Analysis of Security and Performance Aspects in Service-Oriented Architectures

Douglas Rodrigues, Julio C. Estrella, Kalinka R. L. J. C. Branco
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos - SP, Brazil, 13560-970
{douglasr, jcezar, kalinka}@icmc.usp.br

Abstract

This paper presents a study on the use of security standards in the context of service-oriented architectures with the goal of testing which are more costly in terms of performance in building secure Web services. Security policies applied to web services are evaluated through application development, testing and performance evaluation. The results of this preliminary study will be used in the near future to create service level agreements in an SOA with a focus on quality of service called WSARCH (Web Services Architecture). The current prototype of this architecture already allows studies to accurately verify which security techniques have low impact in relation to performance.

Keywords: *Service-Oriented Architecture, Web Services, Security, Performance, Encryption, Digital Signature.*

1. Introduction

The service-oriented paradigm has allowed applications that can be built through a network of collaboration which crosses the boundaries of universities and organizations. The basic idea of Service-Oriented Architecture (SOA) has received significant attention and concern from the community of software design and development. As a result of this attention, there has been a proliferation of many conflicting definitions of SOA. Thus, various types of service-oriented architectures have emerged, and among them, Web services have been the most commonly used ones [6].

Web services enable interoperability of applications due to a series of standards that have been created based on standard XML (eXtensible Markup Language). However, Web services do not have predefined clients and therefore must be adaptable to different contexts and are, somehow, a type of client/server system especially structured to make the best use of Web standards.

The advantage of using the paradigm of SOA is interoperability achieved by the use of standard XML, which allows not only communication of conventional usage in the Web, but also communication between devices ranging from a small sensor to a sophisticated domestic machine, commercial or industrial.

Interoperability is fundamental to the integration of Web-based applications and the Internet. This interoperability is achieved through the use of XML-based standards such as SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) and UDDI (Universal Description Discovery and Integration). The SOAP protocol is used for data transfer between services, while WSDL defines an XML schema to

describe the services available and UDDI specification defines how to publish and discover information about a specific service in a directory or registry services [10].

This interoperability allows the integration of applications. However, it does not guarantee the integrity of the information that travels over the Internet, which is often confidential. Thus, the security of information is a necessity, especially with Web services, since their business flows, processes and internal architectures are exposed.

To ensure security in this environment, new security mechanisms must be considered, such as [16]: (i) WS-Security, a standard of OASIS (Organization for the Advancement of Structured Information Standards) in order to SOAP messaging security and providing integrity and confidentiality [14]. (ii) SAML (Security Assertion Markup Language), another OASIS standard based on XML for exchanging security information. (iii) WS-BPEL (Web Services Business Process Execution Language), or simply BPEL is an XML-based language that is used to coordinate Web services in a single business process. BPEL is a combination of languages defined by IBM and Microsoft that is now part of OASIS standardization effort.

Thus, what is proposed in this paper is to study and identify through the implementation of real applications, where security specifications, or any combination of them, can ensure an adequate level of security end-to-end in the context of the messages exchanged between Web services. This includes the definition and evaluation of security policies by making use of traditional security techniques which are already incorporated in them, cryptographic algorithms and digital signatures.

The remainder of this paper is organized as follows. Section 2 describes the importance and challenges of security in Web services. Section 3 presents a performance evaluation in relation to the implementation of applications. Section 4 discusses the results and statistical analysis. Section 5 describes the WSARCH (Web Services Architecture). And finally, in Section 6, the conclusions are presented.

2. Security in Web services

Security is crucial when Web services exchange business data. There may be legal or financial consequences if data is intercepted by others or if fraudulent data are accepted as valid [18]. For Web services are widely adopted it is essential that their use be safe, since no company wants to risk exposing their applications and business workflows without first ensuring that you will not have losses. For this reason, organizations like the W3C, OASIS and WS-I are proposing specifications in order to make these services more secure. These specifications, along with the security specifications for XML, allow security requirements are guaranteed in Web services.

The challenge inherent in this technology is built-in user information security within a SOAP message [15]. In the same way, depending on the infrastructure of the distributed system, the SOAP messages can be routed between different Web services to meet the requests of a client. This implies ensuring that the information contained in the message cannot be accessed by Web services intermediaries, but only by the Web service and the client involved in the request.

In the transport layer the security control can be reflected in protocols and security mechanisms already in place. In this context fits the SSL (Secure Socket Layer) with use of digital certificates and cryptographic keys. However, the problem in SSL is not

to provide an end-to-end security, and such security is necessary in an environment of Web services, since SOAP messages can travel by several Web services intermediaries before reaching the final recipient. Therefore, if encryption is used only at the transport layer, the information will be revealed to Web services intermediaries through which the messages pass.

As mentioned earlier, the SOAP protocol does not implement any security mechanism, but it only uses existing mechanisms in the network infrastructure responsible for these functions. SSL would be a solution, though it does not provide end-to-end security and data integrity, and it becomes not applicable in all situations where it is necessary a secure Web service. Thus, new specifications and security mechanisms have been proposed, especially the XML Signature, XML Encryption and WS-Security.

The XML Signature is a W3C standard [20] that specifies a process for generating and validating digital signatures expressed in XML, ensuring the integrity and authenticity not only in XML documents, but also any other type of digital document [12].

An important property of XML Signature is the ability to sign only specific portions of the XML document [22]. This property is useful because it allows an XML document has additions or changes to information in other parts of it during his lifetime, without invalidating the signed part. In this case, if the signature was performed on the entire XML document, any modification of the data invalidates the same.

Figure 1 shows the structure of the XML Signature specification, where the line 1 indicates a root element called *Signature*, which has three children: *SignedInfo*, *KeyInfo* and *SignatureValue*. The *SignedInfo* in line 2 characterizes the details of the signature process, represented by the elements *CanonicalizationMethod* in line 3, which indicates the algorithm used in the normalization of the document (line 4) and *SignatureMethod* in line 5, which tells the algorithm used in the signature, in this case the combination of the public key algorithm RSA with the hash function SHA-1 (line 6). The *Reference* element in line 7 contains, in turn, a reference to data that have been signed and other information, such as the algorithm used to generate the message digest, in this case SHA-1 (line 10), and its resulting value (line 11). The *SignatureValue* element in line 14 represents the value of the signature and finally the *KeyInfo* element in line 15 provides information about the key used in signature verification.

Integrity checking of the XML document is performed with the receiver calculating locally the message digest and comparing it with the message digest contained in the message. If the two are equal, then the signature is considered valid, thus ensuring the integrity of the document.

Standardized by W3C, XML Encryption [19] defines a way to encrypt data in a structured and represent the result as an XML document, ensuring the confidentiality requirement in it. XML Encryption is used to provide end-to-end security for applications that need to exchange data in XML format in a secure way, without concern that they may have their contents revealed and misused by third parties [17].

```
01.  <ds:Signature>
02.    <ds:SignedInfo>
03.      <ds:CanonicalizationMethod
04.          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
05.      <ds:SignatureMethod
06.          Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
07.      <ds:Reference URI="...">
08.        <ds:Transforms .../>
09.        <ds:DigestMethod
10.            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
11.        <ds:DigestValue>pkKoUrEWaYhQhJKpfx9...</ds:DigestValue>
12.      </ds:Reference>
13.    </ds:SignedInfo>
14.    <ds:SignatureValue>nnzElamjClN aMA0...</ds:SignatureValue>
15.    <ds:KeyInfo>...</ds:KeyInfo>
16.  </ds:Signature>
```

Figure 1. XML Signature

The XML Encryption aims to provide a feature set that is not provided by SSL/TLS. Thus, XML Encryption allows encryption of data at different levels of granularity, i.e., allows select pieces of data to encrypt. This feature is useful when you choose to encrypt only one specific element of an XML document, allowing the rest of it remains with its original content.

Unlike SSL, which encrypts entire groups of data to transport them, XML Encryption performs encryption only in data that really need security. This is advantageous because encryption consumes much time and computing resources, and therefore it must be used wisely [13].

In Figure 2 is illustrated a basic structure used by XML Encryption to represent the encrypted data on a single element. In line 1 there is the root element *EncriptedData*, which has three child elements and stores information such as namespace (line 2) and indicates that the encryption process is applied only on an element (line 3). Line 4 is the *EncryptionMethod* element, first son of *EncriptedData*, which identifies the algorithm used for encryption, in this case 3DES (line 5). Already in line 6 is the second son, *KeyInfo*, which stores information about the key. And finally, in line 9 is the third and last child, *CipherData*, which contains the value resulting from encryption of the element, storing the same in the *CipherValue* element (line 10).

The decryption process is performed by obtaining the cipher text contained in the *CipherValue* element (line 10) and verifying the algorithm (line 5) contained in the *EncryptionMethod* element, information about the key in the *KeyInfo* element (line 6), and which element was encrypted (line 3). With those information then is possible perform the decryption.

The WS-Security was first proposed by IBM and Microsoft, and currently being standardized by OASIS [14], with the aim of proposing a set of extensions to the SOAP messages in order to implement secure Web services. Thus, the WS-Security defines methods for embedding security into SOAP messages, for example, credential exchange, message integrity and confidentiality [9].

```
01. <xenc:EncryptedData
02.     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
03.     Type="http://www.w3.org/2001/04/xmlenc#Element">
04.   <xenc:EncryptionMethod
05.       Algorithm="http://www.w3.org/2001/04/xmlenc#3des-cbc"/>
06.   <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
07.     <ds:KeyName>Key</ds:KeyName>
08.   </ds:KeyInfo>
09.   <xenc:CipherData>
10.     <xenc:CipherValue>ni320jas2...</xenc:CipherValue>
11.   </xenc:CipherData>
12. </xenc:EncryptedData
```

Figure 2. XML Encryption

The WS-Security has the purpose of ensuring safety end-to-end message level with regard to three main points [5]:

- **Confidentiality of the message:** the SOAP message can be fully or partially encrypted using XML Encryption specification, and it must contain information relating to the encryption performed.
- **Message integrity:** the SOAP message can be signed using XML Signature and the same shall contain the information related to your signature.
- **Security credentials:** security credentials with authentication information may be included in the SOAP message. These credentials are also known as access identification, authentication, or security tokens.

In Figure 3 is illustrated a SOAP message with WS-Security and *UsernameToken* security credential, where the security element of the header is called *Security* (line 3) and the same is used to carry security-related information. Within this element is *UsernameToken* element (line 4), which carries an identity through the elements *Username* (line 5) and *Password* (line 8), both encrypted to ensure confidentiality.

```
01. <soapenv:Envelope ...>
02.   <soapenv:Header>
03.     <wsse:Security ...>
04.       <wsse:UsernameToken wsu:Id="1">
05.         <wsse:Username>
06.           <xenc:EncryptedData>9CgAwIBAg...</xenc:EncryptedData>
07.         </wsse:Username>
08.         <wsse:Password>
09.           <xenc:EncryptedData>tJZc0...</xenc:EncryptedData>
10.         </wsse:Password>
11.       </wsse:UsernameToken>
12.     </wsse:Security>
13.   </soapenv:Header>
14.   <soapenv:Body>
15.   ...
16.   </soapenv:Body>
17. </soapenv:Envelope>
```

Figure 3. WS-Security with UsernameToken

One of the major advantages brought by the WS-Security is the organization of the SOAP header, which is essential for the establishment of a standard that turns the

security of Web services in a unified manner to securely transfer information over the Internet.

While the WS-Security allows the integration of Web services security, performance should also be considered. An overhead can occur due to the extra CPU time for processing information related to WS-Security, and more time to carry SOAP messages on the network is increased because of additional content to the WS-Security [21].

Through the implementation of XML Encryption and XML Signature [11] associated with other forms of security, WS-Security makes the sensitive parts of the message confidential to intermediaries and ensures the integrity of the message while it is on communication channel.

It should be noticed that one of the key benefits of SOAP, when compared to other forms of data exchange, is that it allows modular extensions. Since the release of the SOAP protocol, the extensions have focused on security, which resulted in the standardization of WS-Security and related technologies that allow security to be appropriately set for each service [18].

3. Performance evaluation

In order to evaluate the overhead of messaging, five services that perform the same operation, i.e., given two integers and return the sum of them have been implemented. The idea of a simple implementation of Web services is that this way only the response time would be measured, with no significant additional time spent in processing the business logic.

What differentiates one service from another is the security policy established. Thus, it was possible to obtain the average response times with and without the use of security policies. The five implemented services are:

- ***NonSecureService***: service without security policy.
- ***SecureServiceSSL***: service that uses SSL to exchange messages via HTTPS, along with the *UsernameToken* security policy, i.e., requires authentication with user and password.
- ***SecureServiceCrypto***: service with security policy for encryption.
- ***SecureServiceSign***: service with security policy to include digital signature and timestamp, i.e., a record date and time (useful to prevent attacks like replay on a server).
- ***SecureServiceSignCrypto***: service with security policy for digital signature followed by encryption, besides adding the timestamp.

The security policies used in these services make use of cryptographic algorithms already implemented and listed below, details of which can be obtained by the specifications of XML Encryption [19] and XML Signature [20]: RSA (Rivest, Shamir, Adleman), using a pair of public-private 1024-bit encryption for secret key, 3DES (Triple Data Encryption Standard), using 192-bit key to encrypt the entire message body, SHA-1 (Secure Hash Algorithm), resulting in a brief message of 160 bits, followed by the RSA, again with a pair of 1024-bit keys, to sign the entire message

body. For each of the services already described, there is the implementation of a respective client that complies with the policy required which:

- ***NonSecureServiceClient***: client that accesses the service without the security policy.
- ***SecureServiceSSLClient***: client that accesses the service via HTTPS using SSL and *UsernameToken* policy.
- ***SecureServiceCryptoClient***: client that accesses the service in accordance with the security policy for encryption.
- ***SecureServiceSignClient***: client that accesses the service in accordance with the policy of security for digital signature.
- ***SecureServiceSignCryptoClient***: client that accesses the service in accordance with the security policy for the signature followed by encryption.

3.1. Test environment

Both clients and services were implemented in the Java programming language. Services were published in the provider of Web services using the SOAP processing engine Apache Axis2 1.4.1 [2] together with the application server Apache Tomcat [3]. Clients and services also used the Rampart 1.4 module [1], which is an implementation of WS-Security specification developed by Apache.

The physical environment used to perform the experiments is shown in Figure 1. In this environment, clients and services were performed on different machines, but with identical configurations: Intel Core 2 Quad Q6600 2.4 GHz, 2 GB RAM, 120 GB HD and Gigabit Ethernet.

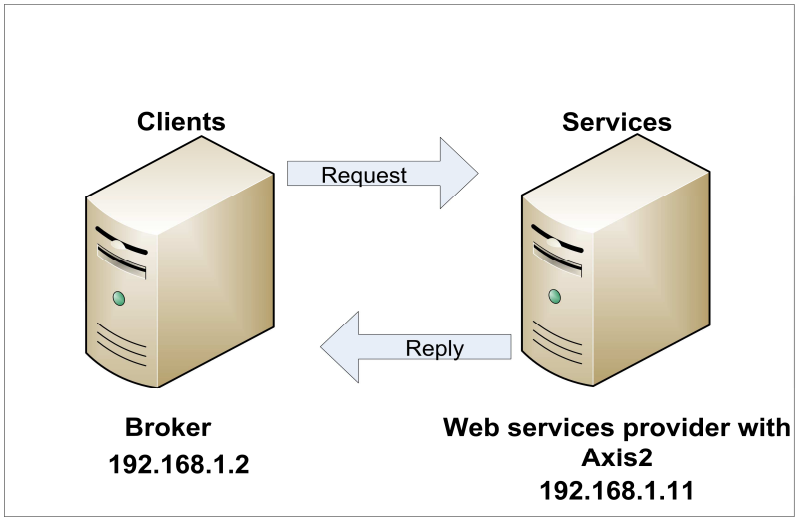

Figure 4. Case study environment

3.2. Planning of experiments

The experiments were designed with three factors and two levels each: encryption, with and without policy for encryption; digital signature, with and without policy for

digital signature; and number of clients: 5 and 10 clients accessing the service concurrently.

Eight experiments were performed, and they are defined in Table 1. Each experiment was repeated thirty times, which is an amount that is consistent with what is suggested in the book based on performance evaluation of computer systems [8], resulting in a total of 240 plays (8 experiments x 30 repetitions).

The variable response of the experiments is the response time, i.e., the total time spent in the request and response on the client machine.

Table 1. Experiments definition

Exp	Type	Number of Clients
1	No Security	5
2	Encryption	5
3	Digital Signature	5
4	Digital Signature and Encryption	5
5	No Security	10
6	Encryption	10
7	Digital Signature	10
8	Digital Signature and Encryption	10

4. Results obtained and analysis

After the execution of experiments and their results, a statistical analysis was required. This analysis is important to summarize the data as appropriate using statistical techniques. Thus, the following subsections are intended to present the results through an analysis that considers the calculation of averages, standard deviations and confidence intervals.

4.1. Response time

Statistical analysis for the response time is illustrated in the graphic in Figure 5, which presents a comparison of response time between 5 and 10 concurrent clients, and in addition all data are presented in Tables 2 and 3. Table 2 refers to the response time for 5 concurrent clients, while Table 3 refers to the response time for 10 concurrent clients. The study with 5 and 10 clients is justified by the fact that the target application of this work (transfer of digitalized fingerprints) does not exceed the total of 10 concurrent clients. For generalization purposes other tests with more clients will be executed, but this case focuses on the scope of work.

The tables show the average response times (in milliseconds - ms), standard deviation and confidence interval of 95% for each type of security policy. Meanwhile, the graphics show the average response times and confidence intervals also for each type of policy. In both cases, the confidence interval remained at a low value, indicating that the results are reliable.

Table 2. Response time for 5 concurrent clients

Type	Resp. Time	Std. Deviation	Conf. Interval
No Security	283.68	38.02154017	13.6055834
Enc	2861.31	77.99662214	27.91021991
Sign	1547.76	62.12568795	22.23098341
Sign and Enc	3094.20	73.42715276	26.27508634

It is possible to notice that with 5 concurrent clients, the lowest response time belongs, of course, to the Web service without security (283.68 ms), followed by the Web service with digital signature (1547.76 ms) and Web service with encryption (2861.31 ms), and finally, the largest response time belongs to the Web service with digital signature followed by encryption (3094.20 ms).

Table 3. Response time for 10 concurrent clients

Type	Resp. Time	Std. Deviation	Conf. Interval
No Security	482.59	36.04660154	12.89887368
Enc	5877.78	138.3788438	49.51732339
Sign	3100.42	90.63035452	32.43105991
Sign and Enc	6038.10	129.2029255	46.23382355

Note that with the increase of 5 to 10 clients, where there is a similar behavior, with 10 concurrent clients, the lowest response time belongs to the Web service without security (482.59 ms), followed by the Web service with digital signatures (3100.42 ms) and Web service with encryption (5877.78 ms), and finally, the largest response time belongs to the Web service with digital signature followed by the one with encryption (6038.10 ms).

In both cases, the response time of Web service with encryption is greater than the response time of Web service with digital signature. This can be explained by the fact that encryption is calculated on the entire body of the message, while the signature of the message is equivalent to the computation of a hash function on the message, followed by a calculation of encryption with the private key of the sender about the contents of message resulting from the hash function. Although the calculation of the digital signature looks longer, which really affects is the amount of encrypted data, as in the case of signature, as explained above, only the message digest is encrypted; while in the case of encryption throughout the body message is encrypted it results in an increased response time.

The graphic in Figure 5 shows that when the number of concurrent clients is increased from 5 to 10, the response time of Web service security ranges from 283.68 to 482.59 milliseconds and increases the response time of 70.12%. For the Web service with encryption response time, with the increase from 5 to 10 concurrent clients, ranging from 2861.31 to 5877.78 milliseconds, there is an increase in response time of 105.42%. For the Web service with digital signatures, with the increase of 5 to 10 concurrent clients, response time increases from 1547.76 to 3100.42 milliseconds, which results in an increase in response time of 100.32%. And finally, the Web service with digital signature followed by the one with encryption when it increases from 5 to

10 concurrent clients, the response time increases from 3094.20 to 6038.10 milliseconds, thus increasing the response time by 95.14%.

It is noteworthy that not only is it necessary to include security, either through encryption, digital signature, or both, but also the number of concurrent clients results in increased response time and consequently the degradation of system performance. Another important observation is that the number of clients competing causes more degradation in performance when there is a security policy applied to the Web service.

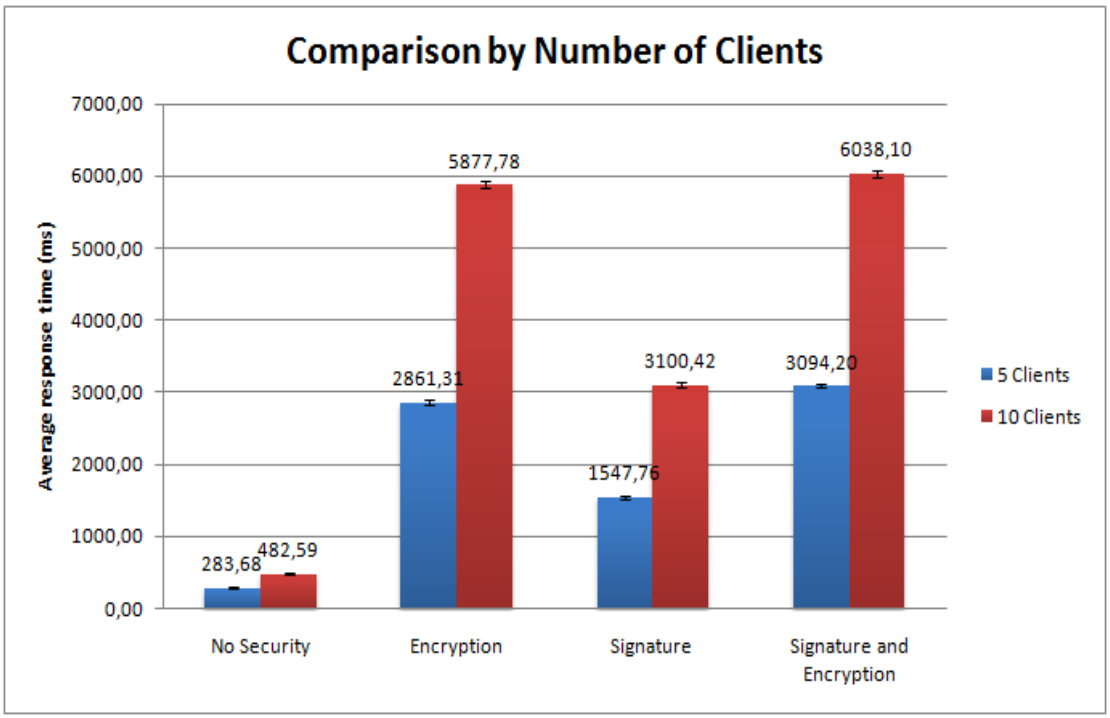

Figure 5. Comparison by number of clients

Another important question to discuss when considering the graphics is why the response time of Web service with digital signature followed by encryption does not match the response time of Web service with digital signature added to the response time of Web service with encryption. This is justified by the fact that in the case of digital signature to the SOAP message is sent with a certificate containing the public key of the issuer in both the request and in response, while in the case of encryption, the client must have the certificate of service send the initial request because the public key of the service is required for encryption. Thus, since the client must have the certificate of service previously, there is no reason to send the certificate of service for the client, being sent only one reference fingerprint (basically a digest of the document) [18].

In the case of digital signature followed by encryption, there is a situation similar to what occurs in the case of cryptography, where the client must previously have the certificate of service, so there is no sending of client certificate to service, and only one reference fingerprint is sent. However, comparing this experiment with digital signatures, it is possible to note that the sent response contains not only the certificate,

but the reference fingerprint, reducing the size of the message, the additional processing costs and hence the time response.

Thus, if the response time of Web service with encryption is X and the response time of Web service using digital signature is Y, the response time T of the Web service using digital signature then encryption will be $T = X + Y - Z$, where Z refers to the time spent on additional processing caused by sending a certificate of service to the client in the case of digital signature, as shown in Figure 6.

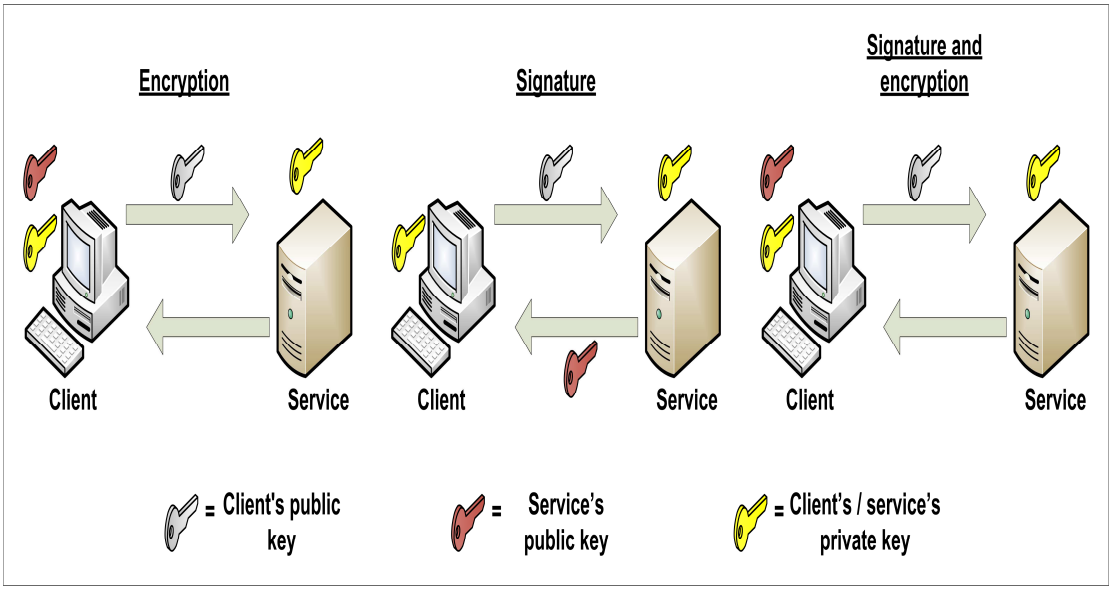

Figure 6. Justification of the response time

4.2. Increment percentage

In the work of [4] there is the definition of a metric which compares the response time of a Web service with and without security policies. Through these response times a value called Increment Percentage (IP), which indicates that the percentage increase in response time of Web service with security policy in relation to the response time of Web service without security is obtained. IP is calculated according to the equation illustrated in Figure 7, where T_{WSS} refers to the response time of Web service with security policy and T_{NonWSS} refers to the response time of Web service without the security policy.

$$IP = \frac{T_{WSS} - T_{NonWSS}}{T_{NonWSS}} \times 100\%$$

Figure 7. Equation for calculation of IP

Statistical analysis for the percentage of increase is in the graphic illustrated in Figure 8, which presents a comparison of percentage increase between 5 and 10 concurrent clients, and additionally by Tables 4 and 5. Table 4 refers to the percentage increase for 5 concurrent clients, while Table 5 refers to the percentage increase to 10 concurrent clients.

The tables show the average response time and increment percentage for each type of security policy in relation to the response time of Web service without security. Since the graphics shows the increment percentage for each type of security policy, in relation to the response time of Web service without security.

Table 4. Increment percentage for 5 concurrent clients

Type	Resp. Time	Increment Percentage
No Security	283.68	-
Enc	2861.31	908.64%
Sign	1547.76	445.60%
Sign and Enc	3094.20	990.74%

It is possible to notice that with 5 concurrent clients, the lowest percentage of increase belongs to the Web service with digital signatures (445.60%), followed by the Web service with encryption (908.64%), and finally, the highest percentage of increment belongs to the Web service with digital signature followed by encryption (990.74%).

Table 5. Increment percentage for 10 concurrent clients

Type	Resp. Time	Increment Percentage
No Security	482.59	-
Enc	5877.78	1117.97%
Sign	3100.42	542.45%
Sign and Enc	6038.10	1151.19%

With 10 clients it is possible to observe a behavior similar to the one with 5 clients, while with 10 concurrent clients, the smallest percentage of increase belongs to the Web service with digital signatures (542.45%), followed by the Web service with encryption (1117.97%), and finally, the largest percentage increase belongs to the Web service with digital signature followed by encryption (1151.19%).

The graphic in Figure 8 shows that when there is an increase in the number of concurrent clients from 5 to 10, the increment percentage in the Web service using encryption is from 908.64% to 1117.97%, resulting in an increase of 23.04%. For the Web service with digital signature, with the increase of 5 to 10 concurrent clients, the increment percentage rose from 445.60% to 542.45%, resulting in an increase of 21.73%. And finally, the Web service with digital signature followed by encryption when it increases from 5 to 10 concurrent clients, the increment percentage ranges from 990.74% to 1151.19%, with an increase of 16.19%.

It is possible to observe through the results that one of the largest overhead is caused by encryption. This leads to the definition of rules for what parts must be effectively and necessarily in an encrypted SOAP message.

4.3. Influence factors

The process of analysis of influence factors is an important step in a performance evaluation.

According to the evaluation presented in Section 3.2, three factors were considered: encryption, digital signature and number of concurrent clients. Regression analysis

allows estimating and predicting a random variable depending on other variables. Such analysis estimated the variable that is called dependent variable, while the variables used to estimate it are called factors. Thus, the analysis using the regression model allows quantifying the influence of each factor in the execution of experiments [8].

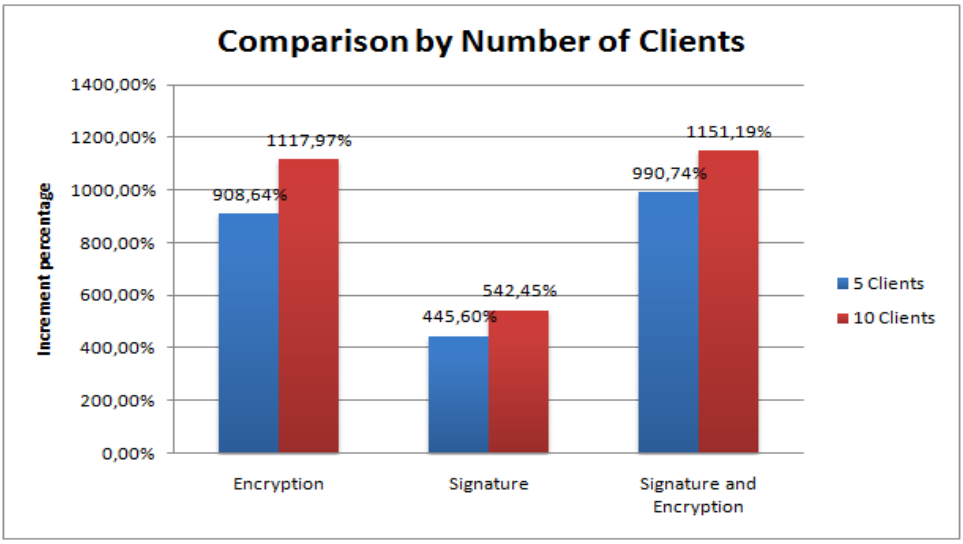

Figure 8. Comparison by number of clients

The graphic shown in Figure 9 shows the percentage of influence of each factor in the experiments. It is possible to notice that the encryption factor (A) has the highest percentage of influence, which is 58%, followed by the factor number of clients (C) with 22% of influence, and finally by the signature (B) with 7% of influence. A combination of encryption factors and number of clients (AC) also has an influence of 7%. Since the combination of encryption factors and signature (AB) results in 4% of influence. Finally, the combinations of signature factors and number of clients (BC) and the encryption factors, signature and number of clients (ABC) have no significant influence, being in the range of 1%.

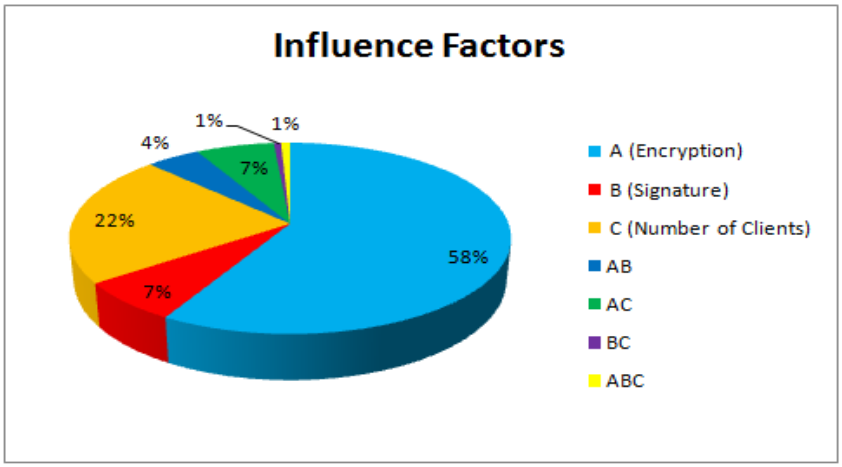

Figure 9. Influence factors

Again, it is possible to notice which the most influential factor is: encryption. Thus, it is evident that the impact and influence caused by each algorithm and the key size

used by the same factor is important for setting and achieving good levels of performance secure applications in an SOA.

4.4. WS-Security vs. SSL

Several security policies using the WS-Security have been presented and its influence factors presented. Thus, it is worth notice that one of the techniques still in use is SSL, which aims to ensure the point-to-point security at the transport layer. For this reason the service *SecureServiceSSL* was added later in order to compare the response time of a service with SSL services using the WS-Security. The security provided by SSL, however, does not guarantee the end-to-end security, despite having a relatively better performance compared to the security policies of the WS-Security, as can be seen in the graphic in Figure 10.

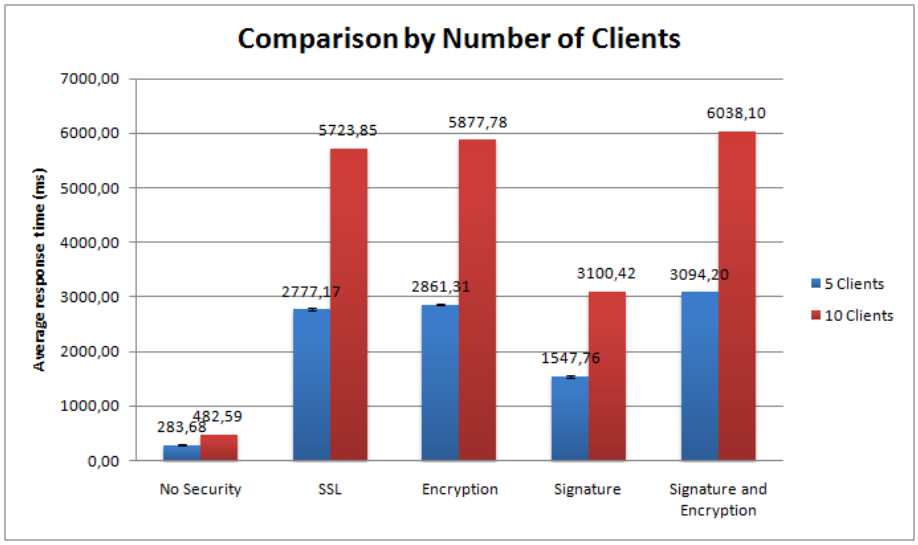

Figure 10. Comparative WS-Security vs. SSL

Keep in mind that the values correspond to experiment with SSL using the same policy *UsernameToken*, which asks username and password on every connection made.

SSL equals in the items authenticity and confidentiality when compared to WS-Security, primarily for point-to-point, where the use of WS-Security can become exaggerated. However, the WS-Security provides data integrity in a unique way when compared to the one provided by SSL, even in a direct connection between client and provider.

5. Application of the preliminary results

Preliminary results comparing the performance and security allows us to reflect on the need for measures to develop secure Web services. The performance and security attributes are antagonistic, and both need to be properly analyzed in a SOA architecture and its components. The next section shows a SOA architecture that supports quality of service designed to address issues related to the performance of Web services [7]. In this sense, the results obtained with the study of this work shall be used to analyze the problems of performance and security in a SOA architecture consisting of clients,

providers, service registry and a service bus (Broker), i.e., in any SOA architecture that support the quality of service. The initial idea is to define Service Level Agreements (SLA) related security and performance.

5.1. WSARCH

The WSARCH (Web Services Architecture) [7] is an architecture which allows accessing Web services using a combination of functional and non-functional aspects of Quality of Service (QoS). These QoS aspects aim at evaluating the performance of Web services in order to achieve QoS in a service-oriented architecture. These QoS attributes were mapped to the components participating in a service-oriented architecture that incorporates quality of service. The architecture provides the monitoring of service providers and the data obtained are used to locate the most appropriated service. A prototype for the WSARCH allows performance evaluation studies being conducted considering different components of the architecture, algorithms, protocols and standards. By now, we want include security attributes in this architecture involving all the components (UDDI, Broker, clients and providers). The WSARCH and its components are presented in Figure 11.

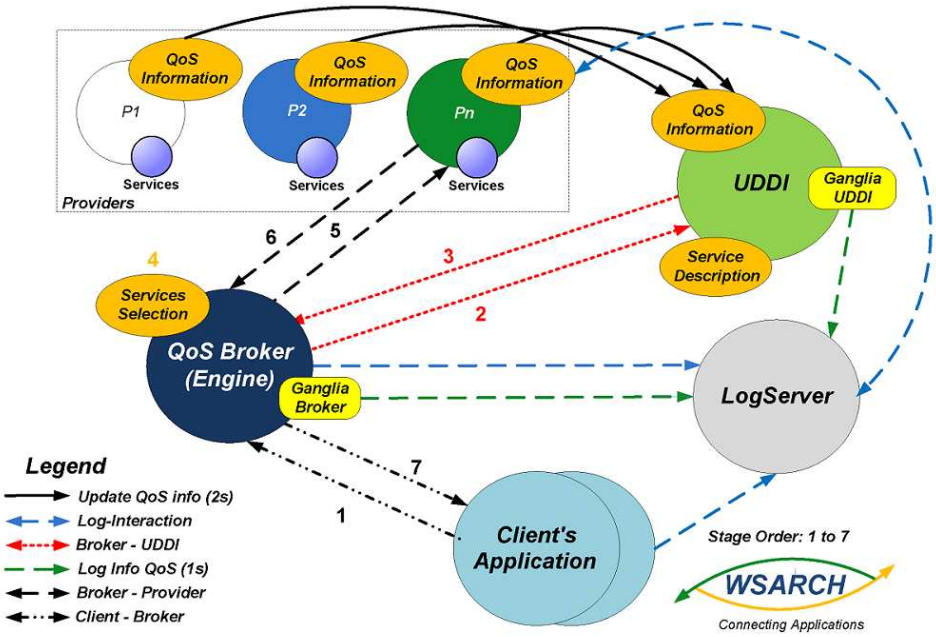

Figure 11. WSARCH

The following steps are necessary for the execution of a request:

1. Client makes request to the Broker, which has updated information from the service provider (load, type of service, client class).
2. Based on the QoS information requested by the client, the Broker performs a search in a service repository in order to find the most appropriate service.

3. Broker gets the specification of appropriate service and information about the QoS of the service provider.
4. With the location of appropriate service in the service repository and information of the service providers (this QoS information is propagated periodically), the Broker chooses the best candidate provider (Services Selection).
5. After selecting the service, the Broker performs the request (invocation of Web Service) to the service provider.
6. After being made the request, the reply is returned directly to the Broker.
7. Finally, the reply is sent to the client that originally requested the Web service.

Other activities occur in parallel with the request of the service client. The QoS information of service providers are updated periodically in the UDDI service registry for each of the registered providers. The information is obtained through the use of the Ganglia Monitoring System [23] running in the service providers and also in the UDDI. Service providers have slave monitors that collect and send information to a master monitor in the UDDI registry, so that the Broker can use it as QoS information (performance index) for selecting the best service provider. Using this performance index the concept of Service Level Agreement (SLA) becomes a key element. Although SLAs represent an additional responsibility for the service provider (it motivates the need of a SLA-enforcement process in the WSARCH infrastructure), they can be seen as containers of the functional and non-functional properties that both parties (service consumer and the service provider) agree by specifying its rights and obligations during the interaction. The performance index of the initial prototype of WSARCH can also be used as a security index for identifying providers which are able in terms of security aspects. As the current stage the prototype does not address security aspects, the results of this paper will contribute with development of methodologies and performance standards for service-oriented architecture with a focus on quality of service.

6. Conclusions

With Web services implemented and the inclusion of their security policies, experiments and data collection were performed for this analysis. Thus, the performance of a Web service without security with other Web services using the WS-Security to add encryption and digital signatures in SOAP messages exchanged in communication have been compared. Furthermore, the results obtained with the WS-Security were compared with results obtained in an experiment where the Web service using the SSL security standard. As could be seen, despite having a relatively lower response time, SSL does not guarantee end-to-end security.

Due to the inherent characteristics of the protocols that make up a service-oriented architecture, security becomes a key item. Thus, studies and performance evaluation of the inclusion of security in this environment are important, since such inclusion causes a considerable reduction in the performance of a service-oriented architecture. The study presented in this paper demonstrates that in addition to encryption factor, the number of concurrent clients requesting a particular service confirms the performance degradation.

But this performance degradation was expected when it is intended to make the service safe, particularly with the use of WS-Security. Therefore, one should carefully plan the balance between performance and security, where the higher the level of security that is intended to implement the services, the lower the performance.

This paper presents a study of the impact of each factor in system performance, demonstrating that security and performance are important factors for SOA applications and the inclusion of one leads to degradation of the other. Further considerations taking into account other factors (size of encryption key, cryptographic algorithm used, among others) are being studied in order to build a solid foundation of experiments that could lead to more accurate inferences about when it is appropriate to raise security with WS-Security in detriment of performance and when security and performance can be used in a balanced way. These foundations will be more discussed by applying the study developed in this paper in service-oriented architecture as the WSARCH and others with similar characteristics.

7. Acknowledgments

The authors acknowledge the support granted by CNPq and FAPESP to the INCT-SEC (National Institute of Science and Technology - Critical Embedded Systems - Brazil), processes 573963/2008-9 and 08/57870-9 and gratefully acknowledge the financial support of CNPq to this project.

8. References

[1] Apache. “Rampart: WS-Security module for Axis2”, 2007. http://ws.apache.org/axis2/modules/rampart/1_2/ security-module.html. Retrieved November 22, 2010.

[2] Apache. “Apache Axis2”, 2009. http://ws.apache.org/axis2/. Retrieved November 22, 2010.

[3] Apache. “Apache Tomcat”, 2010. http://tomcat.apache.org/. Retrieved November 22, 2010.

[4] S. Chen, J. Zic, K. Tang, and D. Levy. “Performance Evaluation and Modeling of Web Services Security”. IEEE International Conference on Web Services, 0:431-438, 2007.

[5] D. C. Chou, and K. Yurov. “Security Development in Web Services Environment”. Computer Standards & Interfaces, v. 27, n. 3, p. 233–240, 2005.

[6] T. Erl. “Service-Oriented Architecture: Concepts, Technology, and Design”. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[7] J. C. Estrella, R. T. Toyohara, B. T. Kuehne, T. C. Tavares, R. C. Santana, M. J. Santana, and S. M. Bruschi. “A Performance Evaluation for a QoS-Aware Service Oriented Architecture”. IEEE Congress on Services, pp. 260-267. 6th World Congress on Services, 2010.

[8] R. K. Jain. “The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling”. Wiley, April 1991.

[9] M. Jensen, N. Gruschka, R. Herkenhoner, and N. Luttenberger. “SOA and Web Services: New Technologies, New Standards - New Attacks”. In: ECOWS ’07: Proceedings of the Fifth European Conference on Web Services, Washington, DC, USA: IEEE Computer Society, 2007, p. 35–44.

[10] N. Josuttis. “SOA in Practice: The Art of Distributed System Design”. O'Reilly Media, Inc., 2007.

[11] T. Knap, and I. Mlýnková. “Towards More Secure Web Services: Pitfalls of Various Approaches to XML Signature Verification Process”. In ICWS '09: Proceedings of the 2009 IEEE International Conference on Web Services, pages 543-550, Washington, DC, USA, 2009. IEEE Computer Society.

[12] M. Mogollon. “Cryptography and Security Services: Mechanisms and Applications”. IGI Global, 2008.

[13] R. Nagappan, R. Skoczylas, and R. P. Sriganesh. “Developing Java Web services”. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[14] OASIS. “Web Services Security (WSS) TC”, 2006.

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss. Retrieved November 22, 2010.

[15] M. O'Neill. "Web Services Security". McGraw-Hill, Inc., New York, NY, USA, 2003.

[16] E. Ort. "Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools", 2005.
http://java.sun.com/developer/technicalArticles/WebServices/soa2/. Retrieved November 22, 2010.

[17] B. Siddiqui. "Exploring XML Encryption, Part 1". IBM Corporation.
http://www.ibm.com/developerworks/xml/library/x-encrypt/. 2002. Retrieved November 22, 2010.

[18] D. Sosnoski. "Java Web Services: Axis2 WS-Security Signing and Encryption", 2009.
http://www.ibm.com/developerworks/java/library/j-jws5/. Retrieved November 22, 2010.

[19] W3C. "XML Encryption Syntax and Processing", 2002.
http://www.w3.org/TR/xmlenc-core/. Retrieved November 22, 2010.

[20] W3C. "XML Signature Syntax and Processing (Second Edition)", 2008.
http://www.w3.org/TR/xmldsig-core/. Retrieved November 22, 2010.

[21] S. S. Yau, Y. Yin, and H. G. An. "An Adaptive Tradeoff Model for Service Performance and Security in Service-Based Systems". In ICWS '09: Proceedings of the 2009 IEEE International Conference on Web Services, pages 287-294, Washington, DC, USA, 2009. IEEE Computer Society.

[22] G. Yue-Sheng, Z. Bao-Jian, and X. Wu. "Research and Realization of Web Services Security Based on XML Signature". International Conference on Networking and Digital Society, v. 2, p. 116–118, 2009.

[23] M. L. Massie, B. N. Chun, and D. E. Culler. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. Parallel Computing, 30(7), July 2004.

Authors

Douglas Rodrigues
M.Sc. in Computer Science and Computational Mathematics at Institute of Mathematics and Computer Science, University of São Paulo - ICMC-USP/São Carlos. B.Sc. in Computer Science at Univem - Marília/SP (2008). His main research topics are: SOA, Web Services, Performance Evaluation, Computer Security, Encryption and Digital Signature.

Julio Cezar Estrella
Ph.D. in Computer Science at Institute of Mathematics and Computer Science from University of Sao Paulo - USP (2010). MSc in Computer Science at Institute of Mathematics and Computer Science from University of São Paulo - USP (2006). BSc in Computer Science at State University of Sao Paulo - Julio de Mesquita Filho – UNESP (2002). He has experience in Computer Science with emphasis in Computer Systems Architecture, acting on the following themes: Service Oriented Architectures, Web Services, Performance Evaluation, Distributed Systems, Computer Networks and Computer Security. He is currently Assistant Professor at Institute of Mathematics and Computer Science - ICMC – USP.

Kalinka Regina Lucas Jaquie Castelo Branco
Assistant Professor of the Institute of Mathematics and Computer Science - ICMC - USP, working in the department of Computer Systems. She has experience in Computer Science, with emphasis on Distributed Computing Systems and Parallel Computer, working mainly in the following areas: distributed systems, computer networks, security, performance evaluation and processes scheduling. She is member of Brazilian Computer Society.