# Secure Socket Layer (SSL) Impact on Web Server Performance

Mohammed A. Alnatheer

***Abstract*—There are always tradeoffs between security and performance, and security protocols are no exception. This report specifically focuses on one of the most common security protocols, Secure Sockets Layer (SSL), and its effect on the web server performance. Secure Socket Layer (SSL) is the most popular protocol used in the Internet for facilitating secure communications through authentication, encryption, and decryption. Although the use of SSL provides adequate security, it might lead to performance degradations compared to non-secure protocols. In this paper, we analyze the performance and impact of SSL on the web server in terms of the percentage of total processing time, memory available (in bytes), network interface bytes total per second and average disk queue length. The experimental study is based on Intel Pentium IV Windows 2003 operating systems. The major conclusions of this paper are first, SSL increases *the* percentage of total processing *time, second, it decreases the memory* available bytes, and third, it has not impacted the network interface bytes total per second as well as physical disk average disk queue length. SSL has more significant impact on the percentage of total processing time than the impact on the memory available bytes.**

***Index Terms*—Average disk queue length, bytes total/sec, memory available bytes, SSL, server performance, total processor time, web server, workloads.**

## I. Introduction

Local Area Network (LAN) was used to conduct our experiment. It is well known SSL consumes too much time encrypting and decrypting to protect the exchange of the privacy key. Also, the handshake protocol consumes significant resources by exchanging information. Depending upon the performance aspects, SSL could cause a major, impact on the windows web server performance. For example, SSL decreases some software performance counters such as CPU utilization, link utilization, packets per second, memory usage, etc. On the other hand, SSL might not affect other counters such as physical disk, page fault per second. Therefore, we will experiment with multiple counters to study and measure SSL impacts on the Windows web server performance with different counters such as processors, memory, network and physical disk. In the next section we will discuss the related work to our project. In the following section, we will discuss the overview of SSL, following by how SSL secures a transaction, and SSL task outline. Then, we will discuss experiment setup, performance measurements, and performance counters. Afterwards, we will explain our results. In the last section, we will discuss our conclusion and future work.

Manuscript received January 15, 2014; revised March 26, 2014.

Mohammed A. Alnatheer is with King Abdul-Aziz City for Science and Technology, Saudi Arabia (e-mail: mnatheer@kacst.edu.sa).

## II. Overview of SSL

The original design of the Internet was based on a noncommercial environment that included research and education. This, of course, was considered a trustworthy environment in which to operate. However, Internet usage has proliferated in the commercial environment. This now makes security considerations of paramount importance to the buyers, merchants, and other users.

Secure socket layer (SSL) is the most popular protocol used in the Internet for facilitating secure communication through authentication, encryption, and decryption. Although the use of SSL provides adequate security, the performance degradation can be drastically compared to non-secured data retrieval [1].

It is estimated that 10-25% of e-commerce transactions are aborted because of unduly long client response times. These aborted uses basically translate into 1.9 billion dollars of lost revenue [2]. According to Zona Research, users are willing to wait about eight seconds for a page to load. After eight seconds, customers go somewhere else, and they think twice about coming back to the same site. Of those users who leave a site because of a bad experience, 42% never go back (Forrester Research) [3].

SSL can consume significant amount of processing resources [4]. Some researchers have concluded in their research that SSL multiplies the computational cost of the transactions by a factor of 5-7 [1].

Netscape developed SSL for transmitting private documents via the Internet. SSL works by using a private key to encrypt the data that is transferred over the SSL connection. Netscape Navigator and Internet Explorer have supported SSL for obtaining confidential user information, such as credit card numbers. So, by convention, URLs that require an SSL connection start with *https:* instead of *http* [5].

SSL protocol establishes secure connections between clients and server applications, which, in tum, provide authentication of one or both end points of the communication session. Also, SSL provides privacy and integrity of the data that client and server applications exchange. Despite the fact that Netscape Communications Corporation originally developed SSL for securing web browser and server communications, SSL was designed in such a way that other applications, such as TELNET and FTP, could also be enabled to use SSL [6]. Netscape could have incorporated this encryption directly into its browser application, but this would not have provided a unified solution that non-HTTP applications could also use. Therefore, an application of an

DOI: 10.7763/JACN.2014.V2.114

independent form of security was essential. Consequently, Netscape developed SSL to sit on top of TCP (Transmission Control Protocol), thus providing a TCP-like interface to upper-layer applications. In theory, application developers could take advantage of the new layer by replacing all traditional TCP socket calls with the new SSL calls [7].

SSL uses *TCP/IP* on behalf of the higher-level protocols and, in the process, allows an SSL-enabled server to authenticate itself to an SSL-enabled client. This enables the client to authenticate himself to the server, thus permitting both machines to establish an encrypted connection. SSL capabilities address fundamental concerns about communication over the Internet and other *TCP/IP* networks. A few of the most basic concerns are SSL server authentication and SSL client authentication [4].

SSL requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thus providing a high degree of confidentiality. Confidentiality is important for both parties to trust the security of any private transaction. In addition, all data sent over an encrypted SSL connection is protected with a mechanism for detecting tampering. This translates into the automatic determination of whether or not the data has been altered in transit [4].

SSL is available in two strengths, 40-bit and 128-bit. This refers to the length of the "session key" generated by every encrypted transaction. The longer the key, the more difficult it is to break the encryption code. Most browsers support 40-bit SSL sessions. The latest browsers, including Netscape Communicator 4.0, enable users to encrypt transactions in 128-bit sessions-trillions of times stronger than 40-bit sessions. Global companies that require international transactions over the web can use global server certificates program to offer strong encryption to their customers [8]-[10]. SSL implements RSA, a public-key cryptosystem for encryption and authentication. The server then performs a mathematics operation $C=M^e$ mod N, where C is the encrypted message, M is the plain message, e is the public key, and N is the modulus. M, N, and C are 1024-bit numbers, and e is a small number. To decrypt the message sent by the client, the web server uses its private key and performs the operation $M=C^d$ mod N, where d is the server's private key and is typically 1,024 bits long. RSA encryption requires 1024 bit numbers, but CPUs web server operates on 64-bit numbers. Therefore, CPU must divide 1024-bit data into 32-bit or 64-bit pieces. Many CPUs do not have dividers on chips and depend upon software to perform division operations. As a result, encryption and decryption require a significant amount of CPU processing power. The server CPUs bog down when confronted with SSL private-key numbers and the arithmetic operations used in RSA algorithms, resulting in poor performance [11].

## III. How SSL Secure a Transaction

SSL consists of two phases: handshake and data transfer. Both client and server use a public key encryption algorithm to determine secret-key parameters during the handshake phase. In the transfer phase, however, clients and server use secret key to encrypt and decrypt successive data transmissions [7].

The client starts initiating the SSL handshake connection by first transmitting a Hello message. The hello message contains a list of the secret-key algorithms, called cipher specs, which the client supports. On the other side of the connection, the server responds with a similar Hello message, selecting its own preferred cipher spec [7].

Following the Hello message, the server sends a certificate that contains its public key. A certificate is simply a set of data that validates the server's identity. The certificate contains information such as server's ID, its public key, and several other parameters. Certificate authority generates (CA) generates the certificate and verifies its authenticity. In order to obtain a certificate, a server must use secure channels to send its public key to a CA. After the CA generates the certificate, which contains its own ID, the server's ID, and the server's public key and other information, it uses a message digest algorithm to create a certificate fingerprint. The message digest takes a given stream of data and produces a deterministic, fixed-length output. Then, CA encrypts fingerprint with its private key to create the certificate signature. In order to validate the server's certificate, the CA's public key must first decipher the signature and read the pre-calculated fingerprint. Then the client independently computes the certificate's fingerprint. If the two fingerprints don't match, the certificate has been tampered with [7].

The client should maintain a list of trusted CAs and their public keys. This will ensure that whenever the client receives a server's certificate, it verifies that the CA signing the certificate belongs to its list of trusted CAs. As soon as the client authenticated the server, both of them can use public-key algorithm to determine secret-key information. In order to complete the handshake phase with finished messages, and the connection enters the data transfer phase, both client and server need to indicate their readiness to begin using the secret key [7].

During the data transfer phase, client and server break up their outgoing messages into fragments and append to them message authentication codes (MACs). The MAC an encrypted message digest (fingerprint) computed from the message contents. When transmitting data, the client or server combines the data fragment and a record header encrypts them with the secret key to produce the completed SSL packet. When receiving data, the client or server decrypts the packet, computes the MAC, and compares the computed MAC to the received MAC [7].

Another SSL interesting feature is session resumption. When SSL was originally designed, the designers knew the fact that public-key algorithms were computationally expensive. Client always made several new connections with the same server in a short period of time. These connections would heavily burden the server, and thus the client would experience a noticeable delay in response time. On the other hand, if the client and the server established a unique session ID at the beginning of a new session, any successive connections within a given time frame could simply refer to that ID and use the same secret key [7].

## IV. SSL Task Outline

There are three categories that an SSL entity must be able to perform: handshake processing, bulk-data operations, and administration. Compared to the other tasks, handshake processing consumes most the power computing, followed by bulk-data encryption and decryption, and then administration [7].

### A. *Handshake* Processing

Handshake processing breaks down into several distinct categories.

#### 1) *Message exchange*

Both the client and the server must recognize the defined messages, such as Hello and Finished [7].

#### 2) *Public-key computations*

The client must use public-key algorithms to either distribute or generate secret-key parameters. Moreover, the client must use a CA's public key to validate a server's certificate. On the other end, the server must use public-key algorithms to either decrypt or generate secret-key parameters. If the server requires client authentication, it must also use a CA's public key to validate the client's certificate [7].

#### 3) *Random-number generation*

Both the client and server might need to generate random numbers to reduce the predictability of secret keys depending on the type of public-key encryption algorithm used [7].

#### 4) *Handshake authentication*

Both the client and server generate a message digest based on, among other inputs, all messages transmitted during the handshake phase. Then, each side encrypts and sends this message digest as part of the finished message. The recipient of the finished message must verify the message digest's authenticity by performing an independent message digest calculation that includes all previously received handshake messages. If the two message digests do not match, one or more of the handshake messages has been tampered with [7].

### B. *Bulk-Data Operations*

Bulk-data operations are performed in actual data that must be securely transferred between the client and server [7].

#### 1) *Encryption/decryption*

During data transfer phase, both the client and server use the secret key to encrypt and decrypt data [7].

#### 2) *Message authentication*

For every SSL data record transmitted, the sender must calculate and adds a MAC. For every SSL data record received, the receipt must verify the MAC [7].

### C. *Administration*

Administrative components require the least amount of computation [7].

#### 1) *Certificate and key maintenance*

If the client plans to access a site that requires client authentication, it must maintain its certificate and the associated private key. If not, the client does not need to maintain a certificate. The server must always maintain its certificate and the associated private key [7].

#### 2) *Session ID storage:*

Both the client and server must maintain a cache of session IDs and associated secret keys to use during a session resumption handshake [7].

Because a single web server services multiple Web clients, the SSL server entity causes much more of a bottleneck during SSL transactions than the client.

## V. Related Work

There is significant related work to our project. In Sandhu [12], the authors discuss analysis of the performance and security in e-commerce applications. Also, the authors describe their findings after measuring the impact of Security Sockets Layer (SSL) protocol on the request response time. The results for the overhead secure connections case show that use of secure connections increased the client response time from 0.1 to 6 seconds on average. In addition, the costs of receiving and sending encrypted data is about 40 % higher than the costs for handling raw data. Thus, while requests that transmit more data are more affected by secure connections, the authors who compared apache modules to CGI determined that the use of modules did not improve the performance of the server significantly. In J. Almeida, Almeida, Yates [1], the authors discuss the analysis and performance impact of SSL on the servers in terms of various parameters. These parameters include throughput, utilization, ache sizes, cache miss ratios, number of processors, control dependencies, file access sizes, bus transactions, and network load, among other various components. 'The authors conclude that processors with higher core frequency will improve the SSL performance, In addition, a processor with high pipeline depth can improve the performance of SSL transactions, whereas the increase in the issue width may not provide any significant performance improvement, especially in cases where the performance is dominated by bulk data encryption. However, increasing the size of L1 cache will have a positive impact on the SSL performance. The frequencies of branches are low, and the efficiency of BTB is also low. Thus a complex logic and large BTB for branch handling will not be beneficial for SSL transactions. Increasing the size of L2 caches to any reasonable extent will have minimal impact on the performance of SSL transactions. SSL handshake and encryption/decryption of large web-pages has very good scaling with respect to the number of processors in s SMP system, which may promote the use of 4-way or 8-ways systems for these applications. In SSL Accelerator [13], the author examines the maximum throughput for Secure Sockets Layer for various combinations of cryptographic algorithms and key lengths. The authors concludes that 1,280 bits public key should be sufficient to protect against attacks from individuals, but a 1,536 bit key will be needed to protect from attacks originating in large corporations. In addition, a 2,048 bit key would be the best protection against attacks from the government. In Securing Application with SSL [14], the authors study the tradeoffs between security and performance in e commerce applications. Their results show that SSL handshake is expensive and the effect of other security

measures is dependent on the specific architecture of the business application. In *System Monitor* [15], the authors develop a tool for evaluating and understanding server performance and present new results for realistic workloads. Their results shows that in a web server saturated by client requests, up to 90 % of the time spent handling HTTP requests is spent in the kernel.

## VI. Experiment Setup Description

### A. Server Setup

The web server on which we have chosen to run our experiment is Internet Information Service 6.0 (lIS 6.0). lIS is a powerful web server and provides highly reliable, manageable, and scalable Web application infrastructure for all versions of Windows Server 2003. lIS helps organizations increase website and application availability while lowering system administration costs [16].

lIS 6.0 supports the Microsoft Dynamic Systems Initiative (DS1) with automated health monitoring, process isolation, and improved management capabilities and provides the services to support a secure, available, and scalable web server on which to run your websites and applications.

### B. Using SSL to Encrypt Confidential Data

We can use SSL security features on our web server to encrypt network transmissions, which will help ensure the integrity of data transmission, as well as to verify the identity of users. We can configure SSL to help protect confidential data on a URL-by-URL basis. One portion of the application might require encryption of data transmissions with SSL, while another portion of the application might allow unencrypted data transmission (by specifying HTTP in the URL). This flexibility in security configuration allows us to provide encryption of confidential data as required, unlike IPSec and VPNs because they require that you encrypt all traffic between the clients and the web server. Our web server, for example, contains both secure and insecure content. The URL for the unsecured home page is

http://server.alnatheer.com. The URL for the secured e-commerce portion of the web site is https://server.alnatheer.com.

The client is simply a web browser making the request to the web server. The client runs on Windows XP operating systems, and it has 2.4 GHZ with Pentium IV Processor. The *server has a specification of 2.79 GHZ, Pentium IV, and a Windows 2003 operating* system. Both the client and the server are covered by Local Area Network (LAN). LAN internet connection speed is 100 Mb per second.

The workload of the experiment varies from 3MB to 250 MB. Due to the fact that it would be relatively difficult to measure the Secure Socket Layer impact on small workloads, we have decided to start with a relatively large workload such as 3 MB to start our first workload of the experiment. For smaller workloads, more complex tools would be needed to obtain more accurate results. Our tool is basically a Windows 2003 performance tool which is part of the Windows 2003 operating system.

### C. Performance Measurement and Performance Counters

The System Monitor, a tool for data collection/monitoring, is the most powerful monitoring tool and can be used in multiple ways for tracking system performance and determining how *to* optimize server functions. The System Monitor is like a window into the inner workings of just about every aspect of the system, such as hard disks, memory, the processor, the page file, etc. The Performance Monitor monitors system components, which are called objects (e.g. Processor, Memory), and there can be multiple instances of the same object. A counter is an indicator of a quantity of the object that can be measured in some unit, such as percent, rate per second, or peak value, depending on what is appropriate to the object. The System Monitor offers three modes of monitoring: chart, histogram and report. A chart is a running line graph of the object that shows distinct peaks and valleys. A histogram is a running bar chart that shows each object as a bar in a different color. The report mode simply provides numbers on a screen which you can capture to put in a report. The System Monitor can be used to monitor not only the local computer, but other computers on the network [17].

In this case, we have concentrated in the following counters [18]:

1) Percentage of Total Processor Time: percentage of processor time shows the percentage of elapsed time that this thread used the processor to execute instructions. An instruction is the basic unit of execution in a processor, and a thread is the object that executes instructions. Code executed to handle some hardware interrupts and trap conditions is included in this count.
2) Memory Available Bytes: Shows the amount of physical memory, in bytes, immediately available for allocation to a process or for system use. It is equal to the sum of memory assigned to the standby (cached), free, and zero page lists.
3) Network Interface Bytes Total/Sec: Shows the rate, in incidents per second, at which bytes were sent and received on the network interface, including framing characters. Bytes Total/sec is the sum of the values of Bytes Received/sec and Bytes Sent/sec.
4) Avg. Disk Queue Length: Shows the average number of both read and writes requests that were queued for the selected disk during the sample interval.

## VII. Experiment Results

In this experiment, we analyze the performance and impact of SSL on the web server in terms of the percentage of total processing time, memory available (in bytes), bytes total per second and average disk queue length. Therefore, we need to use statistical analysis such as mean, variance, and coefficient of variance to verify whether SSL has impacted the web server performance. We will discuss the conclusion in detail later in the report.

### A. Counter 1: Percentage of Total Processing Time

Table I shows the difference in values of SSL and Without-SLL, in reference to the total processing time. Table I illustrates the exponential increase of total processing time

in relation to larger workloads.

TABLE I: PERCENTAGE OF TOTAL PROCESSOR TIME COMPARISON

| Workloads | W/O SSL | SSL | % Increases |
|---|---|---|---|
| 3MB | 0.161 | 0.169 | 5 |
| 10MB | O.I93 | 0.298 | 54.16 |
| 13MB | 0.241 | 0.306 | 26.67 |
| 24MB | 0.249 | 0.322 | 29.03 |
| 30MB | 0.265 | 0.386 | 45.54 |
| 73MB | 0.418 | 1.047 | 150 |
| 167MB | 0.918 | 1.578 | 71.92 |
| 250MB | 1.328 | 2.81 | 111.52 |

Table II shows the difference in Variance values between the two cases. It is clear that variance values for the Without-SSL are smaller than SSL. This table illustrates increases in the workload will result in larger variances of SSL than Without-SSL case.

TABLE II: THE VARIANCE (% TOTAL PROCESSOR TIME)

| Workload | *W /O* SSL | SSL |
|---|---|---|
| 3MB | 0.278 | 0.384 |
| 10MB | 0.458 | 1.739 |
| 13MB | 1.034 | 0.909 |
| 24MB | 1.176 | 2.374 |
| 30MB | 0.850 | 2.646 |
| 73MB | 1.424 | 14.736 |
| 167MB | 2.784 | 18.106 |
| 250MB | 4.605 | 28.11 |

Table III shows the difference in Coefficient of Variance values in SSL and Without-SSL. Coefficient of variance allows us to compare the two cases and clearly shows SSL as having the larger value of coefficient of variance than Without-SSL.

TABLE III: COEFFICIENT VARIANCE (% TOTAL PROCESSOR TIME)

| Workload | *W /O* SSL | SSL |
|---|---|---|
| 3MB | 3.278 | 3.665 |
| 10MB | 3.501 | 4.425 |
| 13MB | 4.209 | 3.116 |
| 24MB | 4.344 | 4.783 |
| 30MB | 3.469 | 4.207 |
| 73MB | 2.850 | 3.666 |
| 167MB | 1.817 | 2.695 |
| 250MB | 1.614 | 1.886 |

The increase in total processing time is a result of decrypting a message that was encrypted with a public-key algorithm is quite CPU-intensive. Additionally, since CPUs web server operates on 64-bit numbers, RSA encryption requires 1024 bit numbers. CPU must then divide 1024-bit data into 32-bit or 64-bit pieces. Many CPUs do not have dividers on chips and depend upon software to perform division operations. Therefore, encryption and decryption require significant numbers of CPU processing power. The server CPUs bog down when confronted with SSL private-key numbers and the arithmetic operations used in RSA algorithms, combining to result in poor performance. Finally, SSL handshakes are performance-intensive operations because of the cryptographic operations using the public and private keys. Handshake processing takes up a lot of CPU time.

### *B. Counter 2: Memory Available Bytes*

Table IV shows the difference in values of SSL and Without-SSL in reference to the Memory available bytes. SSL decreases the memory available bytes by less than 1 %. The SSL impact on memory available bytes does not have any connection with the size of the workloads.

TABLE IV: MEMORY AVAILABLE BYTES COMPARISON

| Workload | W /O SSL | SSL | % Decreases |
|---|---|---|---|
| 3MB | 232081851 | 232778002 | 0.30 |
| 10MB | 232006688 | 232254601 | 0.10 |
| 13MB | 232191514 | 232474730 | 0.12 |
| 24MB | 233111214 | 234657137 | 0.658 |
| 30MB | 232432587 | 232648366 | 0.092 |
| 73MB | 228904833 | 229231837 | 0.142 |
| 167MB | 228774775 | 229165795 | 0.17 |
| 250MB | 229240832 | 229481187 | 0.104 |

Table V shows the difference in variance in SSL and Without-SSL. Table V shows that there are no identifiable trends between SSL and Without-SSL in relation to variance performance comparison.

TABLE V: THE VARIANCE MEMORY (AVAILABLE BYTES)

| Workload | *W /O* SSL | SSL |
|---|---|---|
| 3MB | 1737807531 | 4532839104 |
| 10MB | 10170567424 | 2379316235 |
| 13MB | 1260892821 | 3651523584 |
| 24MB | 7121539029 | 1802386837 |
| 30MB | 7081303979 | 4049135680 |
| 73MB | 11506247851 | 1827127467 |
| 167MB | 1647154347 | 3949366016 |
| 250MB | 5.09E+09 | 4800380928 |

Table VI shows Coefficient of Variance in SSL and Without-SSL. Although the difference in the total memory available bytes is less than 1 % as we have seen in the total processing time, SSL still impacts the memory available bytes.

TABLE VI: COEFFICIENT OF VARIANCE MEMORY (AVAILABLE BYTES)

| Workload | *W /O* SSL | SSL |
|---|---|---|
| 3MB | 1.79E-04 | 2.90E-04 |
| 10MB | 4.34E-04 | 2.10E-04 |
| 13MB | 1.53E-04 | 2.60E-04 |
| 24MB | 3.60E-04 | 1.82E-04 |
| 30MB | 3.62E-04 | 2.74E-04 |
| 73MB | 4.68E-04 | 1.87E-04 |
| 167MB | 1.77E-04 | 2.75E-04 |
| 250MB | 3.1IE-04 | 3.02E-04 |

### *C. Counter 3: Network Interface Bytes Total/Sec*

Table VII shows the difference in values between SSL and Without-SSL. According to Table VII, SSL decreases the Bytes Total/Sec by less than 0.3%. There is a slight change of total bytes/sec., although this change is not significant enough to conclude any changes of Bytes Total/sec. Therefore, SSL does not have an impact on Total Bytes/Sec.

TABLE VII: (NETWORK INTERFACE) (BYTES TOTAL/SEC) PERFORMANCE COMPARISON

| Workload | W /O SSL | SSL | % Decreases |
|---|---|---|---|
| 3MB | 38754.68 | 38745.69 | 0.0232 |
| 10MB | 109407.4 | 109174.9 | 0.2125 |
| 13MB | 139521.5 | 139153.3 | 0.2639 |
| 24MB | 259726.1 | 259069.6 | 0.252 |
| 30MB | 319565.8 | 318587.4 | 0.306 |
| 73MB | 775769.7 | 773707.5 | 0.265 |
| 167MB | 1779364 | 1774814 | 0.255 |
| 250MB | 2659521 | 2654760 | 0.179 |

Table VIII shows the difference in variance for the Network Interface Bytes Total/Sec in SSL and Without-SSL. The large variance occurs during the first few seconds when we transfer the file from client to server.

TABLE VIII: THE VARIANCE NETWORK INTERFACE (BYTES TOTAL/SEC)

| Workload | *W /O* SSL | SSL |
|---|---|---|
| 3MB | 1.11E +11 | 1.35E +11 |
| 10MB | 8.61E +11 | 1.03E +12 |
| 13MB | 1.21E +12 | 1.19E +12 |
| 24MB | 2.39E +12 | 2.32E +12 |
| 30MB | 3.33E +12 | 2.93E +12 |
| 73MB | 6.93E +12 | 7.16E +12 |
| 167MB | 1.25E +13 | 1.57E +13 |
| 250MB | 1.56E +13 | 1.75E +13 |

Table IX shows the Coefficient of Variance in SSL and Without-SSL. SSL has greater coefficient variance than of Without-SSL.

TABLE IX: COEFFICIENT OF VARIANCE NETWORK INTERFACE (BYTES TOTAL/SEC)

| Workload | *W /O* SSL | SSL |
|---|---|---|
| 3MB | 8.60 | 9.48 |
| 10MB | 8.48 | 9.30 |
| 13MB | 7.88 | 7.84 |
| 24MB | 5.95 | 5.88 |
| 30MB | 5.71 | 5.37 |
| 73MB | 3.39 | 3.46 |
| 167MB | 1.99 | 2.23 |
| 250MB | 1.49 | 1.58 |

SSL does not have an impact on Total Bytes/Sec. According to Table VII, SSL impact on the Bytes Total/sec is less than 0.30%. The Bytes Total/Sec is the sum of the values of Bytes Received/sec and Bytes Sent/sec. SSL does not affect the number of packets sent or received from client to server. As we mentioned earlier, SSL mainly affects CPU processing time because the encryption and decryption process takes up a lot of processing time.

### *D. Counter 4: Physical Disk Average Disk Queue Length*

Table X shows a difference in values between SSL and Without-SSL for the Physical Disk (Average Disk Queue length). Table X shows that SSL does not impact Average Disk Queue Length.

TABLE X: PHYSICAL DISK (AVERAGE QUEUE LENGTH) PERFORMANCE COMPARISON

| Workload | W /O SSL | SSL | % Decreases |
|---|---|---|---|
| 3MB | 0.0011 | 0.0010 | 10.32 |
| 10MB | 0.002087 | 0.002007 | 3.83 |
| 13MB | 0.002557 | 0.002522 | 1.368 |
| 24MB | 0.004 | 0.004 | 1.08 |
| 30MB | 0.004 | 0.005 | 10.86 |
| 73MB | 0.011 | 0.014 | -28 .03 |
| 167MB | 0.028 | 0.024 | -13.06 |
| 250MB | 0.188 | 0.186 | 1.0456 |

Table XI shows the difference in variance for the Physical Disk (Average Disk Queue Length) between SSL and Without-SSL. The variance value in this counter was smaller than the previous counters due to the fact that the mean values are relatively small compared to the other counters.

TABLE XI: THE VARIANCE PHYSICAL DISK (AVERAGE DISK QUEUE LENGTH)

| Workload | *W /O* SSL | SSL |
|---|---|---|
| 3MB | 4.18E-05 | 5.70E-05 |
| 10MB | 0.0002 | 0.0002 |
| 13MB | 0.0002 | 0.0002 |
| 24MB | 0.0007 | 0.0006 |
| 30MB | 0.0006 | 0.0007 |
| 73MB | 0.0013 | 0.0027 |
| 167MB | 0.0033 | 0.0029 |
| 250MB | 0.3048 | 0.2911 |

Table XII shows the coefficient of Variance Physical Disk (Average Disk Queue Length) between SSL and without SSL.

TABLE XII: THE VARIANCE OF COEFFICIENT OF PHYSICAL DISK (AVERAGE DISK QUEUE LENGTH)

| Workload | *W /O* SSL | SSL |
|---|---|---|
| 3MB | 5.61 | 7.31 |
| 10MB | 6.98 | 8.06 |
| 13MB | 6.83 | 6.76 |
| 24MB | 5.63 | 5.38 |
| 30MB | 5.14 | 5.13 |
| 73MB | 3.32 | 3.65 |
| 167MB | 2.04 | 2.20 |
| 250MB | 2.93 | 2.89 |

Our findings demonstrate that in the first counter, SSL increases the total processing time proportionally to the increase in workload. In the second counter, SSL still impacts the memory available bytes, but the result is less significant compared to the total processing time. In the third and fourth counters, we have not noticed any noteworthy changes in total bytes per second and the average disk queue length.

## VIII. CONCLUSION AND FUTURE WORK

In the experiments express in this paper, our main objective is to study the SSL impact on a web server running Windows 2003 OS since SSL degrades the server performance compared to Without-SSL. Based on the collected data of the

tools provided with the Windows 2003 operating system, we have concluded the following:

- SSL has a significant impact on the percentage of total processing time with larger workloads such as 73MB or higher.
- SSL has a measurable impact on memory available bytes with all the workloads
- SSL does not have any impact on the network interface bytes total/sec because SSL does not affect the number of packets sent or received from client to server.
- SSL does not have impact on the physical disk Average Disk Queue Length.

SSL is very computational intensive. The increase in total processing time, a result of decrypting a message that was encrypted with a public-key algorithm, is quite CPU intensive. Furthermore, SSL handshakes are performance-intensive operations because of the cryptographic operations using the public and private keys. So, Handshake processing takes up a lot of CPU time. The aforementioned are the most influential reasons for increasing the percentage of the total processing time.

In Web Media SSL terms [19], the authors propose solutions for performance degradation. For example, the adapter off-loads the RSA decryption function as part of the key exchange and generation. By off-loading these functions from the host CPU, the accelerator adapter speeds up the secure Web transactions needed to protect personal and proprietary information during e-business transactions [11]. Therefore, SSL adapter cards help offload the server's CPU load and increase server performance [19]. Another option is to use the Broadcom SSL 800 accelerator to enhance performance [11]. Broadcom's SSL solutions significantly outperform the fastest general-purpose server CPUs on a 1024-bit private-key operation when running at a fraction of the frequency of the CPU [11]. The findings of these experiments can be extended to investigate other performance counters in processor such as thread processor time and thread user time. As in any performance analysis, we should also identify the performance bottlenecks devices. Another approach for this experiment is to use multiple clients performing multiple requests instead of one client performing one request. We can further our study to investigate different workload sizes and their effects on the server performance. Finally, we can look into real live applications of SSL. For example, we can design e-commerce sites and test the performance from different clients as well as different workloads.

## ACKNOWLEDGMENT

The authors wish to thank King Abdul-Aziz City for Science and Technology for funding this project.

## REFERENCES

[1] J. Almeida, V. Almeida, and D. Yates, "Measuring the behavior of a world-wide web server, high performance networking VII," in *Proc. the 7th Conference on High Performance Networking*, White Plains, NY, April 1997.
[2] W. Chou, "Inside SSL: accelerating secure transactions," *IEEE IT Pro*. September/October 2002, pp. 37-41.
[3] W. Chou, "Inside SSL: The secure sockets layer protocol," *IEEE IT*. July/August 2002, pp. 47-52.
[4] C. Pfleeger and S. Pfleeger, *Security in Computing*, Upper Saddle River, Prentice Real, 2003.
[5] Introduction to SSL. [Online]. Available: http://wp .netscape.c01l1/security/techbriefs/ssl.html
[6] Introduction to SSL [Online]. Available: http://developer.netscape.c01l1/docs/manuals/security/sslin/contents. ht1l1
[7] K. Kant, R. Iyer, and P. Mohapatra, "Architectural impact of secure socket layer on internet servers," in *Proc. International Conference on Computer Design*, Sept. 2000, pp. 7-14.
[8] R. Kinicki *et al*., "Electronic commerce performance study," in *Proc. the Euromedia* '98, Leicester, United Kingdom, January 1998.
[9] D. Menasce, "Security Performance," *IEEE Internet Computing*, May/June 2003.
[10] G. Paixao, W. Meira Jr., V. Almeida, D. Menasce, and A. Pereira, "Design and implementation of a tool for measuring the performance of complex e-commerce sites," in *Proc. Tools 2000 Conference*, Chicago, IL, March 2000.
[11] Performance counters reference for windows server 2003. [Online]. Available: http://www.lllicrosoft.com/rcsourccs/doculTIcntation/ WindowsServ/2003/all/dcployguide/enus/Dcfaull.asp?url=/rcsourccs/ doculTIcntation/WindowsScrv/2003/all/dcplovguidc/cn-us/counters1j kwd.asp
[12] R. Sandhu, "Good enough security," *IEEE Conference on Internet Computing*, Jan/Feb 2003, pp. 84-87.
[13] SSL Accelerator. [Online]. Available: http://www.mcdiawcb.com.sg/sonicwall/white1020papcr/SonicWAL L SSL WP.pdf.
[14] Securing Application with SSL. [Online]. Available: http://publib.boulder.ibm.comlhtml/as400/v5rl/ic2931/index.htm'!inf o/rzai n/rzainovcrvicw.htrn
[15] System Monitor. [Online]. Available: http://www.research.umbc.edu/-dgorini/45IM/lecturenotes.htm>
[16] T. Wilson. (May 20, 1999). E-biz bucks lost under SSL strain. Internet Week Online. [Online]. Available: www.inlcrnetwk.com/lead/lead052099
[17] Understanding SSL gigabit ethernet accelerator adapters in power edge servers. [Online]. Available: http://wwwl.us.dell.com/content/topics/global.aspx/power/en/ps4qOI _broadcom?c=us &l=en&s=gen
[18] Windows Server 2003 deployment kit. [Online]. Available: http://www.microsoft.com/downloads/dctails.aspx.?Familvld=F31A5 FD5-03DB46D2-9F34-59CJ ED0039EB9&displaylang=cn
[19] Web media SSL terms. [Online]. Available: http://www.webopedia.com/TERM/S/S L.html

**Mohammed Alnatheer** was born in the city of Arar, Saudi Arabia in July 1978. He holds a bachelor of science in electrical and computer engineering in 2003 and master of science in computer science in 2004 from West Virginia University, WV, USA. Mohammed Alnatheer also holds a PhD degree in information technology in 2013 from Queensland University of Technology, Brisbane, Australia. Currently, he is an assistant research professor at King Abdulaziz City for Science and Technology in Riyadh, Saudi Arabia. Mohammed Alnatheer is conducting researches in research areas related to information security and software engineering.