Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 3063–3074
computer communications
www.elsevier.com/locate/comcom

Evaluation of the effect of SSL overhead in the performance of e-business servers operating in B2B scenarios

Daniel F. García [a,*], Rodrigo García [b], Joaquín Entrialgo [a], Javier García [a], Manuel García [a]

[a] Department of Computer Science and Engineering, University of Oviedo, Campus de Viesques, 33204 Gijon, Spain
[b] Center for the Development of Information and Communications Technologies, Scientific Park, 33203 Gijon, Spain

Available online 23 June 2007

Abstract

In the current business to business environments, transactions between e-business servers must be carried out with a high security level. To carry out secure transactions, the servers must do additional tasks, such us exchanging encryption keys and encrypting and decrypting the information interchanged during the transactions. The combination of several specific algorithms for these tasks constitutes a cipher suite. The additional tasks degrade the server performance and the challenge is to quantify the degradation as a function of the cipher suite selected.

Until now, several research works have evaluated the impact of security on the performance provided by web servers using static and very simple dynamic contents. However there is a lack of research into the impact of security on the performance of e-business servers which execute complex transactions, some of them involving additional transactions with other servers.

This work presents an evaluation of the impact of using SSL, with several representative configurations, on the performance of e-business servers. The business application used to carry out this execution is the TPC-App benchmark, which is a good representation of business-to-business environments. The benchmark runs on a cluster of two layers.

The results of this evaluation are unexpected, because the impact of SSL on performance is small compared to the results of previous works that evaluate web servers, for which the impact of SSL on performance is very high. Therefore, this work provides insight to solve the tradeoff between security and performance when an SSL cipher suite must be selected for a complex e-business system rather than for a simple web server.

Keywords: e-Business server performance; SSL overhead; B2B environments; Impact of security on performance; TPC-App benchmark

1. Introduction

The main goal of an application server in a business-to-business (B2B) environment is to provide its services to the maximum number of concurrent business clients. Generally, the interactions between e-business servers are always carried out within secure sessions commonly based on SSL (Secure Sockets Layer) connections [10,21]. The SSL protocol was originally designed at Netscape for its web browsers and servers, but was later standardized by IETF and is now called TLS (Transport Layer Security) [9]. The SSL/TSL protocol gives security to the HTTP transactions [16] using cryptographic techniques [19] that demand computational resources. Therefore the utilization of secure connections in business sessions between two or more machines would degrade the performance of an application server, that is, the throughput of the server would decrease and its response time would increase.

In this paper, we evaluate the influence (overhead) of security on application server performance, analyzing the server behavior as a function of the number of concurrent e-business clients supported in two scenarios: with and without security.

The experimental environment is based on our implementation of the TPC App benchmark [20], which provides

* Corresponding author. Present address: Department of Computer Science, office 1.2.14, Edf., Departmental Oeste, Campus de Viesques, 33204 Gijon, Spain. Tel.: +34 985 182 066; fax: +34 985 181 986.
E-mail address: dfgarcia@uniovi.es (D.F. García).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.05.040

a realistic emulation of the workload supported by an application server in a B2B scenario.

2. Related work

The impact of security on web servers has been analyzed in the literature. These works, which evaluate the influence of security on client-to-business (C2B) environments, can be classified in two groups. In the first group, we consider the works devoted to evaluating the influence of the SSL protocol on the performance of the e-commerce servers. In the second group, we analyze the works devoted to improving the performance of the SSL protocol.

2.1. Evaluation of the influence of the SSL protocol

In a pioneer work, [14], Kinicki et al. compared the throughput and response times of an e-commerce site with and without SSL security. They carried out their performance comparison experiments on an e-commerce site made up of three computers: a web server, an application server and a database server. The authors conclude that SSL has a very high impact on the performance of C2B servers when the processing of a transaction only consists of serving one static web page. However, when the transaction involves a high level of processing in the application and database servers, the relative impact of SSL is negligible. The results are similar to the conclusions obtained in the research work presented in this paper for B2B environments.

Later in [13], Kant et al. analyzed the impact of SSL on the performance of internet servers for different configurations of the architecture of the servers. They used a static workload of a single page of 30 byte, or 36 Kbyte or 1 Mbyte, which is retrieved by all clients. The page of 30 byte is intended for analyzing the handshake performance and the page of 1 Mbyte is used for analyzing the bulk data encryption performance. The page of 36 Kbyte is used to analyze the two aspects of SSL in a combined manner. Their main conclusion was that the use of SSL increases the computational cost of the transactions by a factor of 5–7. However this result was obtained with an extremely simple workload, in which the main computational effort is due to the SSL protocol. This load scenario and the conclusion obtained from it are not applicable to the current B2B scenarios considered in the evaluation work presented in this article.

In [15], Lin et al. developed an analytical queuing model to analyze the impact of SSL on the performance of web servers when security configuration parameters are changed. The model considers https requests, with probability p, and http requests, with probability $1 - p$. Each type of request carries out a different sequence of activities in the queues of the model. The authors evaluated the increment of response time when the parameter p increases and also when the RSA key length is incremented. Finally, they evaluated the influence of three different symmetric encryption algorithms. This model is inappropriate for current web servers with a high proportion of dynamic content. Furthermore, it is not representative of current e-business servers considered in this research work, in which the web server interacts with back-end systems composed by an application and database servers.

In [7], extended in [8], Coarfa et al. analyzes the increment of the web server throughput when individual components of the TSL protocol are eliminated. Coarfa et al. carried out the experiments by executing two different traces on an Apache web server. One trace was obtained from the purchase of a book in Amazon, but discarding all the back-end processing. This trace is not representative of the global load supported by a typical web server operating in C2B environments, because it only considers an isolated operation. The other trace was obtained from the general web server of a computer science department of a university, which is not representative of real e-business environments either. The main result of this work is the characterization of the relative impact of the components of the TSL protocol on performance, showing that the initial key exchange is the operation with the highest impact. However, only a few of the results provided are useful for the evaluation of the global impact of SSL on server performance, because they are obtained emulating non-realistic operational conditions. For example, it is useless to interchange secret keys in the handshake, but then not use them to encrypt the data transferred or vice versa.

Beltran et al. evaluated in [3] the performance impact of SSL on web applications using the RUBiS benchmark, which realistically emulates an auction site, including browsing, selling and bidding operations. The experimental platform has two computers, an application server based on Tomcat and a database server based on MySQL. Additional computer injects the load using the Httperf tool. This experimental approach, based on a representative benchmark executed on a cluster of two servers, is similar to the evaluation approach presented in this article. However the results are very different. Beltran et al. found that the use of SSL greatly reduces the maximum throughput of a web server and they indicate that this result is probably due to the configuration of the load injector. When the server is heavy loaded and can not serve a client, this client tries to reconnect immediately. These continuous connection trials, each one requiring the execution of a full SSL handshake, consume more and more processor time, further reducing the throughput of the server. Under these experimental conditions, the evaluation of SSL overhead behaves like a denial of service attack.

In [22], Zhao et al. carried out an extensive study of the performance costs of the SSL processing without including external operations. For the standalone SSL analysis they developed a small Crypto-Benchmark. This is a program that creates a server context as well as a client context in a single computer, and sends messages between the two through memory buffers. This program was executed in a virtual emulation environment that allows the CPU cycles

consumed in each phase of the SSL protocol to be counted. The authors analyze the performance execution of the SSL protocol only on the server side, totally isolated from the other software components of the server. They do not provide any information about the influence of SSL on the global performance of the e-business servers.

2.2. Improvement of the performance of the SSL protocol

Other research works have focused on the analysis and optimization of the SSL/TSL protocol. Almost all of them try to improve the performance of the handshake phase, but none of them consider the phase of bulk data transfer.

Goldberg et al. showed in [12] that reusing cached SSL session keys can significantly reduce client response time. This is an interesting measurement-based work, but exclusively focused on a single aspect of the SSL performance.

In [1], Apostolopoulos et al. evaluated the SSL protocol using the SPECWeb96 benchmark. They provided the latency of web requests as a function of the number of requests served per second, and for several levels of reuse of the cached session keys. They proposed two modifications for improving the performance of SSL-based communications. In a later work, [2], they reduced the SSL overhead on clusters of web servers by sharing the cached session keys between all the nodes of the cluster. The main drawback of these evaluation works is the workload used, the SPECWeb96 benchmark, which only uses static content composed by html files without any kind of back-end processing in an application/database server. This benchmark is not representative of the current workload of e-business servers.

In the work [17], Shacham and Boneh proposed a batch mechanism to process the RSA decryption that reduces the processing times but increases latency in the initial handshake phase. They have also developed fast variants of the RSA algorithm to speed up decryption in [5]. These works are focused on the optimization and speedup of the handshake phase of the SSL protocol, without providing an evaluation of the general impact of the SSL processing on the performance of an e-business server.

In article [18], Stubblefield et al. proposed two new systems to reduce the performance impact of web security. These systems retain backward compatibility with standard SSL/TSL technology. The first system, called SCREAM, is for content generated dynamically, which is not cacheable. It provides a mechanism to balance the SSL load in a cluster of web servers based on maintaining the SSL session state in a centralized dispatcher. It also applies a batching optimization technique to RSA operations. The second system, called WISPr, is for static content. The proposed systems can improve the performance of a cluster of web servers operating with SSL. However, the authors present few quantitative results of the performance improvement obtained with these techniques.

In a recent work, [6], Castellucia et al., proposed a new technique, termed Client-Aided RSA, oriented to incrementing the work performed by the clients in RSA processing in order to alleviate the overhead of the servers. This work is also devoted to speeding up the SSL handshake phase, but the impact of the speedup on the global server is not evaluated.

Finally in [4], Bicakci et al. proposed a modification of the standard SSL protocol so that it operates in a reverse-like mode. With the modification, the server has to encrypt the session master key, a fast operation, and to generate a signature, a slow operation. But the signature generation can be partitioned in two phases, one off-line and the other on-line. This modification reduces the server overhead in the handshake phase, but the resulting protocol is not compliant with the standard SSL.

All but one of the related research works analyzed highlight the high impact of the SSL protocol on the performance of e-business servers. However, all this research was carried out in C2B environments. The lack of research in B2B environments has motivated the development of the work presented in this paper.

3. The TPC-App workload and its security issues

The application used for this evaluation work is the TPC-App benchmark. Although several benchmarks have been developed to evaluate the performance of servers operating in B2B scenarios (see [11]) none of them provide such a complete recreation of the operations executed by the servers as the TPC-App benchmark. Furthermore, the Transaction Processing Performance Council (TPC) is the most important organization in the world which specifies the benchmarks for evaluating the performance of transactional systems, and the most important hardware and software manufacturers of the world are full members of TPC. They jointly developed the specification of the TPC-App to properly represent the workload supported by application servers in B2B environments.

A basic knowledge of the TPC-App benchmark is very important to analyze and interpret the final results of the evaluation work correctly. Therefore, the following subsections briefly present the web services implemented in the benchmark and their invocation. Following this, the configuration and operation of the SSL protocol with the TPC-App benchmark is explained in more detail.

3.1. Services exposed and used by the TPC-App benchmark

The TPC-App benchmark emulates the activities of a B2B transactional application server, commonly called the system under test (SUT) in performance evaluation studies. Another computer, called the remote business emulator (RBE), sends requests to the SUT and waits for the responses. The RBE emulates multiple client e-business servers requesting services concurrently from the SUT. During the operation, the SUT requires four services which are provided by external vendors. All the services provided by the application server and the external vendors are remote methods implemented as web services.

Fig. 1 shows the remote methods exposed by the SUT to the RBE as well as the remote methods of the external vendors invoked by the SUT.

The application server exposes seven services in its presentation layer: *Order Status*, *Change Payment Method*, *New Products*, *Create Order*, *New Customer*, *Product Detail* and *Change Item*. The most complex service is *Create Order* because it calls on two internal management processes, *Shipping* and *Stock*, to manage asynchronous requests. All the information used by the web services and the management process is retrieved from and stored into a stand alone database server.

The External Vendors machine provides four services to the application server: *Purchase Order Validation* (POV), *Payment Gateway Emulation* (PGE), *Shipment Notification Emulation* (SNE) and *Inventory Control Emulation* (ICE).

3.2. Service invocation scenario modeled in the TPC-App benchmark

An understanding of the manner in which the RBE invokes the services of the SUT is essential to assess the influence of SSL on the performance of the application server.

The RBE is a multithreaded process, in which each thread emulates an e-business client computer, called an Active EB, which continuously requests the services of the application server. All the interactions between the client and the server are carried out within a Business Session. To start a new business session the client must create a new connection socket using the SSL/TSL protocol by executing a full handshake. Then, all the interactions of the session are carried out through the secure connection. SSL/TSL sessions must not be shared by multiple Business Sessions.

When the Business Session ends, the active EB may close any SSL/TSL session and/or all network connections that are currently established for the Business Session.

At the beginning of a new Business Session, the Active EB calculates the number of requests to be fulfilled out within the session. This number is called the Business Session Length (BSL) and is calculated as a random value obtained from a Beta distribution scaled between 1 and 120. Within a Business Session, the Active EB continuously

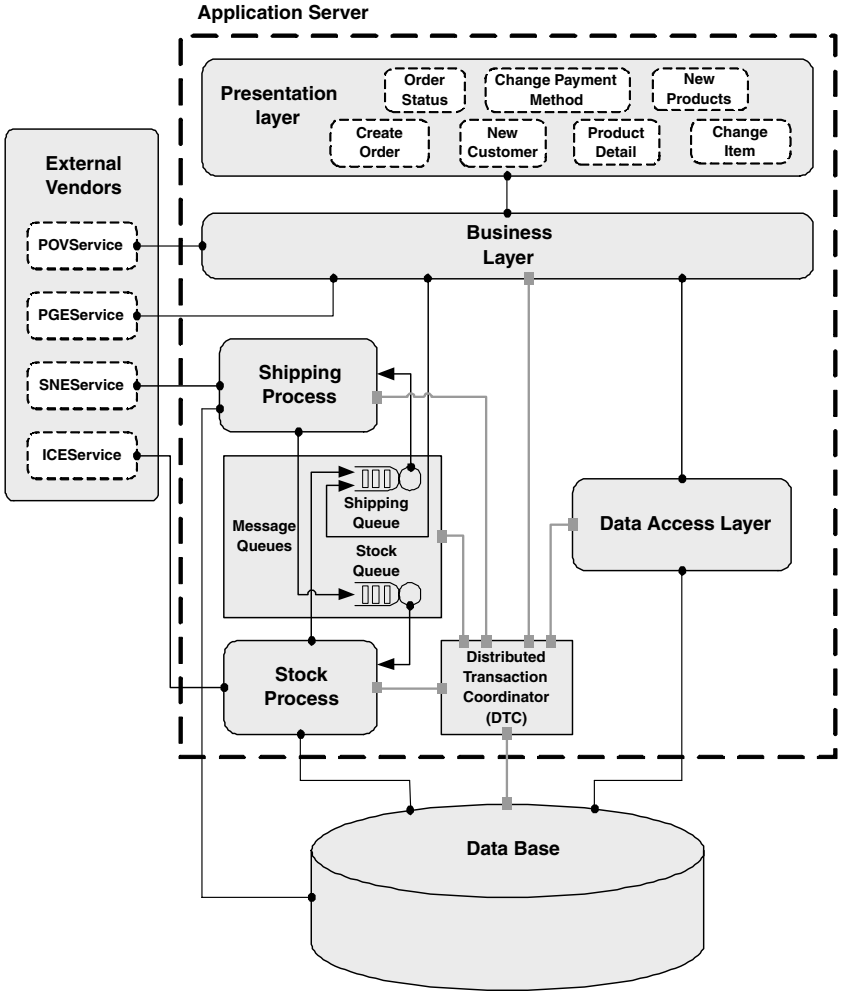

Fig. 1. Software architecture of the application server and interactions with the external vendors and the database server.

sends web service requests, with no think time between successive web service requests.

Below, the client-server interaction scenario modeled by the benchmark is analyzed. The RBE emulates multiple remote application servers (client computers) sending requests to the local TPC-App server. Each Active EB (thread) of the RBE reproduces the behavior of different client computers sequentially opening business sessions on the TPC-App application server. Therefore, the benchmark models a typical B2B scenario, in which, multiple remote application servers send requests to the local TPC-App server.

The model of interaction between the client computers and the application server that is implemented by the TPC-App benchmark involves a high level of performance optimization when the interactions are carried out using the SSL protocol, because only one SSL handshake is required for each e-business session, rather then for each interaction.

3.3. Configuration and operation of the SSL protocol with the TPC-App benchmark

All the information interchanged between the RBE and the SUT, and between the SUT and the external vendors must always be encrypted. The specification of the TPC-App benchmark allows the use of two cipher suites to carry out the encryption tasks: SSL_RSA_WITH_RC4_128_MD5 or TSL_RSA_ WITH_RC4_128_MD5.

The SSL protocol has two main phases of operation: the SSL handshake and the SSL bulk data transfer. A brief review of these phases is presented below, indicating the particular options used by the TPC-App benchmark.

In the handshake phase, the client and the server interchange 9 messages in order to create the symmetric keys used for fast encryption, decryption and tamper detection in the session that follows. This message interchange is illustrated in Fig. 2.

The client sends a *ClientHello* message proposing SSL options and the server responses with a *ServerHello* message which contains the selected SSL options. Next the server sends its public key certificate in a *Certificate* message and concludes its part of the negotiation with a *ServerHelloDone* message. The client responds by sending the session key information, encrypted with the public key of the server, in a *ClientKeyExchange* message. These steps are carried out with the RSA algorithm. After this, the client sends a *ChangeCipherSpec* message to activate the negotiated options for all the future messages it will send. Next, the client sends a *Finished* message to let the server check the newly activated options. The server responds by sending a *ChangeCipherSpec* message to activate the negotiated options for all the future messages it will send. Finally, the server sends a *Finished* message to let the client check the newly activated options. At this point the handshake is completed and the client and server may begin to exchange application layer data.

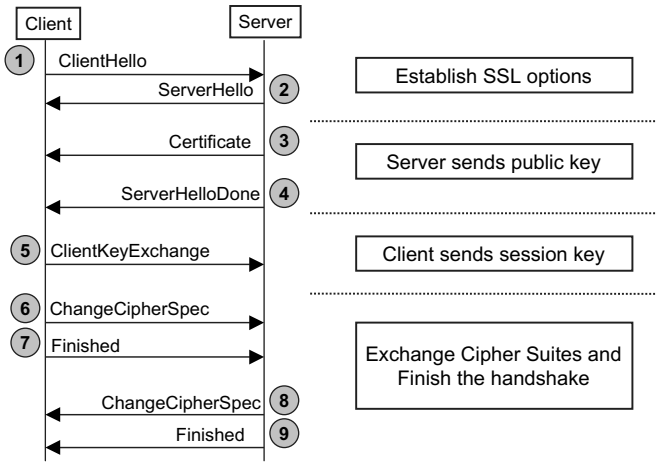

Fig. 2. The SSL full handshake protocol used in the implemented TPC-App benchmark.

It is important to observe the total absence of messages to authenticate the client machine to the server. The benchmark requires SSL/TSL with certificates from server to client, but it does not require client certificate authentication because a review of typical customer environments shows that client certificates were rarely used.

Another important aspect to remark in the handshake is the absence of a *ServerKeyExchange* message from the server to the client just after the *Certificate* message. Therefore, the same public key information used to verify the identity of the server is also used to encrypt key values in the *ClientKeyExchange* message generated by the client.

In the bulk data transfer phase, the SSL record layer receives uninterpreted data from higher layers in non-empty blocks of arbitrary size. The record layer fragments information blocks into SSL plain text records of 2^{14} bytes or less. The records can be compressed using the compression algorithm defined in the current session state. Then, a MAC (Message Authentication Code) is calculated for the record using the hashing algorithm defined in the CipherSpec negotiated in the handshake phase. The MAC is appended to the end of the record data. Finally the record data and the MAC are encrypted using the algorithm defined in the current CipherSpec. The operations carried out during the bulk data transfer phase are illustrated in Fig. 3.

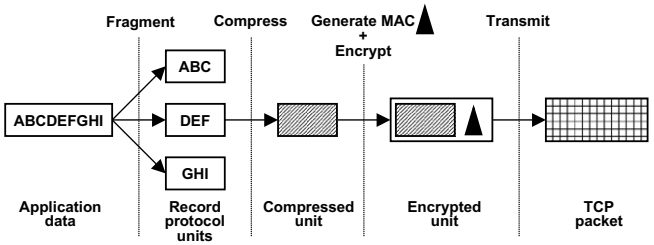

Fig. 3. Operations and data transformations carried out during the bulk data transfer under the SSL protocol.

The implementation of the TPC-App benchmark used in this evaluation work does not compress the information. The benchmark uses the hashing algorithm MessageDigest5 (MD5) to generate the MAC and the stream based symmetric cipher algorithm RivesCipher4 (RC4) to encrypt the information.

4. The experimental environment

The experimental work has been carried out on a set of machines interconnected as shown in Fig. 4. The Remote Business Emulator, the Application Server and the Database Server are similar machines. All of them use two Pentium-III processors at 1.1 GHz and have 2 GB of RAM. The Database server has two RAID-5 units of SCSI disks, while the other two servers have common IDE disks. The External Services Provider uses a single Pentium-III processor at 800 MHz. The four machines are connected by a 100 Mbps Ethernet switch and all of them run the Windows server 2003 operating system.

The implementation of the TPC-App benchmark has been mainly developed in C# language using the .NET framework. In the application server, the web services are hosted by the Internet Information Server (IIS) V6.0 and are executed within the ASP.NET engine V1.1. The database server runs the Microsoft SQL Server 2000 database engine. A set of stored procedures has been implemented in the database server in order to reduce the computational load supported by the application server.

In relation with the configuration of the security issues under the Windows operating system, several steps have been taken to establish security before carrying out the experiments.

Firstly, the Microsoft Certificate Services were used to obtain the server certificate, filling in the information of the server and choosing a Cryptographic Services Provider (CSP), typically a library that provides algorithms to carry out cryptographic tasks. Each CSP provides a specific set of algorithms and supports different key lengths. In this evaluation work, the Microsoft Enhanced Cryptographic Provider was selected. This CSP has default key lengths of 1024 bits for the certificate of the server (RSA) and of 128 bits for symmetric encryption of data (RC4).

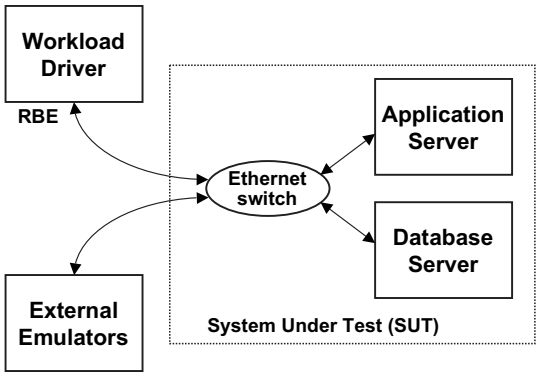

Fig. 4. Architecture of the experimental environment.

Secondly, the server certificate was installed and assigned to the IIS virtual directory hosting the web services. Finally, the certificate of the server was installed in the RBE client in order to establish a trust relationship between them. Similar operations must be done to configure the security of the external vendors.

When the RBE starts a new business session, it must also open a new connection with the application server by executing a full SSL handshake. In the .NET environment the connections are pooled and the RBE process tries to reuse connections of the pool. To force the opening of a new connection, a new *ConnectionGroupName* must be established at the beginning of each business session. Otherwise the full SSL handshake would be carried out only by the first service interaction of the first business session.

5. The design of the experimental work

In order to evaluate the overhead produced by the utilization of the SSL protocol in e-business servers an experimental design has been developed.

The first step of an experimental design is to select the appropriate response variables and factors, which are variables or configuration parameters that affect the behavior of the response variables. These variables can have several alternatives or levels. Fig. 5 shows the selections for this work.

The main response variables that represent the performance of the servers are the response times of the interactions and the throughput. Secondary but important response variables are the utilization of devices. The utilizations can be considered as secondary because the saturation of devices is reflected in the throughput, but it is important to know the first device that reaches the saturation and therefore limits the maximum throughput.

The most important factor is the utilization or non-utilization of the SSL protocol during the operation of the server. The next most important factor is the load intensity supported by the server during the operation, because it is expected that the influence of the SSL overhead varies according to the level of the load intensity. Therefore, the load expressed by the number of concurrent EBs supported by the server varies from 1 to 40 EBs, in all the experiments.

Finally, the secondary factors include the multiple configuration possibilities of the SSL protocol to establish an

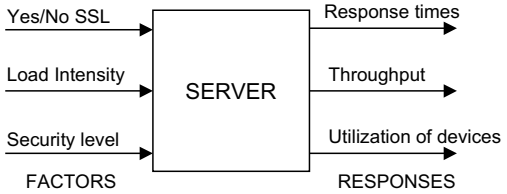

Fig. 5. Response variables and factors in the experimental design.

appropriate security level. These possibilities include the selection of algorithms for:

– The handshake (DHE/DDS, RSA).
– The bulk encryption and decryption of information (RC2, RC4, DES, 3DES).
– The generation of the Message Authentication Code using hashing functions (MD5, SHA).

Additionally, the length of the keys used by the algorithms can be also selected within predefined ranges.

Each specific combination of the above algorithms constitutes a cipher suite that can be specified using the symbolic names of the algorithms used. A good example is the suite used by the TPC-App benchmark:

SSL_RSA_WITH_RC4_128_MD5

The name before WITH specifies the algorithm used for the handshake and the name after WITH indicates the algorithm used to encrypt/decrypt the information jointly with the length of the key used. The algorithm used to generate the MAC is specified at the end.

Combining all the algorithms between them and considering the different key lengths, an enormous number of different cipher suites can be obtained. Although not all the combinations are used, there is a small set of common combinations. Furthermore, when the application server is configured, a specific Crypto Service Provider is used. Each provider offers only a particular set of combinations.

The evaluation of the influence of security level on performance is accomplished using the cipher suite specified by the TPC-App benchmark as the basis for comparisons, because it provides the best performance while maintaining a good security level. All the results obtained with this base suite are represented in Figs. 6–12 by black circles joined with a continuous black line.

A first set of experiments evaluates the performance improvement when SSL is not used and the performance degradation when a new SSL session is opened for each interaction, with respect to the base suite.

In a second group of experiments, suites with less or more security than the base suite are evaluated successively. For the selection of the suites the following factors were taken into account.

For the handshaking phase there is no evidence in the literature of any clear differences in the performance or in the security levels of the two common algorithms used in this phase (DHE/DDS and RSA). Therefore, we always use the RSA algorithm and the security level is varied by increasing or decreasing the length of the public key used.

In the bulk data transfer, for the encryption/decryption of information, the RC4 algorithm is faster than the DES and RC2 algorithms. The 3DES algorithm is even more secure but it is slow when implemented in software. For generating the MAC, MD5 is faster and less secure than SHA. In this evaluation work, we have compared the base suite, which uses RC4 of 128 bits and MD5, with two other suites: a more secure suite that uses 3DES of 168 bits with SHA and a less secure suite that uses RC4 of 40 bits and MD5.

6. Experimental results

In this section, we present the experimental results organized in three subsections. The first shows the comparison

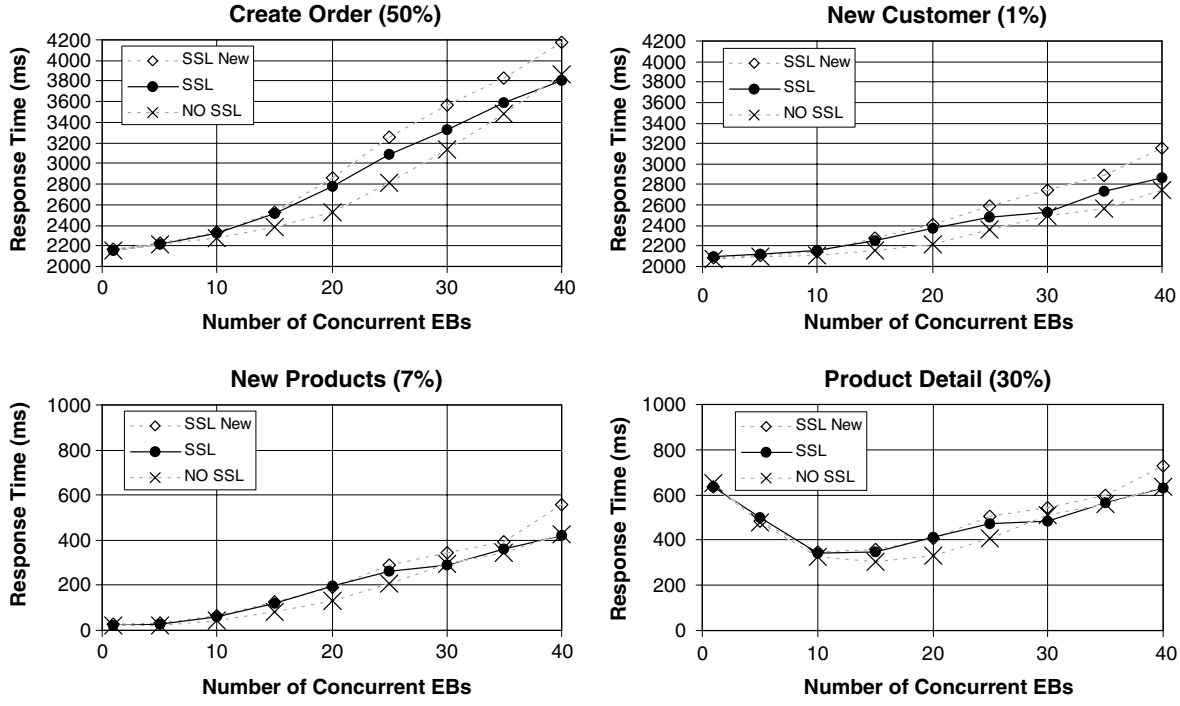

Fig. 6. 90-Percentile response times of four interactions of the TPC-App benchmark.

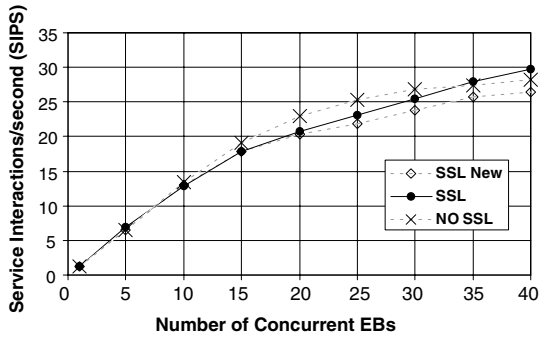

Fig. 7. Throughput of the Application Server, with and without SSL security.

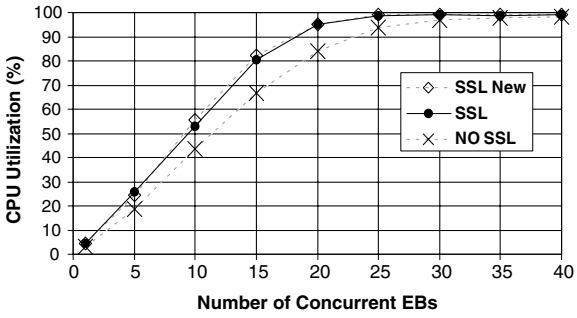

Fig. 8. CPU utilization of the Application Server, with and without SSL security.

of the performance when using and not using the SSL protocol. The next two show the influence of the selected security level on the server performance.

6.1. Influence of using the SSL protocol on server performance

In order to evaluate the influence of using SSL on performance, we have carried out an experiment without security and also the same experiment using SSL in an efficient manner: opening a new SSL session for a complete e-business session containing multiple web requests. In addition, we repeated the experiment using SSL in an inefficient manner: opening a new SSL session for each web request.

The results obtained from this evaluation are summarized in several graphics. Firstly, we present the 90-percentile of the response time of four representative interactions in Fig. 6. In each graph, the measurements without SSL are represented by crosses; the measurements with SSL opening a secure session for the set of interactions of a business session are represented by black circles; the measurements with SSL opening a new secure session for each interaction are represented by diamonds.

In the TPC-App benchmark, 50% of the interactions are of the "Create Order" type, which calls external services, and 30% of the interactions are of the "Product Detail" type, which does not call external services. Therefore, the two graphs on the left in Fig. 6 represent the response time of 80% of all the interactions carried out during the benchmarking experiments. On the right, we have included two more interactions, "New Customer", which calls external services, and "New Products", which does not call external services.

In order to compare the response times easily, the origin of the vertical scale for the upper graphs starts at 2000 because the external interactions requested by that service introduce a minimum delay of 2000 ms.

The graphs show three different behaviors as a function of the load intensity, that is, the number of concurrent EBs supported by the server. With a low load, there is enough CPU time to process the SSL load and there is no noticeable difference between using SSL or not. Next, under medium loads the utilization of SSL provokes increments of response times. Finally with high loads, the opening of a new secure session for each interaction increases the response times more and more.

The "Product Detail" interaction shows a high response time for a minimal number of EBs. When the number of EBs increases, the response time decreases until a valley value, to increase again with the number of EBs. "Product Detail" is the only interaction that accesses the server disk to recover 1 image of the 100,000 images stored in the disk. As the load experiments are carried out, progressively increasing the number of EBs (starting from 1), the lack of caching of the images in the disk buffer cache affects the response time of the initial experiments.

The throughput of an application server, measured in SIPS (Service Interactions per Second), is also affected by the use of the SSL protocol. Fig. 7 shows the results of the executions of the TPC-App benchmark. The SUT provides greater throughput without SSL (curve with crosses) than with SSL (curve with black circles and curve with diamonds), as expected, but only under non-resource saturation conditions. When the load is greater than 30 Active EBs, the throughput without SSL becomes increasingly similar to the throughput with SSL. The maximum decrement of the throughput is of 2.5 SIPS. When a new SSL session is opened with each interaction (curve with diamonds) the throughput of the server decreases for all load levels. The maximum decrease of the throughput is of 5.5 SIPS.

The resource utilization is shown in Fig. 8. As expected, the versions of the TPC-App benchmark using SSL security consume more CPU than the non-secure version, under non-saturation conditions.

We have not included the utilization of other resources for two reasons: Firstly, the other resources of the application server (disk, network, memory) increase their utilization at rates much lower than the CPU when the number of Active EBs is increased. Secondly, the use of the SSL protocol mainly affects the CPU resource.

The device utilizations of the other computers involved in the experiments are not shown because they show low values and are in no way performance bottlenecks.

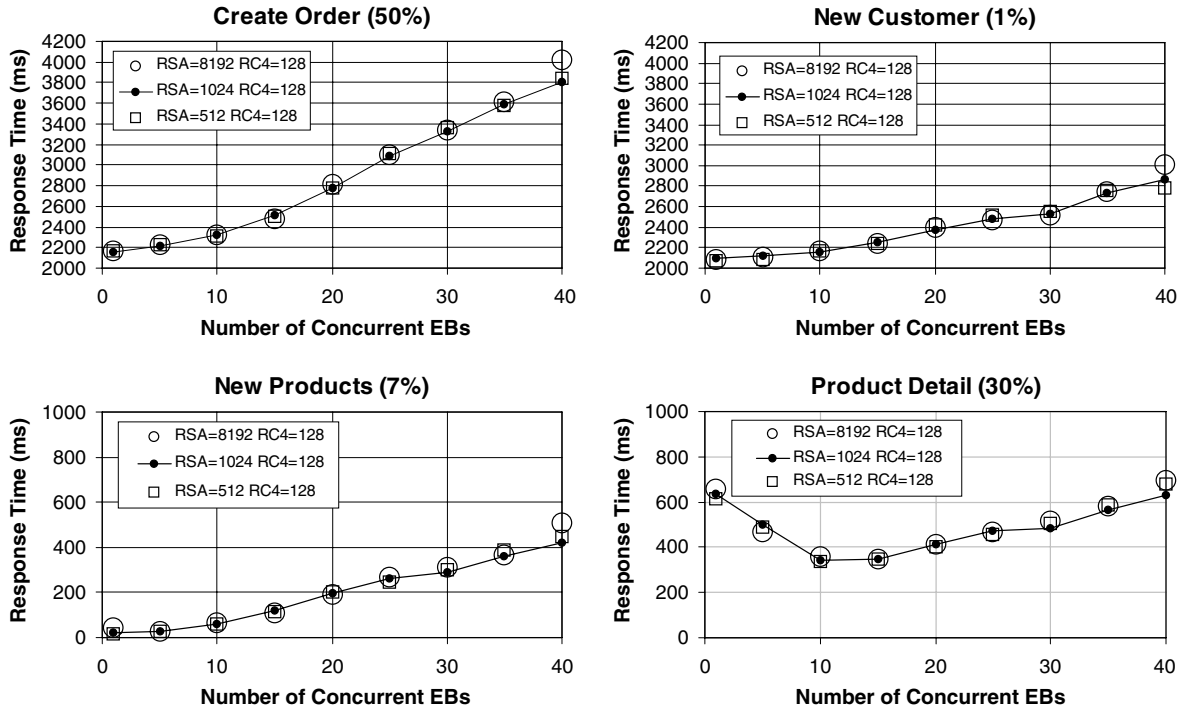

Fig. 9. Ninty-percentile response times of four interactions of the TPC-App benchmark when the length of key used by the RSA algorithm for the handshake phase changes.

The main conclusion obtained from these experiments is that the utilization of SSL reduces the performance of the application servers. This reduction can be clearly appreciated in the throughput curve of the server, and can be quantified in a value between 5% and 10%. The throughput reduction is clearly related to the CPU overhead caused by the use of the SSL protocol in the application server. However, the 90-percentile response times of the interactions increase by very small amounts which are imperceptible for the client business.

6.2. Influence of the security level for the handshake on server performance

After the analysis of the influence of using SSL on the server performance, the next step is to evaluate the influence of the security levels used on the performance. This evaluation is accomplished in two steps: by evaluating the influence on performance of the security level established firstly for the handshake phase and secondly for the bulk data transfer phase.

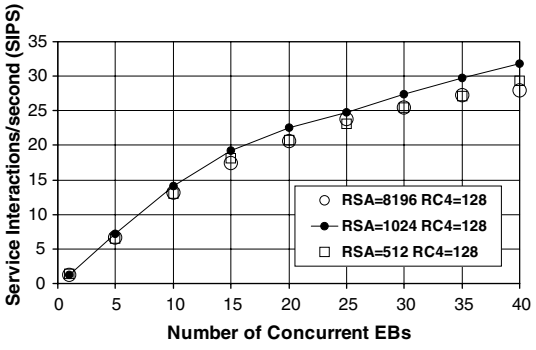

Fig. 10. Throughput of the Application Server, when the key length of the RSA algorithm changes.

In order to evaluate the influence of the security level specified for the handshake on performance we have carried out an experiment increasing and decreasing the length of the key used by the RSA algorithm in the handshake phase. The graphics of Fig. 9 show that there is no appreciable difference between the 90 percentiles of the response times of interactions when the length of the key is increased from 1024 to 8192 bits (curves with circles) or decreased from 1024 to 512 bits (curves with squares).

To assure that the difference between the response times obtained using different lengths of the RSA keys are negligible, an analysis of variance (ANOVA) of two factors without replications is done. The factors are the load represented by the number of concurrent EBs and the length of the RSA key. Table 1 shows the percentages of variation of response times explained by each factor and the residual error. Changes in key lengths are shown to have a negligible influence on response times.

Table 2 shows the values used in an F-test. The F_M columns contain the quantiles obtained from the measurements and the F_T column contains the theoretical quantile of the F distribution calculated for a significance level alpha of 0.1 using the degrees of freedom of each factor. In the F-test a factor explains a statistically significant part of the observed variation if the measured value, F_M, of the factor is greater than its corresponding theoretical

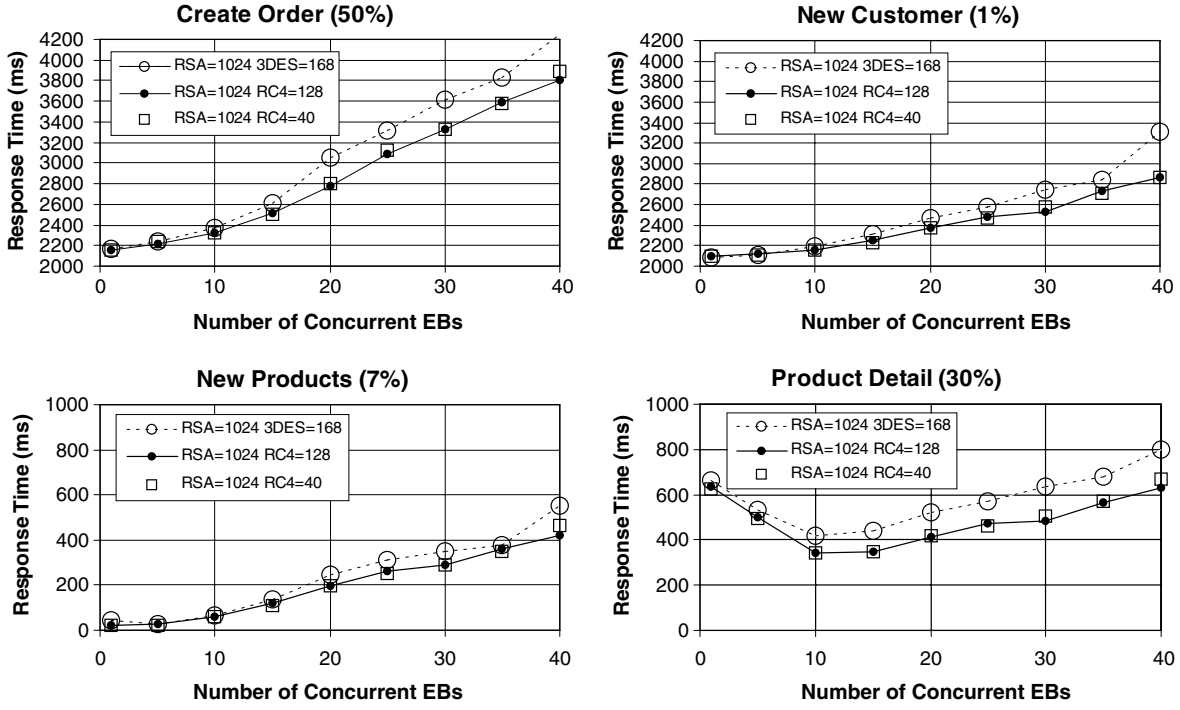

Fig. 11. Ninty-percentile response times of four interactions of the TPC-App benchmark when the encryption/decryption algorithm for the bulk data transfer changes.

value, F_T. This is always verified by the load factor but never by the length factor.

The throughput of the application server is slightly affected by changes in the key length of the RSA algorithm. Fig. 10 shows that the best performance is obtained with the base configuration (curve with black circles) and a similar reduction of throughput can be observed when the key length is changed.

The configuration of the security level used in the handshake phase can slightly reduce the throughput of the server but it does not noticeably affect the response times perceived by the clients.

6.3. Influence of the security level for the bulk data transfer on server performance

In order to evaluate the influence of the security level specified for the bulk data transfer on performance, we have carried out an experiment selecting the best cipher suite offered by the cryptographic service provider used in this evaluation work and comparing it with the base suite used always in this work. The best suite uses the 3DES algorithm for encryption/decryption of the information using a key length of 168 bits and the SHA algorithm to generate the message authentication code (MAC). Both algorithms provide a higher security level than the algorithms used by the base cipher suite, RC4 for encryption/decryption of data and MD5 to generate the MAC. However, 3DES and SHA require more CPU time than RC4 and MD5, respectively. In addition, we carried out a fur-

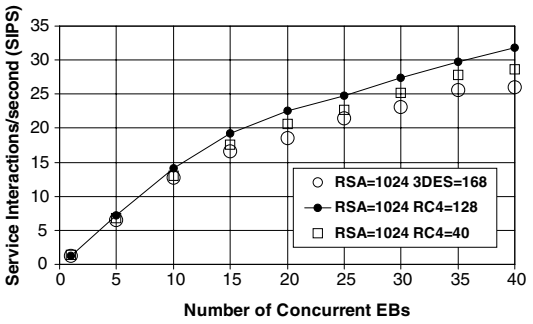

Fig. 12. Throughput of the Application Server, when the encryption/decryption algorithm changes.

Table 1
Percentages of response time variation explained by load and length factors

Factor	Create order (%)	New customer (%)	Product detail (%)	New products (%)
Load	99.70	98.51	98.43	99.14
Length	0.04	0.06	0.23	0.16
Error	0.26	1.43	1.34	0.70

Table 2
PValues of the *F*-test to evaluate the statistical significance of load and length factors

Factor	F_M				F_T
	Create order	New customer	Product detail	New products	
Load	780.2	137.6	146.7	281.6	2.09
Length	1.35	0.35	1.35	1.77	2.67

ther experiment degrading the security of the basic cipher suite using a key length of 40 bits for the RC4 algorithm. However, this small key length does not provide the minimum security level required by the current e-business applications.

The graphics of Fig. 11 show appreciable increments of the response times of the server when the best cipher suite is used (curves with circles) but the reduction of the key length of the RC4 algorithm does not affect the response time of interactions (curves with squares) with respect to the times of the base cipher suite (curves with black circles).

Fig. 12 shows the throughput of the server for three different encryption/decryption algorithms. A reduction of the throughput can be appreciated clearly when the 3DES algorithm is used but also when the length of the key used by RC4 is reduced.

However, the security level established for the bulk data transfer clearly reduces the throughput of the server by up to 20% and increases the response times up to 10% when very secure cipher suites are used.

7. Conclusions

The general conclusion of this evaluation work is that SSL has a small impact on the performance of servers. The performance degradation can be quantified at 5–10%.

These conclusions contrast with the results of other evaluation works, in which the influence of SSL on performance is much higher. These contradictory results are easily explained by the differences in the client-server interaction models used in the evaluation works.

In most evaluation works, each interaction requires a full or resumed SSL handshake and very few computations in the application server. This model is not representative of the current B2B environments.

In the environment modeled by the TPC-App benchmark, all the interactions between two businesses are carried out within one business session. Only one full SSL handshake is performed at the beginning of the session and an average of 50 interactions are carried out per session. The matching of the secure sessions with the business sessions allows a maximum reduction of the SSL handshakes, and therefore, the main SSL overhead is due to the encryption/decryption of the information exchanged. In TPC-App benchmark most interactions require a great number of computations in the application server, and therefore, the computational load due to SSL operations only represents a small fraction of the global load supported by the server. The load scenario modeled by the TPC-App benchmark, is highly representative of the current B2B environments.

It must be said that the effect of the SSL configuration on the performance of e-business servers has not been clearly established until now for real e-business applications. Most previous works evaluate web servers with static and simple contents. This article presents the results of a seminal work to understand the implications of security on the performance of current e-business servers.

Considering the results of this evaluation work, the developers and administrators of e-business systems can apply SSL technology in B2B environments focusing almost exclusively on the aspects related with security. They can select the algorithms and the key lengths for the two phases of the SSL protocol that they consider most appropriate to reach their security objectives, with no need to concern about on the performance implications of their selections. This freedom is possible because the configuration of SSL has marginal impact on performance.

References

[1] G. Apostolopoulos, V. Peris, D. Saha, Transport layer security: how much does it really cost? in: Proceedings of the Conference on Computer Communications, INFOCOM'99. New York, USA, 1999.
[2] G. Apostolopoulos, V. Peris, P. Pradhan, D. Saha, Securing electronic commerce: reducing the SSL overhead, IEEE Network 14 (4) (2000) 8–16.
[3] V. Beltran, J. Guitart, D. Carrera, J. Torres, E. Ayguade, J. Labarta, Performance impact of using SSL on dynamic web applications, in: Proceedings of XV Jornadas de Paralelismo, JP'04. Almeria, Spain, 2004.
[4] K. Bicakci, B. Crispo, A.S. Tanenbaum, Reversed SSL improved server performance and DoS resistance for SSL handshakes. Paper 2006/212 in the Cryptology ePrint Archive, 2006. http://eprint.iacr.org.
[5] D. Boneh, H. Shacham, Fast variants of RSA, CryptoBytes Newsletter 5 (1) (2002) 1–9.
[6] C. Castellucia, E. Mycletun, G. Tsudik. Improving secure server performance by rebalancing SSL/TSL handshakes, in: ACM Symposium on Information, Computer and Communications Security (ASIACCS'06). Taipei, Taiwan, 2006.
[7] C. Coarfa, P. Druschel, D.S. Wallach, Performance analysis of TSL web servers, in: Proceedings of the Network and Distributed Systems Security Symposium, NDSS'02, San Diego, USA, 2002, pp. 6–8.
[8] C. Coarfa, P. Druschel, D.S. Wallach, Performance analysis of TSL web servers extended version, ACM Transactions on Computer Systems 24 (1) (2006) 39–69.
[9] T. Dierks E. Rescorla. The Transport LayerSecurity (TLS) Protocol, version 1.1. RFC 4346, 2006. http://www.ietf.org/rfc/rfc4346.txt.
[10] A.O. Freier, P. Karlton, C. Kocher. The SSL protocol, version 3.0, 1996.
[11] D.F. Garcia, J. Garcia, M. Garcia, I. Peteira, R. Garcia, P. Valledor, Benchmarking of web services platforms, in: Proceedings of 2nd International Conference on Web Information Systems and Technologies, WEBIST 2006. Setubal (Portugal), 2006, pp. 75–80.
[12] A. Goldberg, R. Buff, A. Schmitt, Secure web server performance dramatically improved by catching SSL session keys, in: Proceedings of the Workshop on Internet Server Performance, WISP'98. Madison, Wisconsin, USA, 1998.
[13] K. Kant, R. Iyer, P. Mohapatra. Architectural impact of secure socket layer on internet servers, in: Proceedings of the International Conference on Computer Design, ICCD'00. Austin, Texas, USA, 2000.
[14] B. Kinicki, D. Finkel, M. Mikhailov, J. Sommers, S. Cunningham, Y. Elkin, R. Gopalan, M. Quinlivan, Electronic commerce performance study, in: Proceedings of the Euromedia 98 Conference. Leicester, UK, 1998.
[15] X. Lin, J.W. Wong, W. Kou, Performance analysis of secure web server based on SSL, Lecture Notes in Computer Science 1975 (2000) 249–261.
[16] E. Rescorla, HTTP over TSL, RFC 2818, 2000.

[17] H. Shacham, D. Boneh, Improving SSL handshake performance via batching, Lecture Notes in Computer Science 2020 (2001) 28–43.
[18] A. Stubblefield, A.D. Rubin, D.S. Wallach, Managing the performance impact of web security, Electronic Commerce Research 5 (1) (2005) 99–116.
[19] S. Thomas, SSL and TSL essentials – securing the web, Wiley Computer Publishing, USA, 2000.
[20] TPC Consortium. The TPC-App (Application Server) benchmarks. Transaction Processing Performance Council. http://www.tpc.org. 2004.
[21] D. Wagner, B. Schneier, Analysis of the SSL 3.0 protocol, in: Proceedings of the 2nd USENIX Workshop on Electronic Commerce. Oakland, California, USA, 1996.
[22] L. Zhao, R. Iyer, S. Makineni, L. Bhuyan, Anatomy and performance of SSL processing, in: Proceedings of the International Symposium on Performance Analysis of Systems and Software, ISPASS'2005. Austin, Texas, USA, 2005.

Daniel F. García is a Professor at the Informatics Department of the University of Oviedo, Spain. Since 1994 he has been leading the computer engineering area at the University of Oviedo. His current research interest is in the area of computer performance engineering and quality of service of computer systems. For the last ten years, Dr. Garcia has been conducting research projects in the area of information technologies at national and European levels. He is a member of ACM and the IEEE Computer Society.

Rodrigo García received the BS degree in Computer Science from the University of Oviedo, Spain, in 2003. He is currently finishing his final course project for the MS degree in computer engineering in the computer engineering area at the University of Oviedo. His research interests are in the areas of computer performance engineering and distributed and Internet computing systems.

Joaquín Entrialgo is an associate professor in the Department of Computer Science and Engineering at the University of Oviedo, Spain. He received the MS degree in Computer Science in 1998 and the Ph.D. degree in 2006. His main research interests are monitoring of real-time systems and performance evaluation of computer systems.

Javier García is an associate professor in the Department of Computer Science and Engineering at Oviedo University, Spain, where he teaches courses in computer architecture. He received a Ph.D. in electrical engineering from Oviedo University. His research interest is in the performance evaluation of computer systems.

Manuel García received the industrial engineering degree and Ph.D. degree with a thesis devoted to the analysis and modeling of HFC networks from the University of Oviedo, Spain, in 1992 and 2003, respectively. From 1993 to 1994 he worked at the university with a grant. In 1994 he was an Assistant Professor and, in 2001, he became an associate professor, in the Department of Computer Science and Engineering at Oviedo University. He has taken part in research projects on real-time inspection systems based on vision techniques, and performance analysis of HFC networks. His current research interests are computer and telecommunications systems performance evaluation.