4. System Overview

The Web Services architecture is based upon the interactions between three roles: service provider, service registry and service requestor. The interactions involve the publish, find and bind operations. Together, these roles and operations act upon the Web Services artifacts: the Web service software module and its description. In a typical scenario, a service provider hosts a network-accessible software module (an implementation of a Web service). Service provider and service requestor roles are logical constructs and a service can exhibit characteristics of both. Figure 3 illustrates these operations, the components providing them and their interactions. Since both SOAP and WSDL are XML-based, XML messages have to be parsed on both the server and the client side and proxies have to be generated on the client side before any communication can take place. Which may result in a longer response time of the server in case of a Web service server.

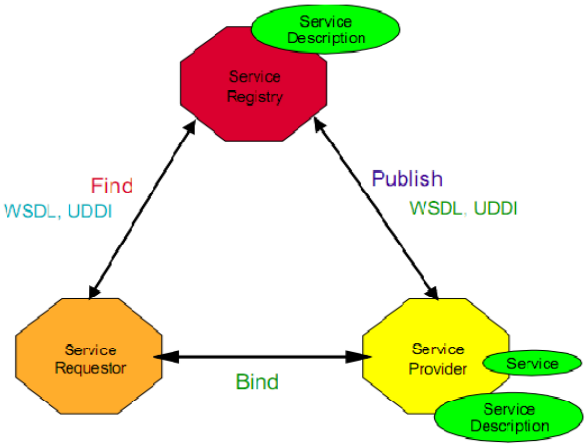

Figure 3: Web Services Architecture

4.1 Roles in Web Services Architecture

1. **Service provider:** From a business perspective, this is the owner of the service. From an architectural perspective, this is the platform that hosts access to the service.
2. **Service requestor:** From a business perspective, this is the business that requires certain functions to be satisfied. From an architectural perspective, this is the application that is looking for and invoking or initiating an interaction with a service. The service requestor role can be played by a browser driven by a person or a program without a user interface, for example another Web service.
3. **Service registry:** This is a searchable registry of service descriptions where service providers publish their service descriptions. Service requestors find services and obtainbinding information (in the service descriptions) for services during development for static binding or during execution for dynamic binding.

4.2 Operations in Web Service Architecture

For an application to take advantage of Web Services, three behaviors must take placepublication of service descriptions, lookup or finding of service descriptions, and binding or invoking of services based on the service description. These behaviors can occur singly or iteratively. In detail, these operations are:

1. **Publish:** To be accessible, a service description needs to be published so that the service requestor can find it. Where it is published can vary depending upon the requirements of the application.
2. **Find:** In the find operation, the service requestor retrieves a service description directly or queries the service registry for the type of service required.
3. **Bind:** Eventually, a service needs to be invoked. In the bind operation the service requestor invokes or initiates an interaction with the service at runtime using the binding details in the service description to locate, contact and invoke the service.

Artifacts of a Web Service

1. **Service:** Where a Web service is an interface described by a service description, its implementation is the service. A service is a software module deployed on network accessible platforms provided by the service provider. It exists to be invoked by or to interact with a service requestor. It can also function as a requestor, using other Web Services in its implementation.
2. **Service Description:** The service description contains the details of the interface and implementation of the service. This includes its data types, operations, binding information and network location. It could also include categorization and other metadata to facilitate discovery and utilization by service requestors. The service description might be published to a service requestor or to a service registry. The Web Services architecture explains how to instantiate the elements and implement the operations in an interoperable manner.

5. Web Services Security

Web services context, security means that the recipient of a message should be able to verify the integrity of the message and to make sure that it has not been modified. WS-Security from OASIS defines the mechanism to include integrity, confidentiality, and single message authentication features within a SOAP message. WS-Security makes use of the XML Signature and XML Encryption specifications and defines how to include digital signatures, message digests, and encrypted data in a SOAP message. WS-Security is concerned with security for SOAP messages, thus, WS Security clearly builds on top of SOAP. In addition, WS Security also makes use of XML Signature and XML Encryption. The Web Services Security (WSS) specifications aim to provide a framework for building secure Web services using SOAP, and consist of a core specification and several additional profiles. The core specification, the Web Services Security: SOAP Message Security specification, defines a security header for use within SOAP messages and defines how this security header can be used to provide confidentiality and integrity to SOAP messages.

The recipient should have received a message confidentially so that unauthorized users could not read it, know the identity of the sender and determine. These are usually met through encrypting messages. Security is critical to the

adoption of Web services by enterprises, but the Web services framework does not meet basic security requirements. The fact that Web services involve exchange of messages means that securing the message exchange is an important issue to consider when building and using Web services. A few standards have come out to alleviate the message security problem, including WS Security and various other initiatives towards enabling digital signatures on XML messages and transactions.

In general, there are four basic security requirements that the Web Services security layer must provide:

1. **Confidentiality** is the property that information is not made available or disclosed to unauthorized individuals, entities, or processes, and guarantees that the contents of the message are not disclosed to unauthorized individuals.
2. **Authorization** is the granting of authority, which includes the granting of access based on access rights and guarantees that the sender is authorized to send a message.
3. **Data integrity** is the property that data has not been undetectably altered or destroyed in an unauthorized manner or by unauthorized users thereby insuring that the message was not modified accidentally or deliberately in transit.
4. **Proof of origin** is evidence identifying the originator of a message or data. It asserts that the message was transmitted by a properly identified sender and is not a replay of a previously transmitted message. This requirement implies data integrity.

6. Security Algorithms

Web Service security is big challenge for researchers as it requires a strong security algorithm for the encryption of data. The xml encryption scheme is being used presently for encrypting the messages between the different programming languages running on different platforms, but this xml encryption algorithm is symmetric key encryption algorithm and it creates communication overhead, hence there is need to use an asymmetric key encryption algorithm.

The more powerful version of DES is used for high security called Triple-DES. To start encrypting with Triple-DES, two 56-bit keys are selected. Data is encrypted via DES three times, the first time by the first key, the second time by the second key and the third time by the first key once more

AES is a newer encryption standard and is now the preferred one to use for XML Encryption. AES is a substitution-linear transformation network with 10, 12, or 14 rounds, depending on the key sizes, which are currently set at 128, 192, or 256 bits. The block size used in AES is 16 bytes. The data block to be processed is partitioned into an array of bytes forming a matrix with rows and columns. Each cipher operation is byte-oriented.

7. Complexity and Overhead

The HTTP may in theory seem simple, when used by modern HTTP servers, clients, and proxies it is not. HTTP has evolved into a highly complex protocol as used between modern servers and browsers. A large number of features and optional headers may be employed, increasing embedded device complexity. The use of XML as a payload adds further parsing complexity. In RPC web services this is further compounded by the use of SOAP.

The RESTful web service paradigm needs to be extended into the constrained domain. To do thiswe need a fresh approach to both the transfer protocol used to convey REST semantics, and the payload formats exchanged between applications. At the same time, this needs to be done asa natural extension of today's HTTP web. The complexity of creating and parsing content must be minimized for very constrained devices at the same time.

7.1 Performance Results in overhead

First we measured local performance for all scenarios, where all tests were run on the same computer to avoid network overhead. The results are shown in Table1. Instantiation was considered separately (shown as instantiation). We can see that the times for basic data types (int, short, long, float, double, boolean, and byte) do not differ considerably, therefore we have calculated the geometric average

Table1: Local performance results

Time in ms	*RMI*	*HTTP-to-port*	*HTTP-to-servlet*	*Web services*
Instantiation	0,686	19,070	19,483	0,616
Simple types average:	0,157	5,052	7,663	2,336
- int	0,157	5,044	7,659	2,330
- short	0,156	5,041	7,689	2,356
- long	0,156	5,050	7,622	2,319
- float	0,157	5,052	7,680	2,375
- double	0,161	5,066	7,689	2,342
- boolean	0,155	5,059	7,670	2,316
-byte	0,155	5,050	7,631	2,313
String	0,172	5,105	7,658	2,353

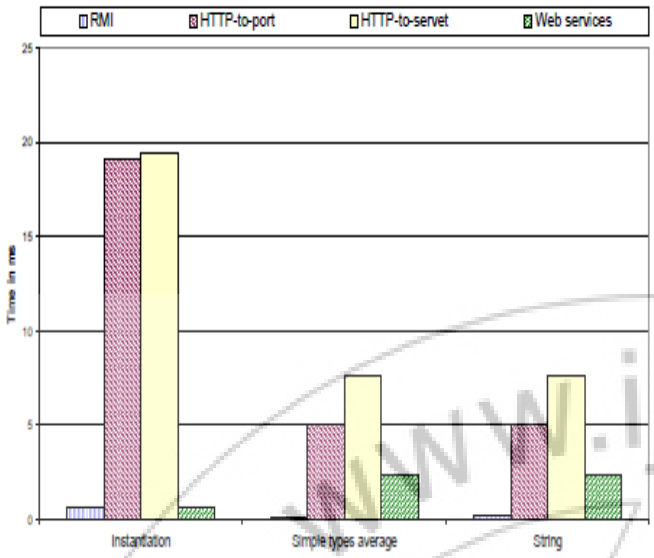

Figure 4: Performance results without network overhead

From Figure .4 is RMI offers results, which are an order of magnitude better than the other alternatives. Web services are the second fastest alternative. By the basic data types, they are on average ~15 times slower than RMI, by string, web services are~13.6 times slower. It is interesting, that by the instantiation, web services are ~10% faster than RMI. The RMI tunneling scenarios are even slower. HTTP-to-port tunneling is ~32 times slower on average basic data types and ~30 times slower on string than RMI and still ~2 times slower than web services on basicdata types and string.

Figure .5 in the remote network scenario RMI still performs much faster than the other alternatives. The average factor comparison however shows a slightly different picture than in local scenario .HTTP-to-servlet tunneling is now considerably faster than HTTP-to-port. The reason can be found in the internals of proxy software and the fact that HTTP-to-port works almost 80% slower when used over a network, because

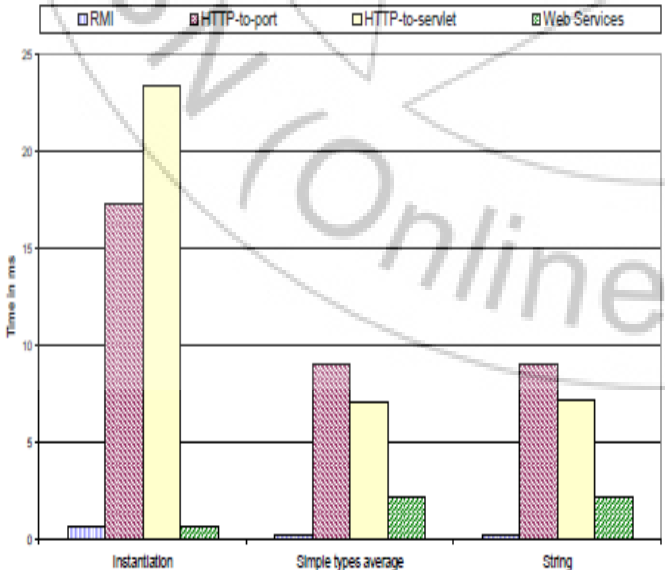

Figure 5: Performance results with network overhead

Local optimizations cannot be applied any more. HTTP-to-servlet is almost ~49 times slower on basic data types and ~44 times on string than RMI and ~3.2 times than web services. By the instantiation all RMI HTTP tunneling alternatives are ~28 times slower than RMI and ~31 times slower than web services.

7.2 Overhead Analysis

To be able to understand the reasons for the differences in performance between the described scenarios, we have done an overhead analysis. We have used the Borland Optimize It Enterprise Suite 6 profiler and the Perianal tool [19].

To be able to understand the reasons for the differences in performance between the described scenarios, we have done an overhead analysis. We have used the Borland Optimize It Enterprise Suite 6 profiler and the Perianal tool [19]. We have profiled the test scenarios and identified the methods and packages in which the majority of time was spent. Then we have calculated the percentage of time and compared the methods between the scenarios. The results are shown in Table.

Table 2: Overhead analysis with relative times spent in packages and methods for different scenarios

Package / Method (in %)	RMI	HTTP -to-port	HTTP -to-servlet	Web Services
java.net.socketOutputStream.socketWrite	29,23	37,76	69,84	16,38
sun.rmi.transport	24,16	<0,10	<0,10	<0,10
com.sun.xml.rpc.sp	<0,10	<0,10	<0,10	14,37
com.sun.xml.rpc.encoding.soap	<0,10	<0,10	<0,10	12,79
java.net.PlainSocketImpl.doConnect	<0,10	17,85	0,68	<0,10
sun.net.www	<0,10	14,63	7,10	4,73
java.io.ObjectInputStream	5,05	<0,10	<0,10	<0,10
sun.rmi.server	3,82	<0,10	<0,10	<0,10
java.io.ObjectOutputStream	5,63	<0,10	1,10	<0,10
java.lang.SecurityManager	<0,10	2,36	<0,10	<0,10
java.net.PlainSocketImpl	<0,10	2,66	0,98	<0,10
sun.nio.cs	<0,10	1,07	<0,10	1,19

8. Conclusion

In this paper we have presented Web services, an emerging technology for the Web, The web service overview and the various security issues occurred in the implementation of the xml encryption of the messages. The security of web services is an important aspect and hence a security algorithm is required to implement in web services for key generation and encryption decryption of the messages. Security is important for any distributed computing environment. Recent advancements in XML encoding with W3C EXI along with industry specific formats have shown promising results in minimizing payload overhead and parsing complexity Further the security is explained in the way of handling web services security with its specifications. In the future work, the discussion can be about authentication and authorization of users, securing the data of users, and tracking the user activity.

The security algorithm described in this paper will be used together in combination for key generation and Encryption

decryption of the messages which will provide strong security in web services.

References

[1] ZACH Shelby, Sensinode Ltd, "The Internet of Thinks Embedded Web Services ",IEEE Wireless Communications, vol. 5, no. 2,pp. 52-57,Dec. 2010.
[2] Zibin Zheng, Member, IEEE, Yilei Zhang, Student Member, IEEE, and Michael R. Lyu, Fellow, IEEE,"Investigating Qosf Real-World Web Services", IEEE Transaction On Services Computing, vol. 7, no. 1, pp. 32-39, Jan/Mar 2014.
[3] PhilippLeitner, Florian Rosenberg, SchahramDustdar ,"DAIOS: Efficient DynamicWeb Services Invocation", IEEE Internet Computing, vol. 3, no. 2, May/June 2009.
[4] QuanZ.Sheng, ZakariaMaamar, HamdiYahyaoui, "Separating Operational and Control Behaviors" IEEE Internet Computing, vol. 2, no. 3, pp. 68-76, May/June 2010.
[5] Hongkun Zhao, WeiyiMeng , "Robust Web Services in Heterogeneous Military Networks " IEEE Communication Magazine, vol. 2, no. 4, pp. 78-83, October 2010.
[6] Spiros Koulouzis, KonstantinosA.Karasavvas, "EnablingWeb Services to Consume and Produce Large Datasets " IEEE Internet Computing, vol. 3, no. 4, pp. 52-60, Jan/Feb 2012.
[7] Frank T.Johnson,TrudeHafsqe, Anders Eggen, "Web Services Discovery across Heterogeneous Military Networks, " IEEE Communication Magazine, vol. 1, no. 3, pp. 84-90, October 2012.
[8] Sylvain Halle´, Member, IEEE, and Roger Villemaire, Member, IEEE Computer Society, IEEE,"Runtime Enforcement of Web Service Message Contracts with Data" IEEE Transaction on Services Computing, vol. 5, no. 2, pp. 192-206, Apr/June2012.
[9] Zaki Malik, AthmanBouguettaya,"Reputation Bootstrapping for Trust Establishment among Web Services", IEEE Internet Computing, vol. 3, no. 4, pp. 40-47, Jan/Feb 2009.
[10] Hassan Artail And HaidarSafabooz Allen Hamilton"A Server-In-The-Middle Approach For Enabling Mobile Devices To Discover And Invoke Web Service Methods On Demand", IEEE Wireless Communications, Vol. 6, No.2 October, 2010.
[11] ZakariaMaamar , Hakim Hacid, Michael N. Huhns,"Why Web Services Need Social Networks" IEEE Internet Computing, vol. 3, no. 4, pp. 90-94, Mar/Apr 2011.
[12] Sitaraman Lakshminarayanan, Member, IEEE,"Interoperable Security Standards For Web Services" IEEE Transaction On Services Computing, vol. 2, no. 3, pp. 42-47, Sep/Oct 2010.
[13] Moo Nam Ko, Gorrell P. Cheek, and Mohamed Shehab "Social – Networks Connect Services" IEEE Transaction On Services Computing, vol. 3, no. 1, pp. 37-43, Aug 2010.
[14] Gibson Lam and David Rossiter,"A Web Service Framework Supporting Multimedia Streaming",IEEEransaction On Services Computing, vol. 6, no. 3, pp. 400-413, Jul/Sep 2013.
[15] FatnaBelqasmi, Jagdeep Singh, SuhibYounisBaniMelhem, "SOAP-Based vs RESTful Web Services", IEEE Internet Computing, vol. 1, no. 3, pp. 54-63, Jul/Aug 2012.
[16] Rui Wang, Shuo Chen, XiaoFeng Wang, "Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services", IEEE Symposium on Security and Privacy, vol. 2, no. 5, pp. 365-379 Jan 2012.
[17] Peisheng Zhao, LipingDi,WeiguoHan, " Building a Web-Services Based Geospatial Online Analysis System", IEEE Journal Of Selected Topics In Applied Earth Observations And Remote Sensing, vol. 5, no. 6, pp. 1780-1792, Dec 2012.
[18] HalvardSkogsrud, Hamid R. MotahariFabioCasati, "Modeling Trust Negotiation For Web Services ", IEEE Transaction On Services Computing, vol. 6, no. 4, pp. 54-69, Feb 2009.
[19] Salekul Islam, Dhaka, Bangladesh Jean-Charles Grégoire," Security Issues in OnlineSocialNetworks " , IEEE Internet Computing, vol. 3, no. 3, pp. 56-63, Jul/Aug 2011.
[20] Joe M. Tekli, Ernesto Damiani ," SOAP Processing Performance and Enhancement " , IEEE Transaction On Services Computing, vol. 5, no. 3, pp. 387-403, Jul/Sep 2012.
[21] Matjaz B. Juric, BostjanKezmah, MarjanHericko, Ivan Rozman, Ivan Vezocnik University of Maribor, FERI, Institute of Informatics, Smetanova 17, SI-2000 Maribor Razvojni center IRC Celje, Ulica XIV. divizije 14, SI-3000 Celje "Java RMI, RMI Tunneling and Web Services Comparison and Performance Analysis " ACM SIGPLAN Notices Vol.39(5), May 2004.
[22] 1st PrachiLabhane, 2nd Prof. Khushboo Saxena,1st Department of Information Technology, Technocrats Institute of Technology, Bhopal, India 2nd Department of Information Technology, Technocrats Institute of Technology, Bhopal, India "Review Paper On Web Service Security" Journal of Advanced Computing and Communication Technologies Volume No.2 Issue No. 1,Februry 2014.

Author Profile

R. Krishnamoorthy received the B.E. degree in Computer science and Engineering from Annai Teresa College of engineering in 2011. He is currently doing his M.E Computer science and Engineering in Nandha engineering college, Erode, India.