A COMPARISON OF HTTP AND HTTPS PERFORMANCE [1]

Arthur Goldberg, Robert Buff, Andrew Schmitt
Computer Science Department
Courant Institute of Mathematical Science
New York University
{artg, buff, schm7136}@cs.nyu.edu
www.cs.nyu.edu/cs/faculty/artg

This study compares the performance of encrypted and non-encrypted Web communications. It presents measurements of the most widely used secure Web encryption—the Secure HyperText Transport Protocol (HTTPS) on the Secure Sockets Layer (SSL) version 3.0 with 40 and 128 bit RC4 encryption. Encryption increases the response times of two popular Web servers, Netscape Enterprise Server 3.5.1 and Microsoft IIS 4.0, by at most 22%. We view this additional delay imposed by encryption as moderate, and therefore encourage Web sites to routinely use secure communications.

1. Introduction

The importance of electronic commerce is widely acknowledged. Surveys of Web users indicate that poor performance is a major cause of dissatisfaction. This paper presents results that show that popular servers that use the Secure HyperText Transport Protocol (HTTPS), the secure version of the HyperText Transport Protocol (HTTP), can transmit typical documents with little performance penalty.

We study both Netscape's Enterprise Server and Microsoft's Internet Information Server (IIS) because these are two of the three most popular servers as indicated by a comprehensive survey [Netcraft].

2. Encrypted Communications

We briefly review the operation of secure Web communications. The Secure Sockets Layer (SSL) [Hickman95, SSL] protocol has become most widely used method for encrypting and authenticating Web communications. Conducting SSL communications involves the following steps:

- A client establishes a Transmission Control Protocol (TCP) connection with a server, which involves one round-trip message delay when no failure occurs.
- On top of TCP, the client and server establish a secure SSL communication channel [Freier96]. The application implements this by issuing an `SSL_connect()` call to the SSL library. The client and server negotiate a mutually agreeable cipher, which is a stream encryption algorithm and an authentication method pair. The client and server use a public key cryptographic protocol to exchange secret session keys that will be used to encrypt and decrypt application layer messages. Secret keys are frequently used for streaming encryption because public key encryption is much more expensive. For more details, see Bolyard's nice trace of SSL session setup [Bolyard97].
- On top of SSL, the client and server exchange one or more HTTP messages. A single HTTP message is exchanged in HTTP/1.0; multiple messages would be exchanged in keep-alive or persistent connections [Fielding98]. To use encryption the application code issues calls to `SSL_write()` and `SSL_read()`, instead of calls to TCP socket `write()` and `read()`, respectively.

The performance of the encryption algorithm RC4 [RC4] was studied. RC4 is a variable key-size stream cipher designed by Ron Rivest. Designed for byte-oriented operations, RC4 is expected to run quickly, as it uses only 8 to 16 instructions per byte.

[1] Supported by an IBM Partnership Award for "Web Performance Measurement and Evaluation", from 1997 to 1998.

We studied the RC4 because it is widely used. A non-statistical survey of 5384 secure Web sites worldwide in March 1997 [Netcraft97] found that 59% of the secure servers were Netscape or Microsoft and used RC4 encryption. The portion for each server and key size is shown in Table 1.

	40 bit	128 bit	Total
Netscape-Enterprise (includes multiple versions)	16.4%	10.4%	26.8%
Microsoft-IIS (includes multiple versions)	20.0%	12.6%	32.6%
Total	36.4%	23.0%	59.4%

Table 1. The Subset of the Secure Servers Which Run Netscape or Microsoft and Use RC4

Secret key sizes of 40 and 128 bits constitute the vast majority of production RC4 keys. The smaller, 40-bit key is called "export strength" because the US Government permits its export from the United States. It is breakable with moderate effort [Netcraft97]. The 128-bit key is considered long enough to be unbreakable by known methods in typical large computer facilities —assuming it is used properly.

3. Experiments

We modeled a small Intranet environment. A 10 Mbps Ethernet connected 2 PCs. Each PC was directly attached to an unswitched hub. The Web servers ran on a PC with a 200 MHz Pentium, 256MB RAM and a fast Ethernet card, running NT 4.0. The clients ran on a PC, also running NT 4.0, with a 100 MHz Pentium with 32MB RAM and an NE 2000 NIC Ethernet card.

Performance was measured in a no-load situation—during the experiments the PCs were otherwise unused and the hub was lightly used. In addition, during the experiments neither machine paged virtual memory.

We call our measurement apparatus WebPerf. WebPerf consists of a Web robot client and a back-end database. The robot measures Web response times and other parameters and stores the results in the database. The robot was written by one of us—Buff—in C++ and compiled with Visual C++ version 5.0, with optimization. It communicates via Winsock 2.0.

To minimize contention with itself the robot browser runs single-threaded on an otherwise idle machine.

A widespread SSL implementation was integrated into the robot by one of us—Schmitt. SSLeay version 0.8.1, written by Eric A. Young [Hudson] was used. It supports SSL versions 2.0 and 3.0. The robot does not authenticate the server since this is a client side activity.

Netscape Enterprise Server 3.5.1 and Microsoft IIS 4.0 were both installed on the server PC. Two identical Web sites were created with 24 documents of sizes 1,000, 2,000, 3,000, …, 20,000, 40,000, 60,000, 80,000, 100,000 bytes. This size distribution is similar to that of documents requested on the Web as observed in traces from an Internet Service Provider and an Intranet in 1998 [Goldberg98B]. The documents contained typical HTML text (actually, the beginning of the HTTP/1.1 specification [Fielding98]).

The robot browser sequentially requested each of the 24 documents many times. The robot measured the *duration* at the client between just before the client robot issued the SSL_write() of the request and just after the client completed the SSL_read() of the response. This measures the streaming application layer encryption performance. The costs of additional delays for establishing SSL connections were presented elsewhere [Goldberg 98A].

This duration and many other measures are stored in an Oracle 7.3 SQL database after the measurements have been made.

4. Analysis

To evaluate the cost of server encryption and client decryption, the performance of HTTPS and HTTP were measured. The data are plotted in Figure 1 and Figure 2.

Each vertical bar represents a set of measurements of HTTP or HTTPS durations. In Figure 1 the bars above 20 KB summarize 87 measurements for the non-secure and 40 bit data and 30 for the 128 bit measurements. The ones below measurements 20 KB summarize 6 times as many data points. In Figure 2 the bars summarize 87 measurements for the non-secure and 40 bit data and 30 for the 128 bit measurements.

The top and bottom of the wide part of a bar marks the range of 75% of the measurements closest to median (the range from 12.5% to 87.5%). The narrow vertical bars mark the range of 95% of the data. A dashed line indicates a least squares fit.

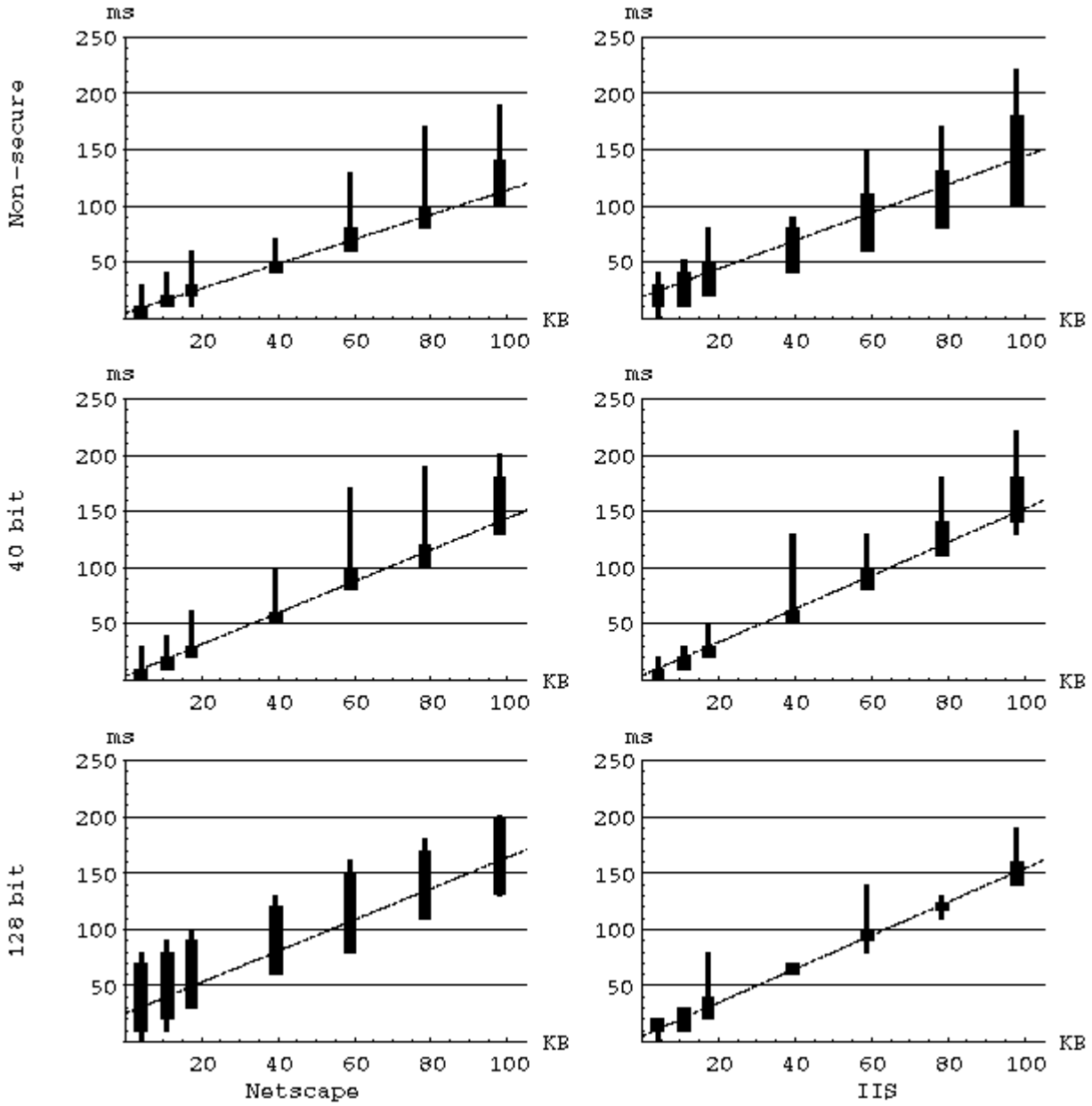

Figure 1. Durations for Non-secure HTTP and Secure HTTPS with Two RC4 Key Sizes for Netscape and Microsoft Web Servers for Documents from 1 to 100 KB. The wide part of a bar marks the range of 75% of the measurements closest to median. The narrow vertical bars mark the range of 95% of the data. A dashed line indicates a least squares fit

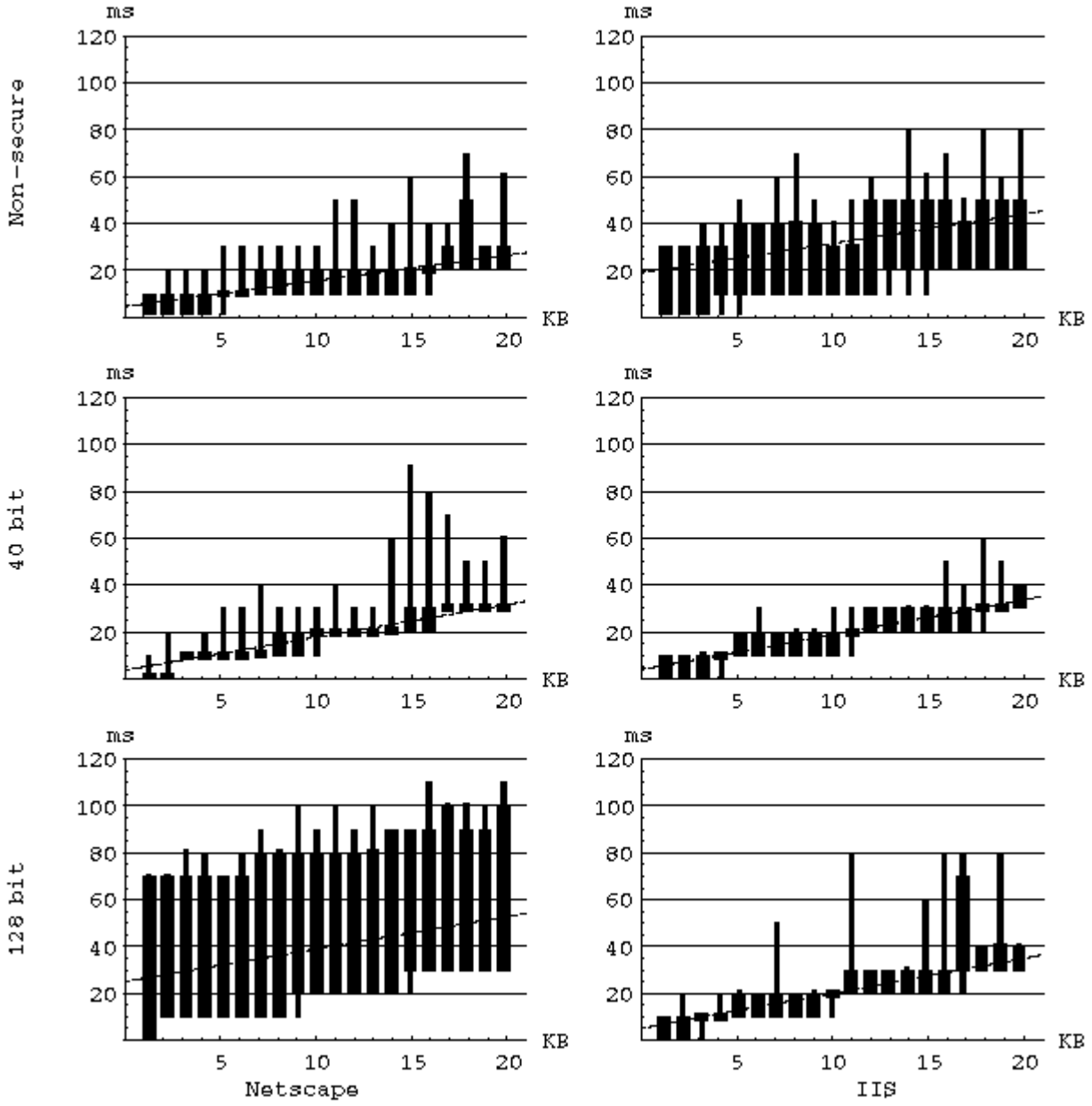

Figure 2. Durations for Non-secure HTTP and HTTPS with Two RC4 Key Sizes for Netscape and Microsoft Web Servers for Documents from 1 to 20 KB. The wide part of a bar marks the range of 75% of the measurements closest to median. The narrow vertical bars mark the range of 95% of the data. A dashed line indicates a least squares fit.

These graphs show, in three rows, measurements of non-secure, and secure RC4 40 bit and 128 bit response times.

Several features are apparent in the data. As expected the duration increases with document size. For most configurations the duration shows little dispersion, that is, 75% of the data falls within a small range.

For many of the document sizes 75% of the data falls near the minimum duration measurement, as seen for example in the Netscape 40 bit configuration. This is consistent with a duration that is determined by the critical path through the code. The 20% of the data that falls between 75% and 95% may result from occasional queuing delays that occur on lightly loaded systems and networks.

However, two configurations, IIS non-secure and Netscape 128 bit, show high variability. For the Netscape 128 bit data, especially for documents less than 20 KB, the variability is similar in size to the data. These configurations show surprisingly large delays for small documents. This variability is apparent in Figure 1 and, especially, in Figure 2. The source of this variability is not understood and is under investigation. These larger values become obvious in the *Offset* of the linear fit, as listed in Table 2.

Each data set was fit with a least squares line parameterized by

Duration = Offset + DocumentSize / TransferRate

The fit parameters are given in Table 2.

	TransferRate (bytes/ms)		*Offset* (ms)	
Server	Netscape	Microsoft	Netscape	Microsoft
Non-secure	946	829	4.5	19.0
Secure 40 bit	730	689	3.6	3.9
Secure 128 bit	736	686	25.0	5.1

Table 2. Parameters of Linear Fits to HTTP and HTTPS Transfers

The *TransferRate*s are consistent with the Ethernet limit of 10 Mbps (1250 bytes/ms). Also, measured TCP transfer rates in good stacks are consistent with our results.

The *TransferRate*s decrease as the level of encryption increases, as expected. However, the *TransferRate*s only decrease by 22% and 17% for Netscape and Microsoft, respectively. We consider this to be a reasonable price to pay for security.

5. Conclusions

We find that secure Web servers perform well in comparison to non-secure servers. In particular, measurements show that on typical PCs encrypted Web communications using SSL and RC4 can transfer data at speeds similar to non-encrypted HTTP. This bodes well for electronic commerce.

6. References

[Bolyard97] Nelson Bolyard, “Export Client SSL Connection Details”, 1997, http://home.netscape.com/eng/ssl3/traces/trc-clnt-ex.html

[Dierks97], Tim Dierks, Christopher Allen, November 12, 1997, “The TLS Protocol Version 1.0”, http://www.ietf.org/internet-drafts/draft-ietf-tls-protocol-05.txt

[Fielding98] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, August 1, 1998, “Hypertext Transfer Protocol -- HTTP/1.1”, http://www.w3.org/Protocols/History.html

[Freier96] Freier, Alan O., Philip Karlton, Paul C. Kocher, “The SSL Protocol Version 3.0” Internet Draft, November 18, 1996, http://home.netscape.com/eng/ssl3/draft302.txt

[Goldberg98A] Arthur Goldberg, Ilya Pevzner, Robert Buff, “Characteristics of Internet and Intranet Web Proxy Traces”, to be published in the Computer Measurement Group Conference, CMG98, December 1998.

[Goldberg98B] Arthur Goldberg, Robert Buff, Andrew Schmitt, “Secure Web Server Performance Dramatically Improved By Caching SSL Session Keys”, Published in the "Workshop on Internet Server Performance", held in conjunction with SIGMETRICS'98, June 23, 1998.

[Hickman95] K.E.B. Hickman. “The SSL Protocol” December 1995, http://www.netscape.com/newsref/std/ssl.html.

[Hudson] Hudson, Tim J., and Eric A. Young. “SSLeay Programmer Reference”, circa 1997, http://psych.psy.uq.oz.au/~ftp/Crypto/ssl.html.

[Netcraft97], “Secure Web Server Survey”, http://www.netcraft.com/surveys/analysis/https/1997/Mar/CMatch/strongsv.html

and http://www.netcraft.com/surveys/analysis/https/1997/Mar/CMatch/strength.html.

[Netcraft], “Web Server Survey”, http://www.netcraft.com/Survey/Reports/9806/.

[RC4] RSA, “What is RC4?”, http://www.rsa.com/rsalabs/faq/html/3-6-3.html.

[SSL] Netscape, “Introduction to SSL”, http://developer.netscape.com/docs/manuals/security/sslin/contents.htm.