Performance Comparison of Web Services Security: Kerberos Token Profile Against X.509 Token Profile

A. Moralis, V. Pouli, M. Grammatikou, S. Papavassiliou, V. Maglaris
Network Management & Optimal Design Laboratory (NETMODE)
National Technical University of Athens (NTUA)
Athens, Greece
{amoral, vpouli, mary, papavass, maglaris}@netmode.ntua.gr

Abstract

Web Services (WS) Security is the set of standards that provides means for applying security to WS. In this paper we present the performance of the WS Security Kerberos Token profile in contrast to the X.509 Token Profile. The measurements are based on the Apache wss4j library for the X.509 Token Profile and the extensions we have made on the same library in order to support the Kerberos Token Profile. The Kerberos Token profile is based exclusively on symmetric cryptography, whereas the X.509 profile uses Public Key Cryptography for encrypting the symmetric encryption key used for deciphering the message. These differences in the nature of cryptography are reflected and quantified on the measurements we have performed. The performance evaluation and numerical results, demonstrated that Kerberos Token profile has up to 28% packet throughput improvement over the X.509 Token profile, under full CPU load on the server.

Keywords: Web Services; Web Services Security (WSS); Kerberos; X.509; Public Key Infrastructure (PKI)

1. Introduction

Web Services (WS) [13] define a set of platform-independent protocols based on XML, that govern the communication among networked applications, allowing the creation of distributed complex applications from the interoperability of different distributed entities. Web Services use, among others, the Simple Object Access Protocol (SOAP) [7] for encoding messages in a common XML [10] format and utilize mostly HTTP as the transport protocol. The extensive XML format that SOAP poses has side effects on the message overhead and thus in performance, compared to protocols that use binary messages like Common Object Request Broker Architecture (CORBA) as shown in [8]. Additionally, sending each message as a different TCP session along with the XML parsing (serialization-deserialization) are very resource intensive in computation and storage [9]. Performance wise, Web Services are demanding and this can pose a problem on applications that have strict Quality of Service (QoS) requirements. In [23], an extensive comparison of Web services, WS Security, RMI and RMI-SSL, is presented.

Secure authentication is also computationally demanding due to the requirements of encryption-decryption used by the various security protocols. Web Services can be “secured” either in the transport layer using Transport Layer Security (TLS) [12] or at the presentation layer using WS Security [19]. Solutions at the transport layer, although they lack of overheads such as encoding keys and signatures written in ASCII, they present limitations in restricted environments, where the message has to pass through a proxy server that needs to examine the message for routing purposes. On the other hand, in [22], there is a comparison of transport level security (SSL) and presentation level security (WS Security) showing a great performance penalty when WS Security is used. In this paper we address the issue of WS security, by studying, evaluating and quantifying the performance improvements that can be achieved, when adopting Kerberos based solutions that utilize symmetric key cryptography.

WS Security provides authentication at the message level by incorporating widely used authentication mechanisms into the SOAP message header and performing the required encryption functions to the required SOAP elements. Authentication mechanisms include Public Key Cryptography and use of X.509 certificates, username-passwords and Kerberos tickets. Performance of the cryptographic algorithms used with WS Security and the impact of the SOAP processing on these is presented in [21]. The above mechanisms are used for establishing cryptographic keys. Thus, message confidentiality, integrity and non-repudiation can be provided by XML Encryption [15] and XML Signature [16]. These can be applied to different parts of the SOAP message, or alternatively to the body. This flexibility allows the design of complex authentication schemes, where various parts of the SOAP message can be encrypted by various intermediate services and decrypted at the destination. These complex policies cannot be implemented by enforcing the security at the transport layer e.g. sending the message in https.

Within the GRIDCC[1] project [3], we have created a security architecture [4] that is based on Kerberos [5] protocol to distribute symmetric keys. Kerberos is based on the Needham-Schroeder Protocol [20] and offers authentication and single sign-on features, using symmetric key cryptography. The goal of GRIDCC is to build a widely distributed system that is able to remotely control and monitor complex instrumentation. As a control system GRIDCC presented high QoS requirements and as a result all components, including security had to be designed with simplicity and maximize the performance. Therefore, Public Key Infrastructure (PKI) was avoided for the message exchange due to the lower performance of public key cryptography compared to the symmetric key one. The use of symmetric key cryptography was also preferred because no certificates are needed to be configured in the server and this leaded to a simplified server design.

In the framework of the GRIDCC project, all the applications that require QoS are authenticated to the Kerberos system and communicate using Web Services Therefore, for the secure communication between them, WS Security Kerberos Token Profile [14] has been chosen, as it fits to the security architecture, regarding message authentication, encryption and signing. Due to lack of implementation using the Apache Axis [17] and WSS4J [1], we have implemented the WS Security Kerberos Token profile by extending the wss4j library to support the protocol.

In this paper the emphasis is placed on the performance improvement of our implementation of WS Security Kerberos Token Profile. In order to better evaluate, understand and quantify the corresponding performance improvement, it is compared with the one of X.509 Token Profile, under various measurement scenarios. The rest of the paper is organized as follows: In section 2 the Web Service security specifications are summarized, and WS Security X.509 and Kerberos Token profiles are analyzed. In section 3 the testbed and the corresponding measurement methodology that were utilized for the performance evaluation are described, while in section 4 some numerical results and relative discussions are presented. Finally section 5 concludes the paper.

[1] This work was supported in part by the EC GridCC Project (IST 511382).

2. The Web Service Security Specification

WS Security standard (version 1.1 [19]) aims at enabling applications to perform secure SOAP message exchanges. Specifically, it supports: Multiple Security Token Formats, Multiple Trust Domains, Multiple Signature Formats, Multiple Encryption Technologies, End-to-End Message Content Security and not just transport level Security.

The core specification defines an abstract security model to protect (confidentiality and integrity) and authorize the SOAP messages using signatures and security tokens. Message protection is achieved by encryption of the body, or parts of the SOAP message, whereas, integrity and message origin are verified by signatures which can sign either the header or the body or any combination of them. The specification defines how the various security tokens are included in the message, but not how they are acquired by the participating parties. These are implementation details and can vary among different security mechanisms and systems.

All the security related information targeted to a specific recipient is included in the <wss: security> security header block that is attached within the SOAP header. More than one security header blocks can exist, if the message is targeted to more than one recipients. Inside the security header these items can be attached: Username Token, Binary Security Tokens, XML Tokens, EncryptedData Tokens, SecurityTokenReferences, Signatures, XML Encryption Reference List.

When sending a SOAP message protected by WS Security a security header as described earlier needs to be attached. It should contain a BinarySecurityToken or a SecurityTokenReference element. Then depending on the actions chosen, a signature could also be included and the elements that are encrypted are substituted by their ciphertext.

Except from the core specification, there are additional documents that define the WS security using X.509 certificates and Kerberos tokens. These documents are actually extensions of the core specification to meet the specific requirements of each security mechanism.

In the following both standards are summarized.

2.1. WS Security X509 Token profile

The WS Security X509 Token Profile [1] describes the use of a X.509 Certificate [6] for WS Security. In order to support this token profile the standard introduces, as described below, three (3) types of binary security token containing the X.509 certificate and three (3) token references so as to specify all references to X.509 token types in signature or encryption elements that comply with this profile:

Token Types:

- X.509v3: for a X.509 single certificate
- X509PKIPathv1: for an ordered list of X.509 certificates that represent a certificate path.
- PKCS7: for a certificate path with optional Certification Revocation Lists (CRLs) packaged in a PKCS#7 data structure.

Token References:

- Reference to an X.509 Subject Key Identifier: this is a reference to a X.509 certificate by the means of its X.509 SubjectKeyIdentifier.
- Reference to a Security Token: this is a reference to an included in the message preceding X.509 security token.
- Reference to an Issuer and Serial Number: this element specifies a X.509 certificate by the means of a certificate issuer name and serial number.

2.2. WS Security Kerberos Token Profile

Kerberos Token specification [14] was released in version 1.1 of the WS Security Specification. The Kerberos Token specification defines how to encode and attach Kerberos tickets to SOAP messages. Also it specifies how to add signatures and encryption to the SOAP message by using the Kerberos ticket. The Kerberos ticket is included in an AP-REQ packet containing the ticket along with an authenticator to serve thwart replay. This AP-REQ packet is defined by Kerberos as a Kerberos Token. The Kerberos token is attached to the header as a Binary Security token. For encryption and signature this token must be referenced as a security token reference.

When exchanging messages to a service the Kerberos Token should be included in the initial message but for performance reasons, the subsequent exchanged messages should include a token reference, the KeyIdentifier reference. The value of the reference should be the encoded (base64) hash value of the Kerberos Token (AP_REQ packet).

The Kerberos Token can be used either to sign or/and encrypt the SOAP message. When the Kerberos ticket is referenced as a signature key, the specification defines that the signature algorithm must be a Hashed Message Authentication Code (HMAC). Respectively, when the Kerberos ticket is referenced as an encryption key, the encryption algorithm must be a symmetric one. The key used for signature or encryption could be constructed from the Kerberos subkey, or the session key or derived from a mechanism agreed to by the communication parties.

3. Testbed and Scenarios

3.1 Testbed Description and Metrics

The testbed consists of a server and 6 clients. The server is a Linux Fedora Core 5 running on an AMD64 X2 4.400 (2200Mhz), 2GB of RAM, 2x120GB hard disks in raid 1 configuration. The operating system was running on the single (not smp) kernel. The client software is running on various machines that are Windows XP and Linux.

The client implementation is written in Java. The clients have the ability to send messages with a specified rate and payload and encrypt the body of the SOAP message using AES128. The server code is also written in java and for the Web Services implementation we used Jakarta Tomcat 5.28 and Apache Axis 1.3. The security code of the server is written as an Axis handler. Handlers can be deployed in front of a service and perform processing to the SOAP message. For the X.509 we have used the included Axis Handler in the WSS4J. For the Kerberos token profile we have written our own handler along with the required mechanisms by extending the WSS4J API in order to support the WSS Kerberos Token Profile. In our implementation we will test the performance of the two WS Security mechanisms by only encrypting the messages sent by the clients. Thus both handlers have been configured to decrypt the incoming requests. Only the receiving direction to the server has been protected with encryption. The reply from the server is in clear text. All the cryptographic functions are performed by the BouncyCastle [18] JCE provider.

For testing purposes, we have chosen an echo web service, which accepts a string and replies sending that string. This service is of low complexity, as the main objective is to picture the performance of the WS security by minimizing the load from other factors, like the ones from the service.

The measurements which are performed by the handler are the following:

- Request Total: Total time of the WS Service security processing (at the handler). This time is actually the addition of all the times shown below.
- Request Preparation: Time that the handler needs to deserialize the message and construct the Java objects from the SOAP message (SOAP processing).
- Request Processing: Time in which all the security tokens are parsed (Kerberos AP REQ, X509), the encryption keys are derived from the token references and the decryption takes place.

- Request to Axis: Time required to reconstruct the message by removing the security elements and by substituting the ciphertext with cleartext.
- Verify header, cert, timestamp: Final verification of the header, cert and timestamp (if present).

It is mentioned that the above times that the handler measures are the times of the WS Security required in order to process the SOAP message. These times do not include other times like the processing of the actual service and the serialization of the reply.

3.2 Methodology

In our testing we compare the performance of the WS Security when using Kerberos tickets against X.509 certificates in order to encrypt the SOAP messages.

The methodology adopted in each case is the following:

X.509 Certificates

The client encrypts the body of the SOAP message using AES128, creates a WS security header and attaches into it first a reference to the issuer and to the serial number of a X.509 certificate (service certificate) and second the symmetric key encrypted using RSA15 with the public key of the referenced certificate.

At the server side, the WS Security handler retrieves from its keystore, where it has all certificates stored, the service's private key and using it, retrieves the symmetric key. Then it decrypts the body and passes the decrypted message stripped from the security header to the actual service.

Kerberos tickets

The client logins to the AS of the Kerberos system and requests from the TGS a ticket for a service that he/she wants to access. Then it attaches the ticket as a Kerberos Token (AP_REQ packet containing the ticket along with an authenticator) to the security header, and encrypts the body using the session key derived from the Kerberos Token. In the subsequent messages, the client encrypts the body using the session key and attaches a reference to the Kerberos token instead of the whole token. This reference is the SHA1 value of the AP_REQ.

At the server side, when the handler accepts a message, it extracts the service ticket along with the authenticator from the Kerberos Token, derives the session key, stores it in a hash table and decrypts the body with it. In the subsequent messages, it retrieves the session key from the hash table by checking the reference to the Kerberos token that the client has sent.

In principle, the two mechanisms that are compared are not equivalent, regarding their functionality. Kerberos, by using tickets, also authenticates the message, whereas, X.509 mechanism, as tested in our comparison does not. In order to be equivalent the X.509 mechanism ought to add the client's signature. However, this does not impact our objective, since in our comparison we mainly aim at demonstrating the impact of Public Key Cryptography in correlation with Web Services, when used in systems where QoS and high performance play an important role.

3.3. Scenarios

We have tested two basic scenarios. The first one (SC1) involves the 5 clients and the server, where the clients send messages constantly with a specific size. Specifically, every client sends 10000 messages (50000 total). All clients start sending messages at the same time. The measurements are repeated with changing the size of the message payload (string) to 60, 200, 400 and 800 bytes each time. It should be mentioned that the message payload size is before the encryption. The encryption of the body of the SOAP request is done afterwards.

The second scenario (SC2) involves configuring the clients to send messages with a steady rate. The size of the payload of the message is fixed at 60 bytes. Again the body of the SOAP request is encrypted. We start with one client and at each iteration one client is added, until we have 6 clients sending simultaneously.

4. Measurements – Results

In this section, we present some comparative results between the WS Security X.509 Certificate Token profile and the WS Security Kerberos Token profile, for both scenarios described above.

4.1. Scenario 1 (SC1)

SC1 mainly aims at studying the behavior and operation of both systems under heavy load with different body sizes. The CPU utilization that the service process presented during these experiments was over 95%.

Table I. Packet Throughput Comparisons using Kerberos Token Profile against X.509 Token Profile

Packet Size	60	200	400	800
Certificate Handler (Packet/sec)	83.53	81.99	81.39	79.34

Kerberos Handler (Packets/sec)	107.37	104.86	102.87	99.47
Gain %	28.54	27.90	26.40	25.37

Table I shows the packet throughput achieved by Kerberos Token and X.509 Token respectively, as well as the corresponding percentage gain of using Kerberos Token (instead of X.509 Token), for different packet sizes. As observed in Figure 1 t there is a relatively small degradation in performance of both solutions, as the size of the SOAP body increases.

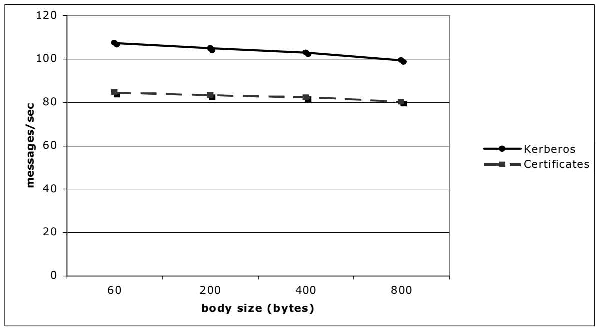

Figure 1. Average packet rate for different body sizes

Figure 2 reveals the average Request Total time each handler (Kerberos and X509 Certificates) needs for the complete security processing (processing the security header, decrypting the message, preparing and verifying the SOAP call for Axis).). Based on the results of this figure we observe that the processing time of the Kerberos tickets is significantly less than the one of the X509 certificates.

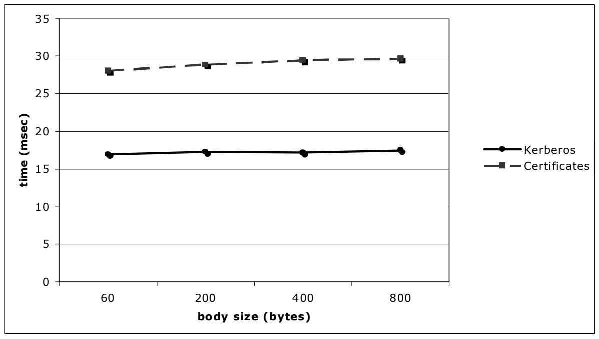

Figure 2. Average Request Total Time (in msec)

Figure 3 shows the average Request Processing time that the handler needs for the processing of a single request in order to find the key and decrypt the body, for different body sizes. Both Kerberos and Certificates decrypt the SOAP body using AES128.

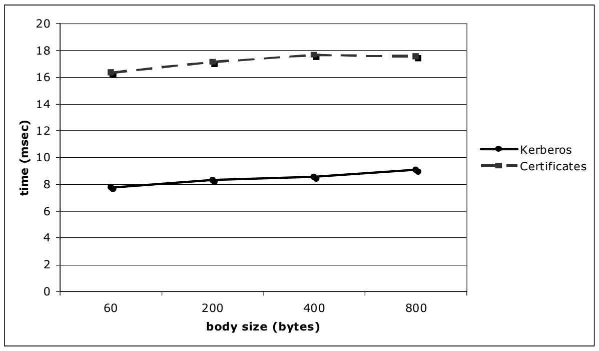

Figure 3. Average Request Processing Time of security processing for different packet sizes

The significant differences observed in these results are mainly due to the way that the handlers derive the key. X.509 mechanism, extracts the key included in the message using RSA15 (Public Key algorithm). The Kerberos just uses the session key already acquired in the first message from the Kerberos token. Consequently Kerberos lowers the processing time by approximately 50% when compared to the case of Certificates. This difference is justified by the fact that Kerberos handler is using symmetric cryptography exclusively, whereas the Certificate handler is mixing Public Key and Symmetric Cryptography.

The values of the various components of the processing times of the Kerberos handler are presented in Figure 4, for different body sizes.

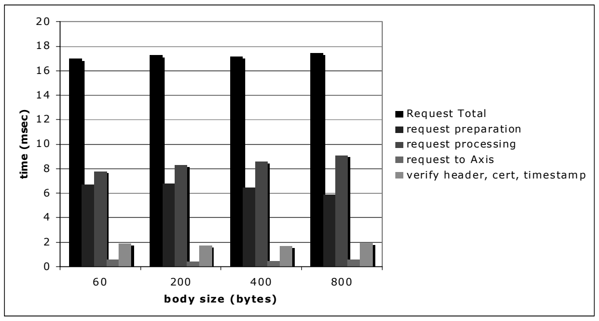

Figure 4. Average processing times of the Kerberos handler

We observe that the largest portion of the time is needed for the Request Preparation and Request Processing. As mentioned in III-A, Request Preparation is the time that the handler needs to deserialize the message and construct the Java objects, while Request Processing is the time needed by the handler to derive the secret key and decrypt the message. The interesting point is that the Request Preparation (XML SOAP processing) is a little less than Request Processing (Security processing). This gives an indication of how expensive is the use of decryption (using exclusively symmetric cryptography) in WS security comparable to the SOAP serialization.

Respectively, Figure 5 shows the various processing times of the X.509 handler. It is worth mentioning here that the time of the security processing is almost double compared to the SOAP processing.

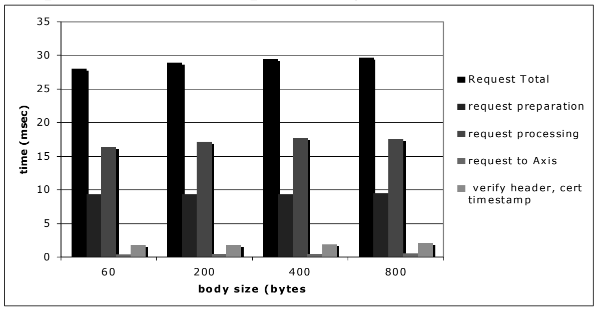

Figure 5. Average processing times of the X.509 Certificate handler

Furthermore, the results of SC1 reveal the fact that the payload size degrades the performance slightly. We have chosen payload sizes up to 800 bytes, so that the whole SOAP message does not exceed the 1500 bytes and consequently IP fragmentation is not performed. Larger message sizes might have greater performance penalties, however these messages are out of our scope, as we wanted to test how both security profiles behave when sending small messages that fit the profile of real time Grids (such as the GRIDCC project).

4.2. Scenario 2 (SC2)

The second scenario aims at revealing how both security mechanisms behave under different load. As we have mentioned, each client transmits using a fixed message rate (21 messages/sec) and fixed message body size (60 bytes).

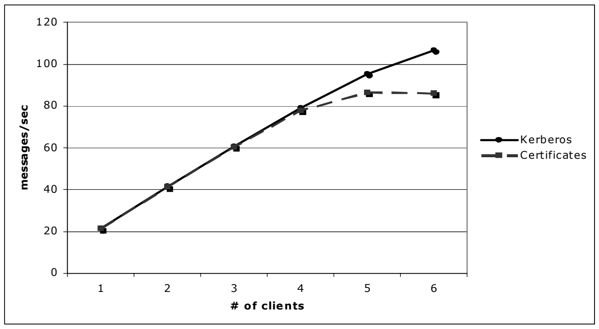

Figure 6. Average packet rate for increasing number of clients

In Figure 6, the average packet rate in each of the two handlers for increasing number of clients, is presented.

As we can observe, both Kerberos and X.509 handlers present similar performance up to the point that four clients were used. The only difference that we observed, when performing these experiments, is that the X.509 token profile exhibited higher CPU utilization. Specifically, when used four clients the CPU utilization was approximately 90% for the X.509 case, while for the Kerberos profile it was approximately 65%. The difference in the CPU utilization, as in the previous scenario, stems from the difference in the acquisition of the symmetric key. The RSA algorithm that the Certificate handler utilizes to get the symmetric key, contributes to the high CPU utilization. This difference in CPU utilization allows the Kerberos profile to escalate up to 6 clients and achieve a packet throughput of approximately107 packets/sec.

Figure 7 demonstrates the scalability of Kerberos solution, as a function of the number of clients. Although until the use of 4 clients the packet throughput is almost identical for both solutions as shown in Figure 6, Figure 7 reveals that the Kerberos handler exhibits significantly smaller processing times.

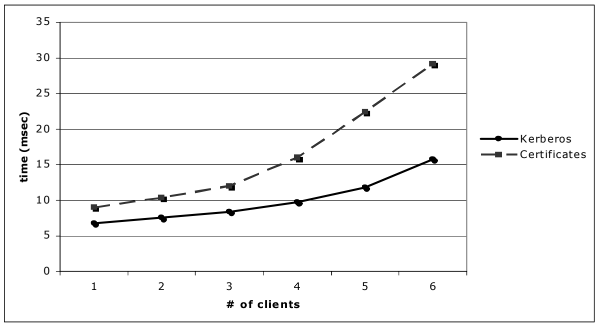

Figure 7. Average Request Total Time for the whole handler processing a single request

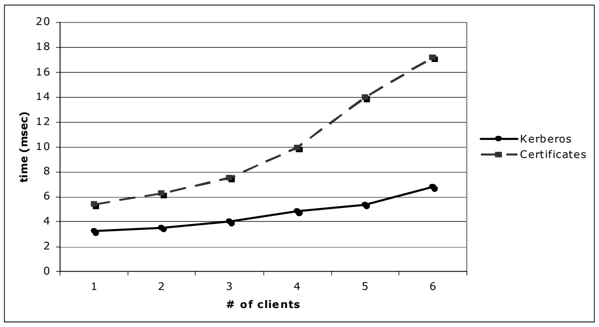

Figure 8. Average Request Processing Time for different number of clients

Figure 8 shows the average Request Processing Time for a different number of clients. It shows how the security processing (symmetric key derivation, decryption of message) that mainly is responsible for the difference in the total handler processing (Figure 7), scales as the number of clients increases.

Similarly to the SC1, Figures 9 and 10 show the average processing times of the two handlers respectively, for different number of clients. We confirm that the Request Preparation time (SOAP deserialization) is important and comparable to the Request Processing time (security processing) and the SOAP processing greatly affects the performance.

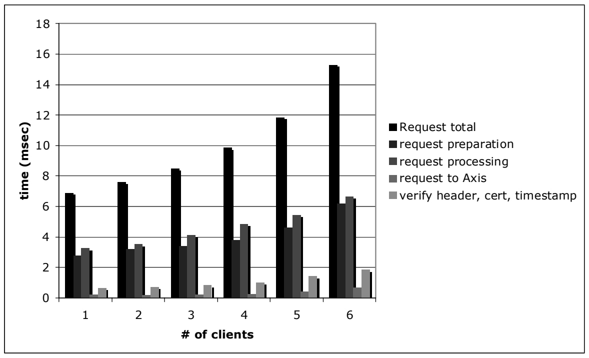

Figure 9. Average Processing times of Kerberos handler for different number of clients

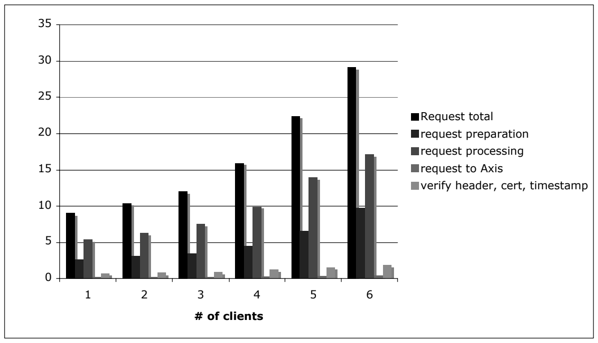

Figure 10. Average Processing times of X509 certificate handler for different no. of clients

5. Conclusions

In this paper, we have compared the WSS4J implementation of the X.509 Certificate Token Profile with our implementation of the Kerberos Token Profile (extending the WSS4J API) on encrypting SOAP message payloads. We have shown that the implementation of the Kerberos Token profile has up to 28% packet throughput improvement over the X.509 Token profile, under full CPU load on the server. This difference is attributed to the different cryptographic approach each handler utilizes. The Kerberos handler is using exclusively symmetric cryptography (AES128), where the Certificate handler is using a mixed scheme (symmetric key encrypted with RSA15 and body encrypted with the symmetric key via AES128). Furthermore, we observed that there was a small influence when using different payload sizes, ranging from 60 to 800 bytes. Additionally, under low CPU loads (and low incoming message rates) the handlers perform similarly regarding the message rate. In this situation the main difference is that the Certificate handler presents higher CPU load and the security processing time is higher.

6. References

[1] WS-Security X509 Token Profile: http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf

[2] Apache WSS4J - http://ws.apache.org/wss4j

[3] GRIDCC Project – http://www.gridcc.org

[4] A. Moralis, A. Lenis, M. Grammatikou, S. Papavassiliou & V. Maglaris, "A Distributed Kerberized Access Architecture for Real Time Grids", 4th International Workshop on Security In Information Systems WOSIS, 2006

[5] IETF RFC 1510 – "The Kerberos Network Authentication Service (V5)"

[6] IETF RFC 2459 – "Internet X.509 Public Key Infrastructure Certificate and CRL Profile"

[7] SOAP, http://www.w3.org/TR/soap12-part0

[8] Mike Olson, Uche Oqbuji, "Messaging technologies compared" http://www-128.ibm.com/developerworks/library/ws-pyth9/, Technical report

[9] Chiu, K. Govindaraju, M. Bramley, R., "Investigating the limits of SOAP performance for scientific computing", 11th IEEE International Symposium on High Performance Distributed Computing, 2002.

[10] WWW Consortium. XML, http://www.xml.org.

[11] The official CORBA standard from the OMG group http://www.omg.org/docs/formal/04-03-12.pdf

[12] The Transport Layer Security (TLS) Protocol Version 1.1 (RFC 4346)

[13] W3C WS Activity http://www.w3.org/2002/ws/

[14] WS Security Kerberos Token Profile, http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf

[15] XML Encryption WG, http://www.w3.org/Encryption/2001/

[16] XML Signature Syntax and Processing, http://www.w3.org/TR/xmldsig-core/

[17] Apache Axis http://ws.apache.org/axis/

[18] BouncyCastle JCE, http://www.bouncycastle.org/

[19] WS Security Core Specification 1.1, http://www.oasis-open.org/specs/index.php#wssv1.1

[20] Roger Needham and Michael Schroeder, "Using encryption for authentication in large networks of computers", Communications of the ACM, 21(12), December 1978

[21] H. Liu, S. Pallickara, G. Fox, "Performance of Web Services Security", in Proceedings of 13th Annual Mardi Gras Conference, Feb. 2005.

[22] Hirasuna, S.; Slominski, A.; Fang, L.; Gannon, D., "Performance comparison of security mechanisms for grid services", in Proc. 5th IEEE/ACM International Workshop on Grid Computing, pp. 360- 364, Nov. 2004

[23] M. Juric, I. Rozman, B. Brumen, M. Colnaric, M. Hericko, "Comparison of performance of Web services, WS-Security, RMI, and RMI-SSL", Journal of Systems and Software, Vol. 79: 5, pp. 689-700, May 2006