transport level (for example, through HTTPS). This binding can be used to secure any message type, not just SOAP.

(ii) sp:AsymmetricBinding: The asymmetric binding refers to message protection provided at the SOAP message encoding layer, where the protection features are implemented using asymmetric cryptography.

(iii) sp:SymmetricBinding: The symmetric binding refers to message protection provided at the SOAP message encoding layer, where the protection features are implemented using symmetric cryptography [14].

3.1. Signing and Encryption Scenario in WS

The WSS Policy specification defines policies for applying protection to individual XML elements. The scenario in **Figure 1** is a client-server application, where an asymmetric binding policy is set up to encrypt and sign the SOAP body of messages that pass back and forth between the client and the server.

When the client invokes an operation on the recipient's endpoint, the request and reply message are processed as follows:

(i) As the outgoing request message passes through the WSS Policy handler, the handler processes the message in accordance with the policies specified in the client's asymmetric binding policy. The handler performs the following:

a. Encrypt the SOAP body of the message using B's public key.
b. Sign the encrypted SOAP body using A's private key.

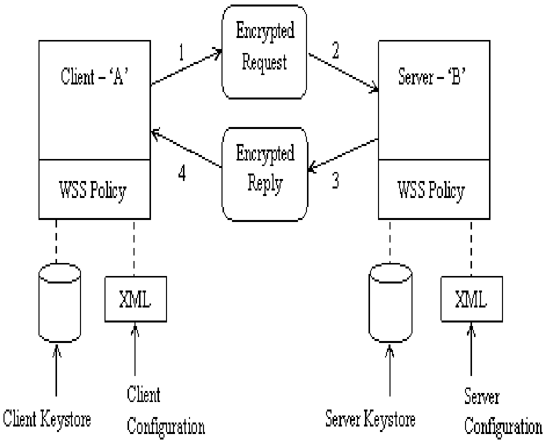

Figure 1. Basic Signing and Encryption Scenario

(ii) As the incoming request message passes through the server's WSS Policy handler, the handler processes the message in accordance with the policies specified in the server's asymmetric binding policy. The handler performs the following:

a. Verify the signature using A's public key.
b. Decrypt the SOAP body using B's private key.

(iii) As the outgoing reply message passes back through the server's WSS Policy handler, the handler performs the following processing:

a. Encrypt the SOAP body of the message using A's public key.
b. Sign the encrypted SOAP body using B's private key.

(iv) As the incoming reply message passes back through the client's WSS Policy handler, the handler performs the following processing:

a. Verify the signature using B's public key.
b. Decrypt the SOAP body using A's private key.

The security policy scenario presented below shows a WS policy for symmetric binding that supports message protection with signatures and encryption, where the signing and encryption is done using a single symmetric key.

Scenario: WSS policy scenario for symmetric binding

```
Line   Policy assertions

1      <wsp:Policy

wsu:Id="SecureConversation_MutualCertificate10SignEncrypt_IPingService_policy">

2        <wsp:ExactlyOne>

3        <wsp:All>

4        <sp:SymmetricBinding
         xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">

5        <wsp:Policy>

6         <sp:ProtectionToken>

7        <wsp:Policy>

8        <sp:SecureConversationToken>
```

```
9         ...
10       </sp:SecureConversationToken>
11      </wsp:Policy>
12      </sp:ProtectionToken>
13       <sp:AlgorithmSuite>
14      <wsp:Policy>
15       <sp:Basic256/>
16       </wsp:Policy>
17       </sp:AlgorithmSuite>
18      <sp:Layout>
19      <wsp:Policy>
20      <sp:Lax/>
21      </wsp:Policy>
22      </sp:Layout>
23      <sp:IncludeTimestamp/>
24       <sp:EncryptSignature/>
25      <sp:OnlySignEntireHeadersAndBody/>
26      </wsp:Policy>
27      </sp:SymmetricBinding>
28      <sp:Wss10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
29      <wsp:Policy>
30      <sp:MustSupportRefKeyIdentifier/>
31      <sp:MustSupportRefIssuerSerial/>
32      </wsp:Policy>
33       </sp:Wss10>
34       ...
35       </wsp:All>
36      </wsp:ExactlyOne>
37      </wsp:Policy>
```

The sp:ProtectionToken in line 6 specifies a symmetric token to use for both signing and encrypting messages. The sp:AlgorithmSuite in line 13 specifies the suite of cryptographic algorithms to use for signing and encryption. The sp:Layout in line 18 specifies whether to impose any conditions on the order in which security headers are added to the SOAP message. The sp:IncludeTimestamp in line 23 adds a timestamp element. The sp:EncryptSignature in line 24 specifies that the message signature must be encrypted. The sp:OnlySignEntireHeadersAndBody in line 25 specifies that signatures can be applied only to an entire body or to entire headers, not to sub-elements of the body or sub-elements of a header. While the WSS policies allow the integration of WS security, performance should also be considered. An overhead can occur due to the extra CPU time for processing information related to WS-Security, and time to carry SOAP messages on the network is increased because of additional content to the WS Security [16].

4. Software Aspects of MedWS

The Java language is a natural choice for developing WS. Its strong security guarantees, concurrency control and wide spread deployment in both browsers and servers makes it relatively easy to create WS [11-13]. The most important features of APIs for XML are that they all support industry standards, ensuring interoperability [13, 17]. The WS application can be developed and implemented using Java APIs with Spring framework which is Model View-Controller Model 2 (MVC2) architecture [18] and tools provided by an integrated WS Stack called Metro. The Metro stack consisting of Java API for XML WS (JAXWS), Java Architecture for XML Binding (JAXB), and WS Interoperability Technologies (WSIT), enables any one to create and deploy secure, reliable, transactional, interoperable WS and clients [19]. Different software specifications for the proposed work are: (i) web server: Apache Tomcat version 7, (ii) the database server: MySQL 5.0, (iii) platform: NetBeans 7.0 IDE and (iv) web browser: Mozilla Firefox. Along with these, jdk7 and jre7 is also used. The WS and its client applications have been run on different servers with identical hardware specifications: Intel® Xeon® CPU E5620 2.40 GHz; 8 GB RAM and 600 GB hard drive. The operating system is 64-bit Windows Server 2008 R2 Standard.

5. Design Aspects of MedWS

5.1. The Architecture

The diagrammatic representation of the architecture for the proposed WS is shown in **Figure 2**. This architecture represents the following three WS.

5.1.1. The Client WS: The client WS contains the user interface and presentation code such as Java Server Page (JSP) pages, Hyper Text Markup Language (HTML) form controls, server side class files *etc*. The HTML form controls are provided in client WS for capturing the end user data. The role of client WS is to capture data and forward it to parent WS.

5.1.2. The Parent WS: This service is responsible for capturing and forwarding the request from the client WS to child WS. The parent WS is also responsible for informing the client WS the response of child WS. The security policy is implemented in parent WS.

5.1.3. The Child WS: The child WS is responsible for executing the required BL operation. The parent WS acts as a mediator between the client WS and the child WS. The child WS manages the BL operations. It captures parameter from the parent WS and passes it to the physical storage and vice versa. The child WS holds the database queries for performing necessary operation.

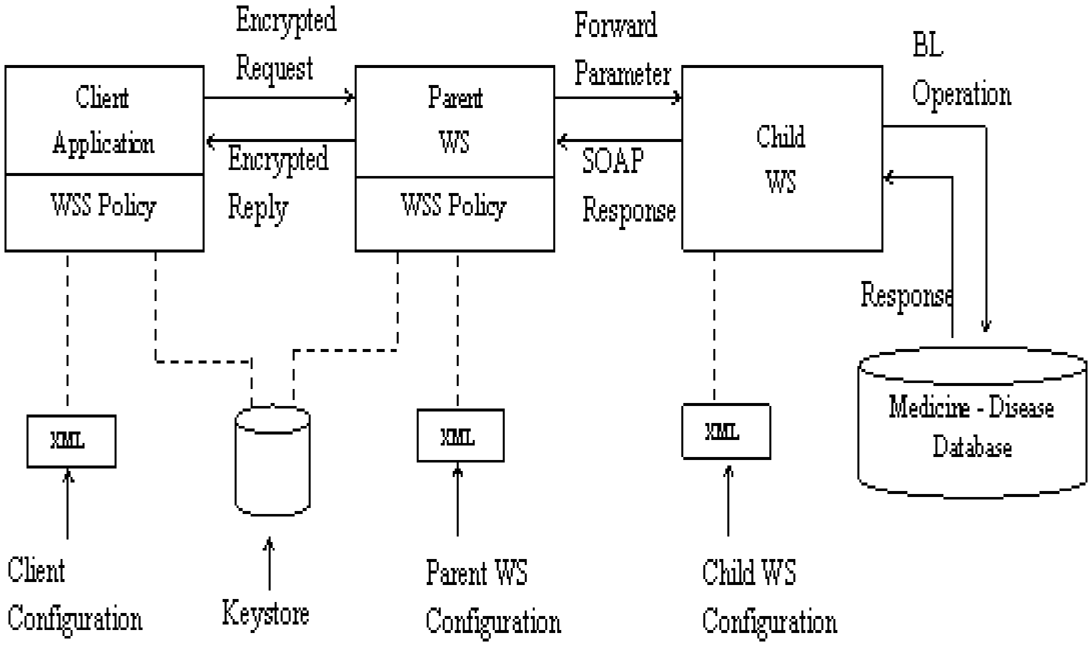

Figure 2. The Architecture for the MedWS with Security Policy for Symmetric Binding

5.2. The Algorithm

We developed the algorithm for the parent WS with the WS-security policy for encryption, signature, and signature followed by encryption.

Algorithm: Algorithm for developing the parent WS with security policies

```
Step Instruction
1. Begin
2. Establish a WS communication as a WS client using wsdl file of the child WS
3. If success
4. Go to step 6
5. Else step 2
6. Create another WS as a parent WS
7. Configure security using Username authentication symmetric key
8. If security policy required is Signature then
9. Go to step 14
10. else if security policy required is Encryption then
11. Go to step 15
12. else if security policy required is Signature followed by Encryption then
13. Go to step 16
14. Generate the security policy for Signature
15. Generate the security policy for Encryption
16. Generate the security policy for Signature followed by Encryption
17. Store received parameter from the client application of this parent WS
18. Create an object child_O of the child WS class file
19.         Invoke the child WS method containing port details using child_O to
            establish http  communication
20. Create an object child_port to capture the HTTP communication details
```

```
21.         Use child_port to invoke the desired method child_M of the child WS
22.         Pass the parameter obtained from step 17 to method child_M
23. Return the response object of the child WS to the parent's client application
    through the security policy
24. End
```

6. Evaluation Method

To evaluate the WSS overhead, we have implemented the parent WS with or without the security policies. The security settings that are implemented in the parent WS are as follows:

i. Non Secure parent WS.

ii. Encrypt security policy.

iii. Signing security policy.

iv. Signing and Encrypt security policy.

For each parent WS with or without the security policy, we have implemented the client WS as follows:

a. Non Secure WS client: The client has been developed to conform to non secure parent WS.

b. Encrypt WS client: The client has been developed to conform to encrypt security policy of parent WS.

c. Signing WS client: The client has been developed to conform to signing security policy of parent WS.

d. Signing and Encrypt WS client: The client has been developed to conform to signing with encrypt security policy of parent WS.

We use Username as symmetric public key infrastructure for encryption and signing, which is one of the most popular key infrastructures [3, 4].

7. Testing Environment

The MedWS is deployed on Mercury Load Runner version 8.1 for testing. Load Runner is an automated-testing tool that helps to predict the systems' behavior and performance. It stresses the system by creating virtual users, collecting the systems' performance information and then analyses it [20]. Each WS request causes an execution of SQL SELECT query to retrieve disease related clinical data from the database table. We follow the various steps for the test discussed and presented elsewhere [12]. The test case to select method invocation is given in Table 1 below.

7.1. Testing Parameters

The parameters that we set during the testing procedure includes: (i) the stress level, which defines the number of virtual users accessing the WS, (ii) the think time, which defines the maximum time taken by the user in thinking before requesting a parameter, and (iii) the network speed, which specifies the bandwidth that the virtual user will use in the network.

8. Experimental Results and Statistical Analysis

Four experiments have been performed for 8 virtual users. Each experiment was repeated 30 times, based on the performance evaluation of a computer system [21]. The statistical analysis of the WS with or without the security policy is performed. The observed results of WS response time are given in Tables 3-6 below to analyse the robustness of the implemented security policy for the WS.

8.1. Response Time

The average response time is presented in **Figure 3**, which represents a comparison of response time for 8 virtual users with/without the security policy implemented.

8.2. Increment Percentage

We measure and compare the response of the WS with and without the WSS settings. Through these response times a value called Response Increment Percentage (RIP), which indicates that, the percentage increase in response time of WS with security policy in relation

Table 1. Test Case for Select Method Invocation

Step	Step description	Expected outcome
1	Open URL http://server2:8084/t22EncrIO/index.jsp	Client WS index page will be displayed and contain the below fields a) “Enter keyword” text field b) “Submit” button
2	Enter valid disease name and click “Submit” button a)Enter “Cold” in “Enter keyword” text field	Pass the parameter to child WS through parent WS for necessary SQL Select operation and wait for the response.
3	Child WS response is displayed	Response page http://server2:8084/t22EncrIO/result.jsp is displayed with a result set containing following data a) Disease name b) Component name c) Medicine name d) Company name e) Remarks/Instruction

12.922

Copyright ⓒ 2014 SERSC

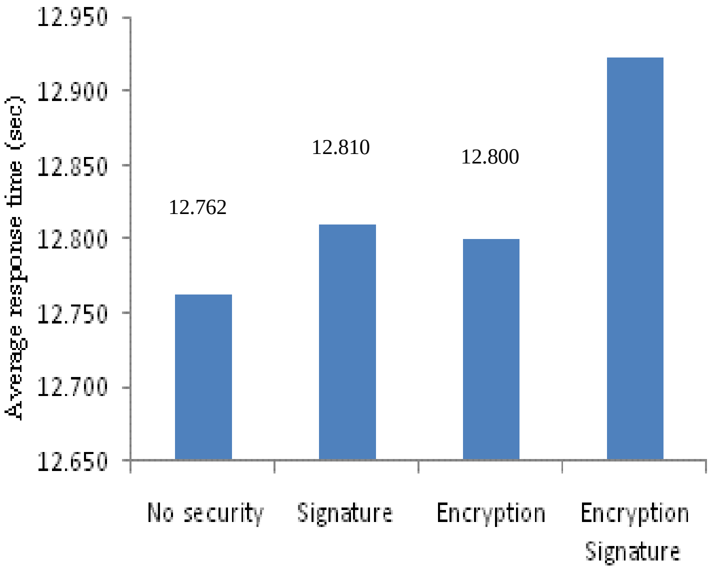

Figure 3. Comparison of 8 Virtual Users

to the response time of WS without security policy is obtained. RIP is calculated according to the equation (1) [4, 5], where RTwss is the response time of WS that includes security policy and RTnon is the response time of non secured WS.

$$\textbf{RIP} = (\textbf{RTwss} - \textbf{RTnon}) / \textbf{RTnon} \times 100\% \quad (1)$$

The statistical analysis of RIP for each security policy is shown in Figure 4. It gives a comparison of RIP between security policies. The percentage increase for 8 virtual users with or without security policies is shown in Table 2.

The table shows the average response time and increment percentage for each type of security policy in relation to the response time of WS without security. It is observed that there is a RIP value for each type of security policy, in relation to the response time of non secured WS.

Table 2. Increment Percentage for 8 Virtual Users

Security policy	**Response time**	**RIP**
No security	12.762	-
Signature	12.810	0.3761%
Encryption	12.800	0.2977%
Signature and encryption	12.922	1.253%

1.253%

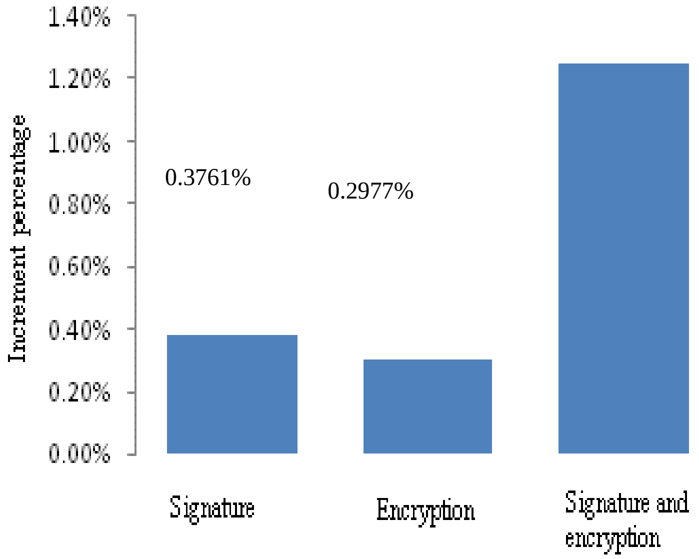

Figure 4. Comparison of 8 Virtual Users with Security Policies

8.3. Distribution of Response Time

The recorded 30 samples of 8 virtual users with or without the security policies are divided into six classes depending on their range. The bin and frequency for response time of no security, signature, encryption and signature followed by encryption security policy are given in Tables 3-6, respectively. To find out the distribution of response time we plot the histogram as shown in Figures 5(a)-5(d) respectively. According to the histogram, the applied distribution is normal but slightly right skewed for WS with or without security policy.

However, there is a major drawback with histograms, that is, depending on the used bin sizes; it is possible to draw very different conclusions. A better technique is to plot the observed quantiles against the recorded data in a quantile plot. If the distribution of observed data is normal, the plot is close to be linear [11, 12, 22]. The observed plots are shown in Figures 6(a)-6(d). The response time of WS with or without security policy shows that the distribution is normal.

Table 3. Bin and Frequency without Security Policy

Bin	Frequency
12.51	1
12.67	11
12.83	9
12.98	4
13.14	3
>13.14	2

Table 4. Bin and Frequency with Signature Security Policy

Bin	Frequency
12.53	1
12.68	12
12.84	7
12.99	0
13.15	7
>13.15	3

Copyright ⓒ 2014 SERSC

Table 5. Bin and Frequency with Encryption Security Policy

Bin	Frequency
12.62	1
12.92	21
13.23	4
13.54	2
13.85	1
>13.85	1

Table 6. Bin and Frequency with Signature followed by Encryption Security Policy

Bin	Frequency
12.62	1
12.92	21
13.23	4
13.54	2
13.85	1
>13.85	1

The normal probability plot can be used to perform the test of normality. If the data samples are taken from a normal distribution, the plot will appear to be linear [11-13]. The normal probability plots for the WS with or without security policy are shown in Figures 7(a)-7(d). The plots represent that the response time of the WS is normally distributed as the observed data is following a straight line.

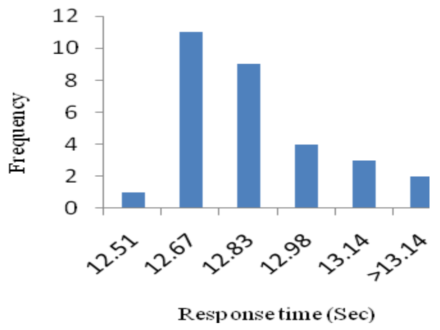

Figure 5(a). Histogram of Response Time for WS without Security Policy

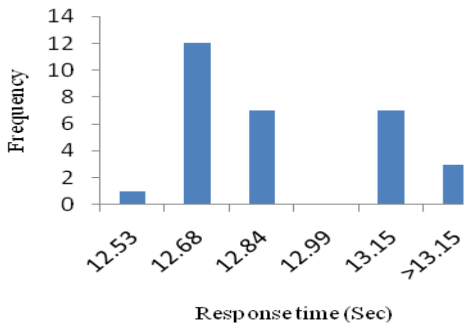

Figure 5(b). Histogram of Response Time for WS with Signature Policy

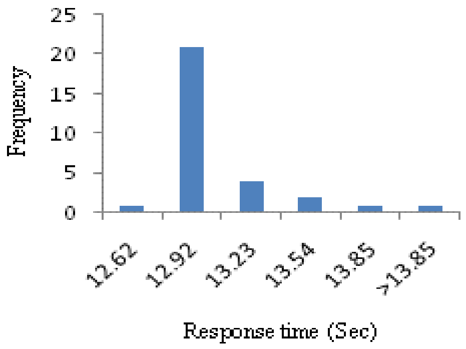

Figure 5(c). Histogram of Response Time for WS with Encryption Policy

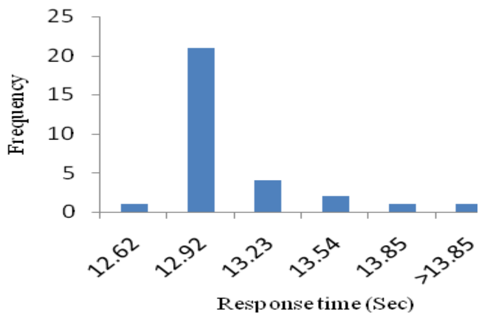

Figure 5(d). Histogram of Response Time for WS with Signature Followed by Encryption Policy

Copyright ⓒ 2014 SERSC

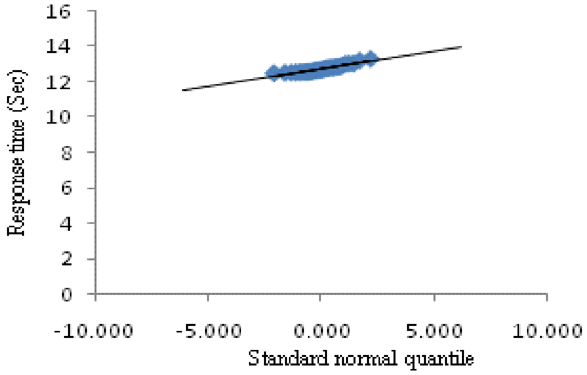

Figure 6(a). Quantile Plot of Response Time for WS Without Security Policy

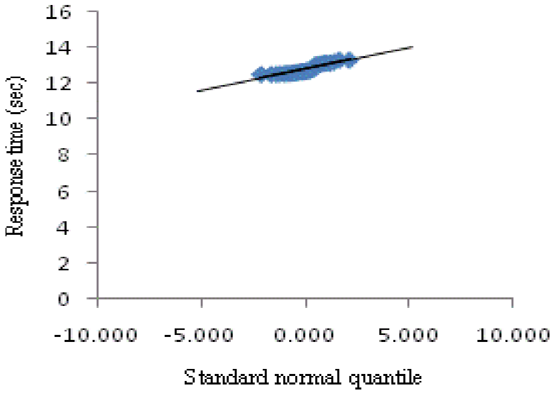

Figure 6(b). Quantile Plot of Response time for WS with Signature Policy

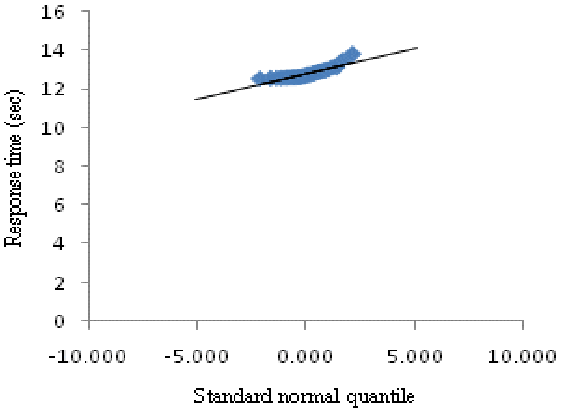

Figure 6(c). Quantile Plot of Response Time for WS with Encryption Policy

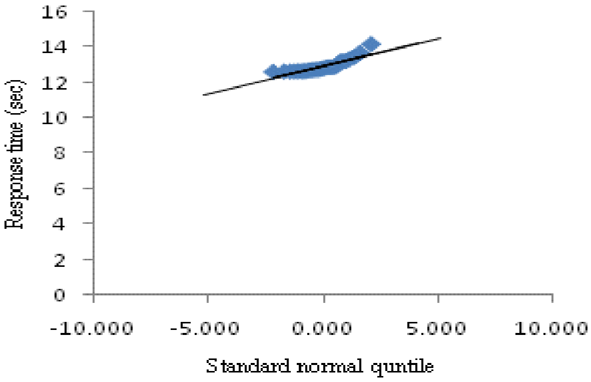

Figure 6(D). Quantile Plot of Response Time for WS with Signature Followed by Encryption Policy

8.4. Chi Square Test

Chi square test is performed to see how good the frequency distribution fits to its expected distribution [23]. It enables us to test whether there is a significant difference between an observed distribution and a theoretical distribution.

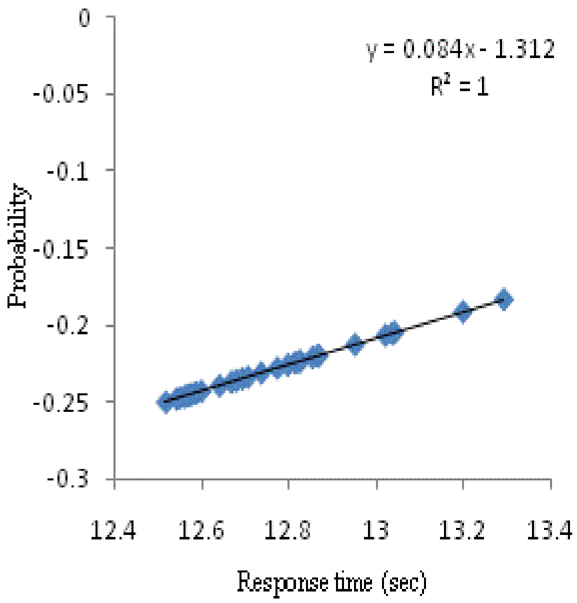

Figure 7(a). Normal Probability Plot of Response Time for WS Without Security Policy

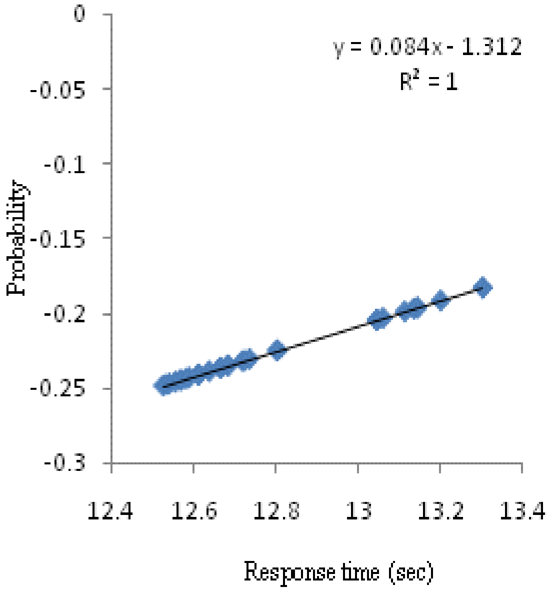

Figure 7(b). Normal Probability Plot of Response Time for WS with Signature Policy

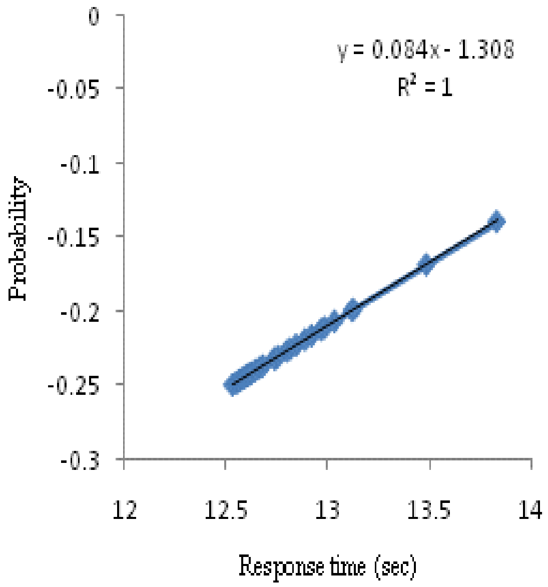

Figure 7(c). Normal Probability Plot of Response Time for WS with Encryption Policy

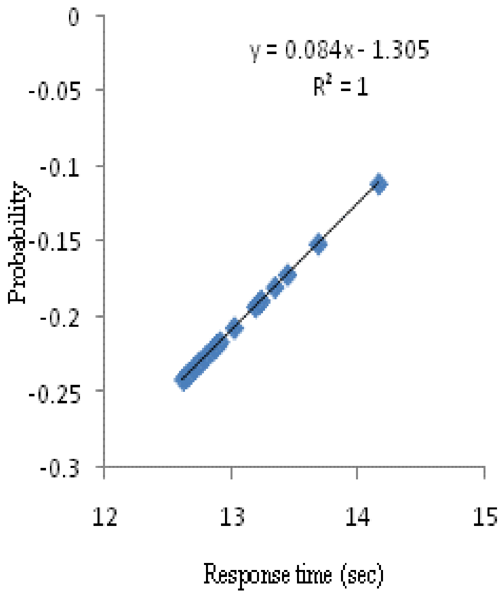

Figure 7(d). Normal Probability Plot of Response Time for WS with Signature Followed by Encryption Policy

To determine the goodness of fit between expected and observed data, the chi square equation can be expressed as [24]:

$$\chi^2 = \sum (f_0 - f_e)^2 / f_e \qquad (1)$$

Where χ^2 is calculated chi square value, f_0 is frequency of observation and f_e is expected frequency. The bigger the value of χ^2 the greater will be the difference between the observed and expected values. We compare the χ^2 value with a theoretical chi-square value obtained from the table presented elsewhere [25]. If the calculated χ^2 value is greater than the theoretical value then the fitness of good may be rejected.

It is observed from Table 7 that the sample contains 3% of response values that are 12.51, 37% in the range of >12.51 to <= 12.67, 30% in >12.67 to <= 12.83, 13% in >12.83 to <=12.98, 10% in >12.98 to <=13.14 and 7% are >13.14. The different responses of χ^2 tests for WS without security policy are given in Table 7.

From Table 8 it is observed that the sample contains 3% of response values that are 12.53, 40% in the range of >12.53 to <=12.68, 23% in >12.68 to <=12.84, 0% in >12.84 to <=12.99, 23% in >12.99 to <=13.15 and 10% are >13.15. The different responses of χ^2 tests for WS with signature security policy are given in Table 8.

From Table 9 it is observed that the sample contains 3% of response values that are 12.62, 70% in the range of >12.62 to <= 12.92, 13% in >12.92 to <= 13.23, 7% in >13.23 to <=13.54, 3% in >13.54 to <=13.85 and 3% are >13.85. The different responses of χ^2 tests for WS with encryption security policy are given in Table 9.

From Table 10 it is observed that the sample contains 3% of response values that are 12.62, 70% in the range of >12.62 to <= 12.92, 13% in >12.92 to <= 13.23, 7% in >13.23 to <=13.54, 3% in >13.54 to <=13.85 and 3% are >13.85. The different responses of χ^2 tests for WS with signature followed by encryption security policy are given in Table 10.

We assume a null hypothesis (H_o): the observed distribution fits the expected distribution. The alternate hypothesis (H_A): the observed distribution does not fit the expected distribution.

Table 7. Responses of Chi Square Test for WS without Security

Response/s	**Observed (0)**	**Expected (E)**	**O-E**	**(O-E)^2**	**(O-E)^2/E**
≤ 12.51	1	3% of 30 = 0.9	0.1	0.01	0.011
>12.51 - ≤ 12.67	11	37% of 30 = 11.1	- 0.1	0.01	0.0009
>12.67 - ≤12.83	9	30% of 30= 9	0	0	0
>12.83 - ≤ 12.98	4	13% of 30= 3.9	0.1	0.01	0.0026
>12.98 - ≤13.14	3	10% of 30= 3	0	0	0
>13.14	2	7% of 30= 2.1	- 0.1	0.01	0.0048
The calculated chi square					0.0193

The degree of freedom (DF) is calculated to be the number of levels (k) of the categorical variable minus 1: DF = k - 1. In our case it is 6-1=5. We test at the 0.05 confidence level. It is observed that the critical χ^2 value is 11.0705 for degree of freedom 5 and probability is 0.05.

Since the calculated χ^2 value is less than critical χ^2 value i.e. 0.0193 < 11.0705, we accept the null hypothesis that the data fits the expected distribution. Figure 8 shows the acceptance region of null hypothesis.

Similarly, the χ^2 value for WS with signature security policy is calculated to be 0.011 which is less than critical χ^2 values 11.0705 at degree of freedom 5 and probability 0.05 and hence we also accept the null hypothesis that the data fits the expected distribution.

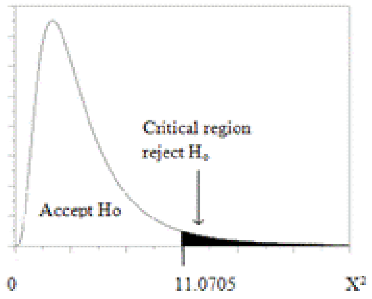

Figure 8. Goodness of Fit Test at 0.05 Level of Significance, Showing Acceptance Region

The χ^2 value for both WS with encryption security policy and signature followed by encryption security policy is calculated to be 0.0404 which is less than critical χ^2 value 11.0705 at degree of freedom 5 and probability 0.05 and hence we also accept the null hypothesis that the data fits the expected distribution.

Table 8. Responses of Chi Square test for WS with Signature Security Policy

Response/s	Observed (0)	Expected (E)	O-E	(O-E)^2	(O-E)^2/E
≤ 12.53	1	3% of 30 = 0.9	0.1	0.01	0.011
>12.53 - ≤ 12.68	12	40% of 30 = 12	0	0	0
>12.68 - ≤ 12.84	7	23% of 30= 7	0	0	0
>12.84 - ≤ 12.99	0	0% of 30= 0	0	0	0
>12.99 - ≤ 13.15	7	23% of 30= 7	0	0	0
>13.15	3	10% of 30= 3	0	0	0
				The calculated chi square	0.011

Table 9. Responses of Chi Square Test for WS with Encryption Security Policy

Response/s	Observed (0)	Expected (E)	O-E	(O-E)^2	(O-E)^2/E
≤ 12.62	1	3% of 30 = 0.9	0.1	0.01	0.011
>12.62 - ≤ 12.92	21	70% of 30 = 21	0	0	0
>12.92 - ≤ 13.23	4	13% of 30= 3.9	0.1	0.01	0.0026
>13.23 - ≤ 13.54	2	7% of 30= 2.1	- 0.1	0.01	0.0048
>13.54 - ≤ 13.85	1	3% of 30= 0.9	0.1	0.01	0.011
>13.85	1	3% of 30= 0.9	0.1	0.01	0.011
				The calculated chi square	0.0404

Table 10. Responses of Chi Square Test for WS with Signature Followed by Encryption Security Policy

Response/s	Observed (0)	Expected (E)	O-E	$(O-E)^2$	$(O-E)^2/E$
≤ 12.62	1	3% of 30 = 0.9	0.1	0.01	0.011
>12.62 - ≤ 12.92	21	70% of 30 = 21	0	0	0
>12.92 - ≤ 13.23	4	13% of 30= 3.9	0.1	0.01	0.0026
>13.23 - ≤ 13.54	2	7% of 30= 2.1	- 0.1	0.01	0.0048
>13.54 - ≤ 13.85	1	3% of 30= 0.9	0.1	0.01	0.011
>13.85	1	3% of 30= 0.9	0.1	0.01	0.011
				The calculated chi square	0.0404

9. Results and Discussion

The present investigation aims at monitoring the performance aspects of WS with or without using security policy in hierarchical WS communication based on JAVA technique using tomcat web server and predicts the influence of encryption, signature and signature followed by encryption security policy on response time.

For the sample of 8 virtual users average response time of the WS without security policy is 12.762 s, with signature security policy is 12.810 s, with encryption security policy is 12.800 s and with signature followed by encryption security policy is 12.922 s.

It is observed that, the lowest response time is 12.762 s for non secured WS, followed by the WS with encryption that is 12.800 s and WS with digital signature that is 12.810 s, and the largest response time is 12.922 s for WS with digital signature followed by encryption.

Again with 8 virtual users, the lowest RIP is 0.2977% for WS with encryption, followed by the WS with digital signature that is 0.3761%, and finally, the highest RIP is 1.253% for WS with digital signature followed by encryption.

In our investigation, the default WSS for quality of service setting in NetBeans is set to sign body part, addressing information, message reliability part of SOAP message structure and for encryption purpose the setting is configured to encrypt only the body part of the SOAP message structure. This might be the reason for which the response time of WS with signature security policy is greater than the response time of WS with encryption security policy.

The histograms are slightly right skewed. However, the normal probability plots and quantile plots of the recorded metrics with or without security policy for response time are linear and normally distributed which evident that the service is robust and reliable. We observe straight line in normal probability plot, which resembles normality.

The χ^2 tests of goodness of fit using response time of WS with or without security policy resembles that the observed data meets the expected data and hence fits the expected distribution.

10. Conclusions

A study on the inclusion of security policy in a hierarchical SOAP based medical WS and evaluating the performance of the service against each security policy specification is presented in this paper.

From our above investigations it can be concluded that response time of WS with signature security policy is greater than the response time of WS with encryption security policy using username as public key for deployment. The response time of WS with signature followed by encryption security policy seems to be the sum of each response times from individual security policy of signature and encryption settings.

The statistical analysis on recorded data shows that distributions of WS response time are normal and the observed parameters are similar to the expected parameters.

From the above analysis we can conclude that inclusion of security policy in WS increases the performance latency of the service.

The default WSS settings in NetBeans can be configured to sign body part, addressing information, message reliability information of SOAP message structure. Hence before making a decision planning on security implementation using signature and encryption security policy, one should concern about the fact that sign and encrypt only the most specific portion of the SOAP message structure to reduce the degradation of WS performance.

Acknowledgments

The authors are thankful to the All India Council of Technical Education (AICTE), Govt. of India for financial support towards the work (Grant No. 8023/BOR/RID/RPS (NER)-84/2010-2011 31st March 2011).

References

[1] Available at: http://www.w3.org/TR/ws-arch/.
[2] M. I. P. Salas and E. Martins, “Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-Security”, Elsevier, **(2014)**, pp. 133-154.
[3] K. Tang, S. Chen, D. Levy, John Zic and Bo Yan, “A performance evaluation of web service security”, Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference, EDOC, **(2006)**.
[4] S. Chen, J. Zic, K. Tang and D. Levy, “Performance Evaluation and Modeling of Web Services Security”, IEEE International Conference on Web Services, ICWS, **(2007)**, pp. 431-438.
[5] D. Rodrigues, J. C. Estrella and K. R. L. J. C. Branco, “Analysis of Security and Performance Aspects in Service-Oriented Architectures”, International Journal of Security and its Applications, vol. 5, no. 1, **(2011)** January, pp. 13-29.
[6] X. Ye and L. Zhong, “Improving Web Service Security and Privacy”, IEEE World Congress on Services, (DOI 10.1109/SERVICES.2011.109), **(2011)**, pp. 406-413.
[7] T. Bleier, A. Bonitz and C. Wagner, “Efficient Security Implementation in eGovernment Workflows: a Practical Application”, World Telecommunication Congress, Vienna, **(2010)**, pp. 198-200.
[8] J. Hongbin, Z. Fengyu and X. Tao, “Security Policy Configuration Analysis for Web Services on Heterogeneous Platforms”, International Conference on Applied Physics and Industrial Engineering, Elsevier, (doi:10.1016/j.phpro.2012.02.211), **(2012)**, pp. 1422-1430.
[9] Drug Index, Passi Publications, India, **(2012)** January-March.
[10] A. Bora, M. K. Bhuyan and T. Bezboruah, “Investigations on Hierarchical Web service based on Java Technique”, Proceedings of the World Congress on Engineering (WCE), London, UK, vol. 2, pp. 891-896, **(2013)** July.
[11] M. Kalita, S. Khanikar and T. Bezboruah, “Investigation on performance testing and evaluation of PReWebN: a JAVA technique for implementing web application”, IET Softw., vol. 5, no. 5, pp. 434-444, **(2011)**.
[12] M. Kalita and T. Bezboruah, “Investigation on performance testing and evaluation of PReWebD: a .NET technique for implementing web application”, IETSoftw., vol. 5, no. 4, **(2011)**, pp. 357-365.
[13] M. Kalita and T. Bezboruah, “Investigation on implementation of web applications with different techniques, IETSoftw., vol. 6, no. 6, **(2012)**, pp. 474-478.
[14] Available at: http://fusesource.com/docs/framework/2.4/security/MsgProtect-SOAP- Intro.html.
[15] Available at: http://wso2.com/library/3132/.
[16] S. S. Yau, Y. Yin and H. G. An, “An Adaptive Tradeoff Model for Service Performance and Security in Service-Based Systems”, Proceedings of the IEEE International Conference on Web Services, ICWS, IEEE Computer Society, Washington, DC, USA, **(2009)**, pp. 287-294.
[17] A. S. Christensen, A. Moller and M. I. Schwartzbach, “Extending Java for high level web service construction”, ACM Trans. Program. Lang. Syst., vol. 25, no. 6, **(2003)**, pp. 814-875.
[18] Available at: http://www.javasamples.com/showtutorial.php?tutorialid=350
[19] Available at: http://www.oracle.com/technetwork/java/index-jsp-137004.html
[20] Application-testing tool: Mercury LoadRunner 8.0, Available at: http://www.pcquest.com/pcquest/news/183659/application-testing-tool-mercury-loadrunner-80

[21] R. K. Jain, The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling, Wiley, April **(1991)**.

[22] A. Bogardi-Meszoly, Z. Szitas, T. Levendovszky, H. Charaf, Investigating factors influencing the response time in ASP.NET web applications, LNCS, pp. 223-233, **(2005)**.

[23] Available at: http://fsweb.bainbridge.edu/dbyrd/statistics/goodnessfit.htm

[24] I. Levin, S. Richard, D. Rubin, Statistics for management, Pearson education, Inc., South Asia, **(2009)**.

[25] Available at: http://www.statisticslectures.com/tables/chisquaretable/

Authors

Abhijit Bora, Research Scholar, Department of Electronics and Communiaction Technology, Gauhati University, India received Master of Computer Applications (MCA) degree from Jorhat Engineering College (Under Dibrugarh University), India in 2008. His research interests include web service, web security and software engineering.

Tulshi Bezboruah, (M’12) received the B.Sc. degree in physics with electronics from the University of Dibrugarh, Dibrugarh, India, in 1990, and the M.Sc. and Ph.D. degrees in electronics and radio physics from the University of Gauhati, Guwahati, India, in 1993 and 1999, respectively. In 2000, he joined in the Department of Electronics Science, Gauhati University, as a Lecturer. He is currently the Professor & Head, Department of Electronics and Communication Technology, Gauhati University. His current research interests include instrumentation and control, distributed computing, and computer networks. Prof. Bezboruah is a member of the IEEE Geoscience and Remote Sensing Society as well as an Associate Member of the International Center for Theoretical Physics, Trieste, Italy.

Copyright ⓒ 2014 SERSC

Copyright ⓒ 2014 SERSC