# Tivoli WebSEAL - Sizing and Capacity Planning

# Contents

# Tivoli WebSEAL - Sizing and capacity planning

WebSEAL provides an authentication and authorization mechanism based on Tivoli Access Manager. It enables an end-to-end Single Sign On (SSO) solution for secure transactions for WebSphere application servers.

Published September 2007

Here WebSEAL is used as proxy inside a DMZ from a secure WebSphere Application server environment. Each server runs on virtual hardware under z/VM.

The paper describes how to setup the environment and how it performs in various scenarios. Additionally it shows the impressive advantage of the IBM system z cryptographic hardware features CPACF and CEX2C cards and how to setup the system to get cryptographic hardware support. It also demonstrates that the implementation of a DMZ with all its services and servers is a very good case for server consolidation on z/VM.

To view or download the PDF version of this document, click on the following link:

Tivoli WebSEAL - Sizing and Capacity Planning (about 948 KB)

## Objectives for the Tivoli WebSEAL performance tests

The Tivoli WebSEAL Sizing and Capacity Planning project performance tests were performed to determine performance characteristics for the WebSEAL environment on Linux for IBM System z.

WebSEAL is a component of Tivoli® Access Manager for e-business that provides an authentication and authorization mechanism to enable end-to-end Single Sign On (SSO) for HTTP based Web servers. One such Web server is WebSphere® Application Server residing on z/OS®. With the WebSEAL solution, the z/OS system can remain safely isolated from Internet traffic and threats. In this solution, WebSEAL acts as a reverse proxy (a proxy that lets users into a protected network as opposed to one that lets users out). The WebSEAL reverse proxy would likely be deployed in the DMZ or similar security zone. A client would be authenticated by WebSEAL via userID and password, and a token would be assigned and associated with the transaction for the remainder of the communication. Authentication via WebSEAL can be safely performed in the DMZ with no risk to applications running on z/OS.

While WebSEAL does not run on the z/OS platform, it does run on Linux® for IBM® System z®. An integrated solution with WebSEAL on Linux for System z providing authentication, authorization, and SSO for applications running in WebSphere on z/OS has been tested and documented as an IBM System z Linux Utility Service. See the System z Linux Utility Service Web page at:

http://www.ibm.com/systems/z/os/linux/utilities/index.html

The objectives of this project were to determine performance characteristics for the WebSEAL environment. The information obtained in these tests should not be used

to develop capacity sizing rules-of-thumb for customer environments. For sizing and capacity planning, contact our educated specialists from the IBM System z sizing team who use the appropriate capacity planning tools.

## Summary for the Tivoli WebSEAL performance tests

After performing performance tests on a Tivoli WebSEAL environment on Linux for IBM System z, we compiled a summary of our results and recommendations.

Test results and recommendations are specific to the test environment. Findings useful in the test environment might not apply in other environments with other workloads. For our detailed test results information, see "Results for the Tivoli WebSEAL performance tests" on page 10.

The following are our summary results:

- The most critical resource was the CPU at the WebSEAL server. Just increasing the number of CPUs from one to two increased the throughput by nearly a factor of two without increasing the number of workload generators.
- The CPU load of the other servers needed for WebSEAL was very low, making this is an ideal environment to host on z/VM® because z/VM offers the capability of running with more virtual CPUs than physical CPUs available, which allows the sharing of unused CPU capacity among the guests.
- The log file is a resource where the access is serialized. This is a bottleneck whose impact increases with the number of workload generators. This shows that the I/O bandwidth for the log file is another constraint besides CPU capacity. To ensure the maximum I/O bandwidth, it is recommended to place the log file at least on a separate disk to avoid the sequential I/O flow being interrupted by other I/O requests or use a logical volume striped over disks from several ranks, which is the better option. Be aware that all our measurements were logging to a file.
- As expected, SSL encryption has a negative impact on throughput because the encryption/decryption is a significant effort. Using CP Assist for Cryptographic Function (CPACF) hardware support reduces that impact significantly. By using large pages, throughput is improved by a factor of 2.4.
- When the WebSEAL server runs CPU constrained it is a bottleneck limiting throughput. The cryptographic hardware support gives the CPU some relief, which enables the possibility to increase the workload beyond what could be driven without the hardware support.
- Using an SSL junction from WebSEAL to the back end WebSphere application server leads to a significant degradation of the throughput. The WebSEAL server has much more work to do in this scenario. The encrypted packages have to be decrypted, analyzed, and encrypted again. The degradation could be reduced by using hardware support for encryption. Be aware that the alternative to this overhead, like an isolated network using a HiperSockets™ connection or a z/VM guest LAN, can be easily implemented on System z and produces the same grade of security without this overhead.
- Using the CPACF from the CPU for decryption and encryption of the requests and the CEX2C cryptographic adapter feature for the authentication results in an improvement by the hardware cryptographic support up to a factor of three.

  See "Enable and run WebSEAL with cryptographic hardware support" on page 24 for a description on how to setup the cryptographic hardware support.

# Hardware equipment and software environment for the Tivoli WebSEAL performance tests

To perform our Tivoli WebSEAL performance tests, we created a customer-like environment. We configured the hardware and software, the network, the storage server, and the z/VM guests.

## Host Hardware

Two LPARs on a 16-way IBM System z9®, type 2094-S18, equipped with:

*Table 1. System memory and CPUs*

| System/LPAR | Memory | Dedicated CPUs |
|---|---|---|
| z/VM | 5120 MB | 3(4) |
| z/OS | 4096 MB | 2(4) |

Additionally, the z/VM LPAR used:

- OSA Express 2 gigabit Ethernet card (OSA code level 0805)
- Crypto type: CEX2C (Supplies 2 cards)

## Network setup

- The z/VM LPAR is connected to the clients via Fiber Gigabit Ethernet Interface
- The z/VM guests used virtual guest LANs
- The z/VM LPAR and the z/OS LPAR were connected via a HiperSockets connection

## Storage server setup

IBM TotalStorage® Enterprise Storage Server® (ESS) 2105 800

- IBM 3390 disk model 3
- Physical DDMS with 15,000 RPMs

## z/VM guest setup

Table 2 shows the z/VM guest configuration we used for our tests. We used this configuration for our tests using WebSphere Application Server on z/OS and our tests using Apache HTTP Server on Linux.

*Table 2. Test system configuration: WebSphere Application Server*

| System/Guest | Memory | CPUs |
|---|---|---|
| WebSEAL | 512 MB | 1(2) |
| Tivoli Access Manager/IBM Tivoli Directory Service | 1024 MB | 1 |
| DB2® UDB | 2048 MB | 1 |
| Firewall1 | 512 MB | 1 |
| Firewall2 | 512 MB | 1 |

## Client hardware

4 x330 PCs with 2 processors, 1.26 Ghz

## Software

*Table 3. Host and client software used*

| Product | Version/Level |
|---|---|
| Apache HTTP Server | 2.0.49 |
| DB2 Client RTE | Version 8.2 FixPak 4 (equivalent to Version 8.1 FixPak 11) |
| Curl | curl-7.12.1-8.rhel4, curl-devel-7.12.1-8.rhel4 |
| IBM DB2® Universal Database™ (DB2 UDB) | 8.2 |
| IBM Key Management (gskit) | 7.0.3.13 |
| OpenSSL | 0.9.7d-15.21, openssl-32bit-9-200511222035 |
| Red Hat Enterprise Linux | RHEL 4 ES (client) |
| SUSE Linux Enterprise Server | SLES9 SP3 |
| Tcl | Tcl-devel-8.4.7-2, tcl-8.4.7-2 |
| Tivoli Access Manager for e-business (contains WebSEAL, IBM Tivoli Directory Service, and Tivoli Access Manager | 6.0.0.1 |
| WebSphere Application Server | 6.0.2 |
| z/OS | 1.07 |
| z/VM | 5.2 |

## z/VM guest configuration

z/VM guests were interconnected by a z/VM guest LAN configured as a HiperSockets, MFS 24 K. All guests ran SLES9 SP3, kernel level 2.65-7.244-s390x.

*Table 4. z/VM guests*

| Host Name | IP address | # Virtual CPUs | Memory | Function |
|---|---|---|---|---|
| LNX00080 | 10.10.80.95<br>192.168.40.95<br>10.10.50.95 | 1(2) | 512 MB | WebSEAL Server |
| LNX00081 | 10.10.80.200<br>192.168.30.200 | 1 | 2 GB | DB2 UDB |
| LNX00082 | 10.10.80.85<br>192.168.30.85 | 1 | 1 GB | Tivoli Access Manager / IBM Tivoli Directory Service |
| LNX00090 | 10.10.80.21<br>192.168.30.21<br>192.168.40.21 | 1 | 512 MB | Firewall 1 |
| LNX00091 | 10.10.80.22<br>192.168.40.22<br>10.10.60.22 | 1 | 512 MB | Firewall 2 |

## Test environment

Our Tivoli WebSEAL test environment consisted of an IBM System z and an IBM System x server.

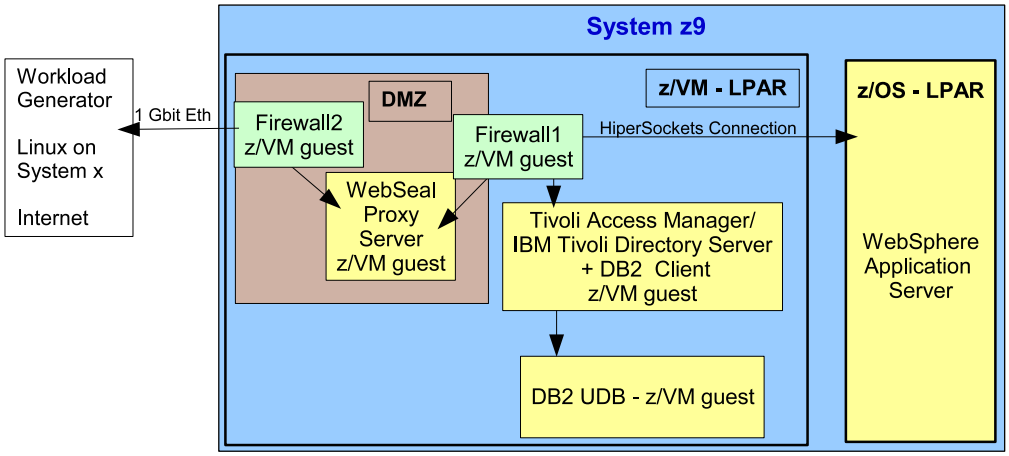


*Figure 1. WebSEAL test environment*

Figure 1 shows only the setup of the WebSEAL environment.

The System z contained a z/VM LPAR with five guests and a z/OS LPAR. The network was split into three parts:

- The System x® and the System z systems were connected in the unsecured external zone through a 1 Gb Ethernet connection. The System x system contained the WebSEAL workload generator, which generated the workload.
- The DMZ contained one guest with the WebSEAL Proxy Server protected from the external zone with a firewall (Firewall 2) running in a separate guest.
- The trusted internal zone is protected with another guest with a more restrictive firewall (Firewall 1) and contains one guest for each of the following servers:
  - The Tivoli Access Manager / IBM Tivoli Directory Service server and the DB2 client
  - The DB2 UDB database server
- The z/OS LPAR is connected to the z/VM LPAR through a HiperSockets connection and contained the WebSphere Application Server.
- The network zones on z/VM were implemented as HiperSockets guest LANs.

See the Red Paper *IBM Tivoli Access Manager for e-business* for general information on the software products used in this test. The red paper can be found at:

`http://www.redbooks.ibm.com/redpapers/pdfs/redp3677.pdf`

# Test case setup for the Tivoli WebSEAL performance tests

To perform our Tivoli WebSEAL performance tests, we identified and set up the software and hardware for various test case scenarios. We set up our DB2 server, set up three levels of security, created a WebSEAL junction, put our Web pages in the correct directories, created ACL descriptions, and used the System z9 cryptographic hardware functionality.

### Creating the IBM Tivoli Directory Server (LDAP) Database on the DB2 Server

The following commands were used to create the database used by IBM Tivoli Directory Server for our measurements:

```
db2 create db ldapdb2 using codeset UTF-8 territory US
db2 CREATE BUFFERPOOL LDAPBP SIZE 1230 PAGESIZE 32K NOT EXTENDED STORAGE
db2 ALTER BUFFERPOOL IBMDEFAULTBP SIZE 29500
db2 "CREATE TABLESPACE LDAPSPACE PAGESIZE 32K MANAGED BY SYSTEM
db2 update dbm cfg using svcename idsservice
db2set DB2COMM=TCPIP
db2empfa ldapdb2
db2 update db cfg for ldapdb2 using APPLHEAPSZ 1024
db2 update db cfg for ldapdb2 using DBHEAP 1200
```

### Creating the users in the LDAP Registry

The following commands were used to create 10,000 user entries:

```
user create user1 cn=user1,c=us user1 user1 linux390
user modify user1 account-valid yes
```

where user1 was varied from user1 though user10000.

## Benchmark scenarios

To test the different levels of security, we used three categories of test procedures for our Tivoli WebSEAL performance tests.

There were three categories of test procedures we used:

- Non-SSL (or TCP)

  This accesses an unprotected Web page. There is no SSL or authentication set up.

  Used in "Non-SSL page access through a TCP junction, unauthenticated" on page 10.
- SSL

  This accesses a protected Web page that requires authentication. It establishes the SSL session and authentication at the beginning of the test and reuses both throughout the remainder of the test.

  Used in "SSL page access through a TCP junction" on page 13, "SSL page access through a TCP junction with "-c all" option" on page 17, and "SSL page access through an SSL junction" on page 20.
- SSL authentication

  This is an authentication test. It requests a protected web page that requires authentication. The SSL session is re-established and a new user is authenticated with each page requested during the test.

  Used in "SSL page access with new authentication and new SSL session every request" on page 22.

The Tivoli Access Manager administrative command utility (pdadmin) was used to configure the WebSEAL server for the workloads driven by the benchmarks. The following table describes the WebSEAL configuration, including the pdadmin command options, used for each benchmark scenario.

*Table 5. Benchmark scenarios*

| # | Benchmark Description | Junction | Crypto | Auth. type | Page Size | Back end |
|---|---|---|---|---|---|---|
| 1 | HTTP page access to a TCP junction to back end | TCP | None | None | 5.8 KB | WebSphere z/OS |
| | **pdadmin -a sec_master -p <password> server task webseald-<webseal host> create -t tcp -p 9080 -h <websphere server> /ivload/tcp-jct-wasserver-unauth** | | | | | |
| 2 | HTTPS page access to TCP junction to back end | TCP | AES-128 sw/hw | Once, with session reuse | 5.8 KB<br>12 KB<br>2.9 MB | WebSphere z/OS |
| | **pdadmin -a sec_master -p <password> server task webseald-<webseal host> create -t tcp -p 9080 -h <websphere server> /ivload/tcp-jct-wasserver** | | | | | |
| 3 | HTTPS page access to TCP junction to back end with "-c all" option | TCP | AES-128 sw/hw | Once with session reuse | 5.8 KB | WebSphere z/OS |
| | **pdadmin -a sec_master -p <password> server task webseald-<webseal host> create -t tcp -p 9080 -h <websphere server> -c all /ivload/tcp-jct-wasserver** | | | | | |
| 4 | HTTPS page access to SSL junction to back end | SSL | DES-168 sw/hw | Once with session reuse | 5.8 KB 12 KB | WebSphere Linux |
| | **pdadmin -a sec_master -p <password> server task webseald-<webseal host> create -t ssl -p 9443 -h <websphere server> /ivload/ssl-jct-apache2** | | | | | |
| 5 | HTTPS authentication page access | None | AES-128 sw/hw | Each, with new session | 100 B | None |
| **Note:** While it is possible to have an HTTP page access to an SSL junction back end, this configuration was not tested because it is counter intuitive. | | | | | | |

### Authentication process

Basic authentication requires two requests from the browser. The first request is unauthenticated. It asks for the Web page without providing authentication information, for example, the authentication header. The Web server rejects this request with a HTTP 401 error indicating that authentication is required. The Web browser pops up a user name and password authentication header so that re-authentication does not occur.

### SSL sessions

Negotiating a new SSL session is costly in terms of performance (CPU usage). The test that authenticates with each request also negotiates a new session with each request. Although both are very costly, both events occur together, so it is a realistic representation of common usage to perform the test in this way.

"Session reuse" means the SSL session is negotiated at the beginning of the test and the session is reused on each subsequent request.

### Authentication types

Authentication type ″None″ means the request is unauthenticated. For these tests, an Access Control List (ACL) is placed in the WebSEAL object space indicating that the page does not require authentication.

Authentication type ″Once″ means there is one authentication per load generating client and subsequent requests are performed after this authentication by reusing the same session. The authentication is not part of the measured results. Other than at the beginning of the test, there are no LDAP operations for consecutive tests refers to an authenticated page access request after an initial authentication. Post authentication cases do not perform any operations to the LDAP server.

Authentication type ″Each″ means that each request creates a new session, which requires a new SSL session negotiation. This includes a user login using Basic Authentication (BA).

## WebSEAL setup

For our Tivoli WebSEAL performance tests, we setup a WebSEAL junction, a connection between a WebSEAL server and a backend Web application server.

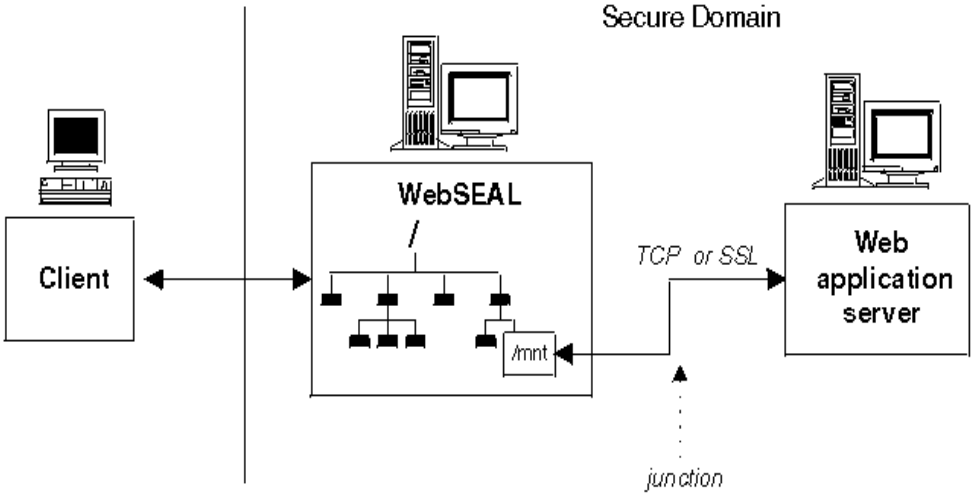


*Figure 2. WebSEAL junction*

- The connection between a WebSEAL server and a back end Web application server is known as a WebSEAL junction. A WebSEAL junction is a TCP or SSL connection between a front end WebSEAL server and a back end server. The back end server can be another WebSEAL server or, more commonly, a third-party Web application server. The back end server Web space is ″connected″ to the WebSEAL server at a specially designated junction (mount) point in the WebSEAL Web space. A junction allows WebSEAL to provide protective services on behalf of the back end server. WebSEAL can perform authentication and authorization checks on all requests before passing those requests on to the back end server.
- WebSEAL maintains the following HTTP log files, which record HTTP activity:
  - request.log
  - agent.log
  - referer.log

By default, these log files are located in the /var/pdweb/www-default/log directory. Stanza entries for configuring traditional HTTP logging are located in the [logging] stanza of the WebSEAL configuration file. The WebSEAL web server request logs are not automatically deleted by WebSEAL. The logs will grow until manually deleted or the file system in which they are placed becomes full. In our tests the log files, when used, were reset at the beginning of each run.

## Web pages

For our Tivoli WebSEAL tests, our web pages needed to be stored in specific directories. Also, commands were used to create ACL definitions.

All Web pages were stored in the WebSphere Application Server installed applications directory structure under the sample application PlantsbyWebSphere directory. Pages were also placed in the WebSEAL document root directory on the WebSEAL server. Benchmarks that did not use a junction retrieved their pages from WebSEAL's document root directory.

### ACL definitions

The following commands create the ACL that was attached to the web pages for the unauthenticated benchmark:

```
pd_host=websealserver.pdl.pok.ibm.com
cat <<EOF | pdadmin -a sec_master -p lsc2lsc
acl create test-unauthenticated
acl modify test-unauthenticated set Group iv-admin TcmdbsvaBRrxl
acl modify test-unauthenticated set Group webseal-servers Tgmdbsrxl
acl modify test-unauthenticated set User sec_master TcmdbsvaBRrxl
acl modify test-unauthenticated set Any-other Trx
acl modify test-unauthenticated set Unauthenticated Tr
EOF
```

The above ACL was attached to the following objects in the WebSEAL object space:

```
pdadmin -a sec_master -p lsc2lsc acl attach ↓
/WebSEAL/$pd_host-default/ivload/tcp-jct-wasserver-unauth tcp test-unauthenticated
```

**Note:** The ↓ symbol indicates that the text continues on the next line. These lines should be entered on one line, not broken into multiple lines.

## System z9 cryptographic hardware

Several of the workloads run in our tests used the HTTPS protocol. For these workloads we compared software encryption/decryption against the System z9 hardware assist for cryptographic functions. Our tests used asymmetric and symmetric cryptographic functions.

The SSL handshakes use asymmetric functions while the data is encrypted and decrypted with symmetric functions. The System z9 cryptographic adapter cards, Crypto Express 2 (CEX2), can accelerate asymmetric cryptographic operations. The System z9 CPACF from the System z processor can accelerate symmetric cryptographic operations (DES, TDES, and SHA-1 on z990/z890 and DES, TDES, SHA-1, SHA-256 and AES-128 on a z9®). Software support for System z9 cryptographic hardware is provided by libICA and the z90crypt device driver. WebSEAL requires the openCryptoki and GSKit libraries to interface to the z9 cryptographic hardware.

The CEX2C cryptographic adapter feature was used in these measurements. It provides two processors and both processors were used to accelerate asymmetric cryptographic operations.

SSL handshakes use asymmetric functions. Handshakes occur at the start of all SSL sessions. The encryption/decryption of the data is performed by symmetric functions.

### Data collection and output

For our Tivoli WebSEAL tests, the number of successful HTTP/S requests for a page were tracked and recorded.

The WebSEAL benchmarks were driven by tcl Curl scripts which ran on RedHat Linux on a System x box. The tcl Curl script generated HTTP/S requests for a page. The scripts counted the number of satisfied requests within the total run length and reported this number as transactions per second.

## Results for the Tivoli WebSEAL performance tests

After performing our Tivoli WebSEAL performance tests, we charted our test results, interpreted the results, and created recommendations.

We ran the following tests:

- "Non-SSL page access through a TCP junction, unauthenticated"
- "SSL page access through a TCP junction" on page 13
- "SSL page access through a TCP junction with "-c all" option" on page 17
- "SSL page access through an SSL junction" on page 20
- "SSL page access with new authentication and new SSL session every request" on page 22

### Non-SSL page access through a TCP junction, unauthenticated

In this test case the client accesses the Web page on the back end WebSphere Application Server, unauthenticated via an unencrypted connection (TCP junction).

## Using a page size of 5.8 KB

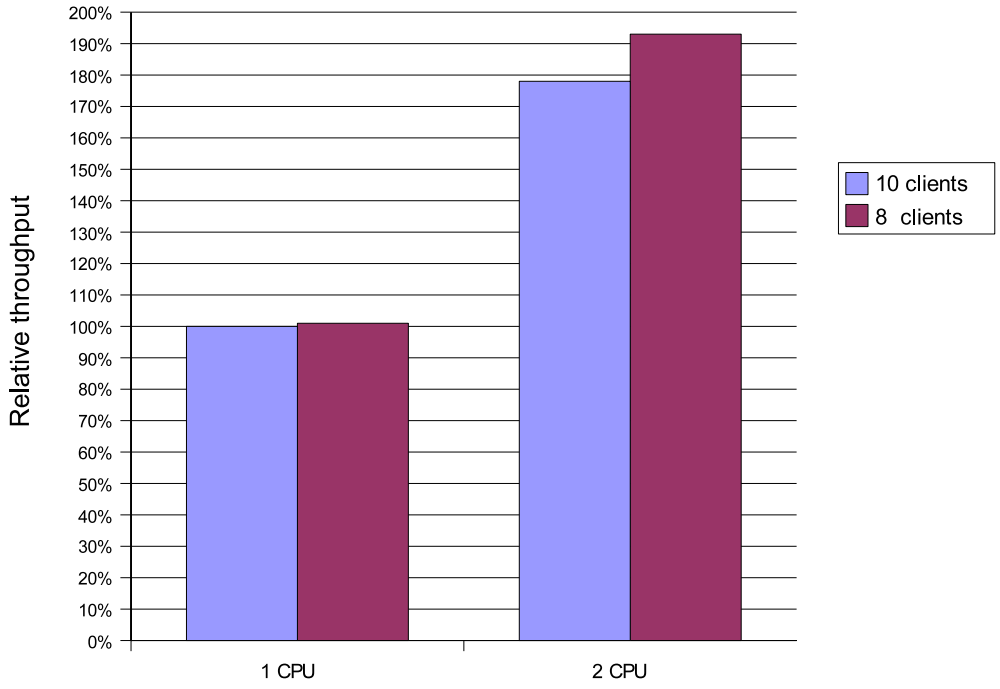


*Figure 3. 5.8 KB non-SSL page access through a TCP junction, unauthenticated - throughput per second*

Figure 4 shows the CPU utilization of the z/VM and z/OS LPARs in a curve and the details of the z/VM guests as stacked bars. Guests with a utilization of nearly zero, like the Tivoli Access Manager server, do not appear.

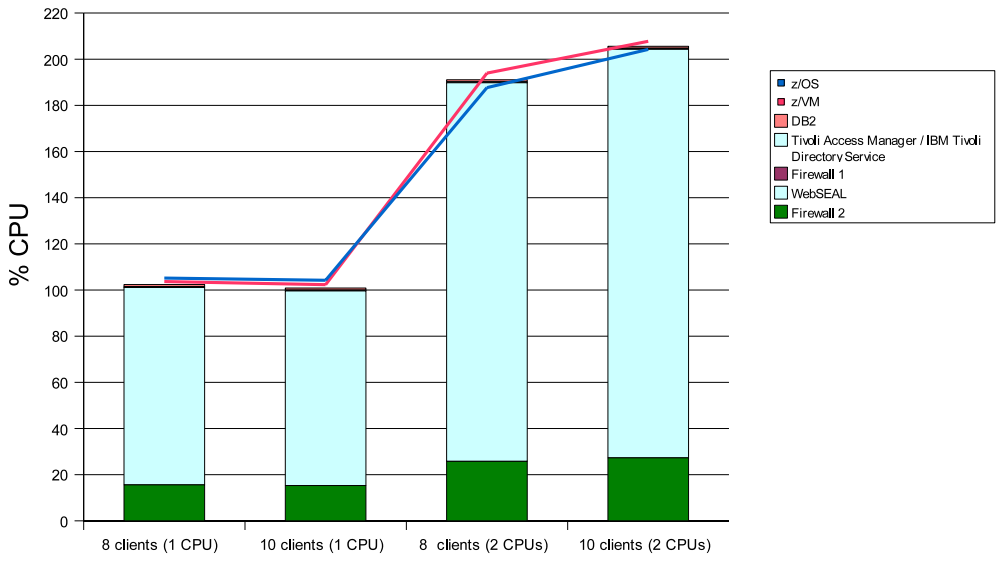


*Figure 4. 5.8 KB non-SSL page access through a TCP junction, unauthenticated - % CPU*

### Observations

- The CPU utilization of the Firewall 1 system, the system with the Tivoli Access Manager and LDAP servers, and of the DB2 UDB server is very low.
- Scalability was good with 1.9 for 8 clients and 1.8 for 10 clients.

- The CPU utilization on the WebSEAL server seemed to be a major bottleneck, even though it was only 85% utilized.
- Increasing the number of clients with two CPUs resulted in a throughput improvement of about 9%.
- The CPU load on the other related systems (firewalls, DB2 UDB, Tivoli Access Manager / IBM Tivoli Directory Service server) was very low.

**Conclusion**

The most critical resource was the CPU at the WebSEAL server. Just increasing the number of CPUs from one to two increased the throughput by nearly a factor of two without increasing the number of workload generators. The CPU load of the other servers needed from the WebSEAL server was very low, making this is an ideal environment to host on z/VM. For details on logging see "Effects of WebSEAL logging on throughput."

## Effects of WebSEAL logging on throughput

This test is a repeat of the scenario one test except that it is being used to compare the effect of WebSEAL logging. We reran the scenario one test with the WebSEAL logs assigned to files and with the logs assigned to /dev/null. These tests were done with WebSEAL's guest machine assigned two CPUs.

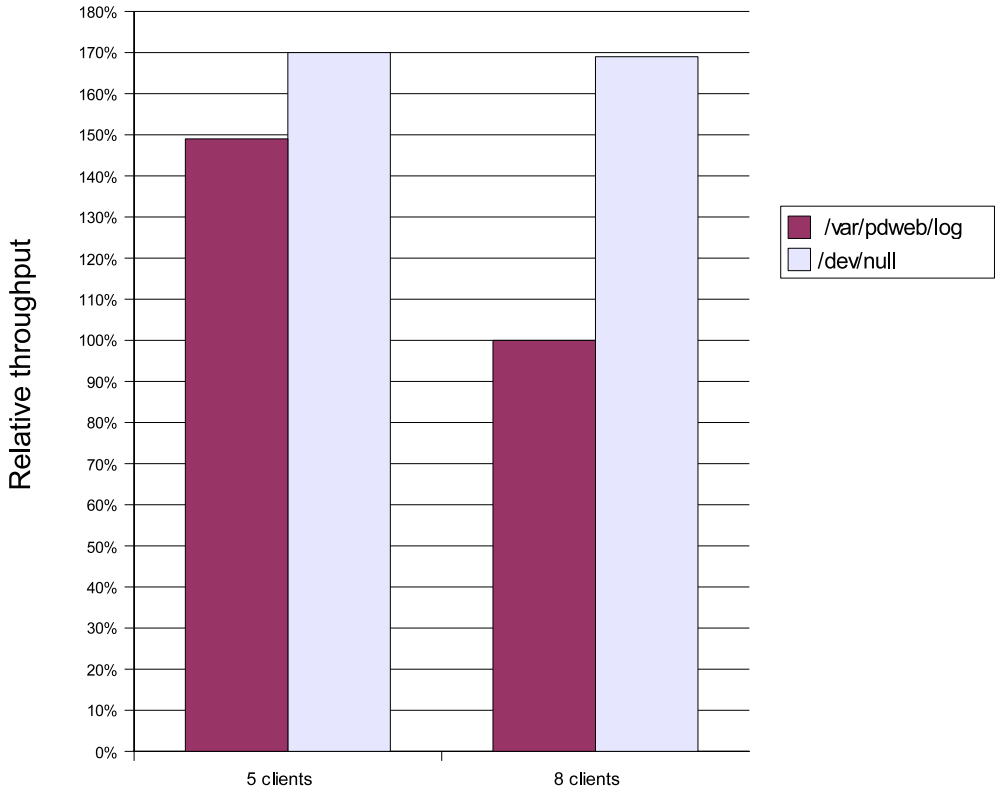


*Figure 5. 5.8 KB non-SSL page access, no junction, unauthenticated*

**Observations**

Logging to /dev/null improves the throughput, where the improvement for eight clients is 4.3% and for ten clients is 5.1%

**Conclusion**

The log file is a resource where the access is serialized. This is a bottleneck whose impact increases with the number of workload generators. This shows that the I/O bandwidth for the log file is another constraint besides CPU capacity. To ensure the maximum I/O bandwidth, it is recommended to place the log file at least on a separate disk to avoid the sequential I/O flow being interrupted by other I/O requests or use a logical volume striped over disks from several ranks, which is the better option.

Be aware that all our measurements using logging to a file suffered from this constraint.

**Related information**

Tivoli WebSEAL - hardware equipment and software environment
To perform our Tivoli WebSEAL performance tests, we created a customer-like environment. We configured the hardware and software, the network, the storage server, and the z/VM guests.

Tivoli WebSEAL - test case setup
To perform our Tivoli WebSEAL performance tests, we identified and set up the software and hardware for various test case scenarios. We set up our DB2 server, set up three levels of security, created a WebSEAL junction, put our Web pages in the correct directories, created ACL descriptions, and used the System z9 cryptographic hardware functionality.

Tivoli WebSEAL - detailed set up examples
The detailed setup we performed to enable WebSEAL with cryptographic hardware support is provided here.

Tivoli WebSEAL - other sources of information
Additional resources to provide information on the products, hardware, and software discussed in this paper can be found in various books and at various Web sites.

# SSL page access through a TCP junction

In this set of tests, we investigated three scenarios; using a page size of 5.8 KB, using a page size of 12 KB, and using a page size of 2.9 MB.

## Using a page size of 5.8 KB

In this test case the client has an SSL connection to the WebSEAL server, which requires authentication and encryption, and accesses the Web page on the back end WebSphere Application Server via an unencrypted connection (TCP junction).

Because it involves SSL access, these tests compare software encryption against hardware encryption. Encryption occurs between the client and WebSEAL. The AES-128 encryption algorithm was used for encryption operations between the client and WebSEAL. In this test, only one handshake occurred.

**Encryption algorithm**: AES-128

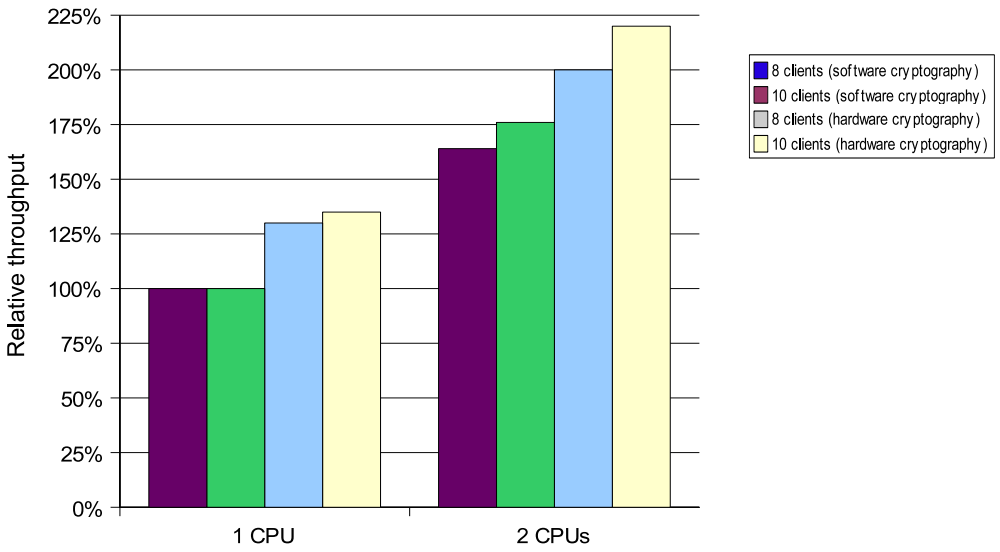


*Figure 6. 5.8 KB SSL page access through a TCP junction*

**Observations**

The WebSEAL system ran fully CPU constrained with one CPU in these tests. The cryptographic hardware support increased the throughput. However, using more workload generators only had an impact when additional CPU capacity was available.

**Conclusion**

In this environment there are two systems, the WebSEAL server and the WebSphere Application Server, that have high CPU loads. Therefore, the availability of CPU resources for these systems is a major component that affects the total throughput. The cryptographic hardware support gives the CPU on the WebSEAL server some relief, which leads to an increase in throughput. Using more CPUs resulted in higher throughput and allowed us to increase the number of workload generators, which increased the throughput further. In all cases, the CPU utilization for the other servers needed to implement the WebSEAL environment is very low. This confirms that this environment is ideal for consolidation under z/VM because of the capability to run with more virtual CPUs than physical CPUs available. The four CPUs of the z/VM LPAR are only utilized to 50%, which means we could have run this test with only three CPUs on the LPAR while granting one more CPU to the WebSEAL system to use the full system.

## Using a page size of 12 KB

In this test case the client had an SSL connection to the WebSEAL server, which requires authentication and encryption, and accesses the Web page on the back end WebSphere Application Server via an unencrypted connection (TCP junction). In order to better show the benefits of the symmetric encryption acceleration, the page size was increased to 12 KB.

**Encryption algorithm**: AES-128

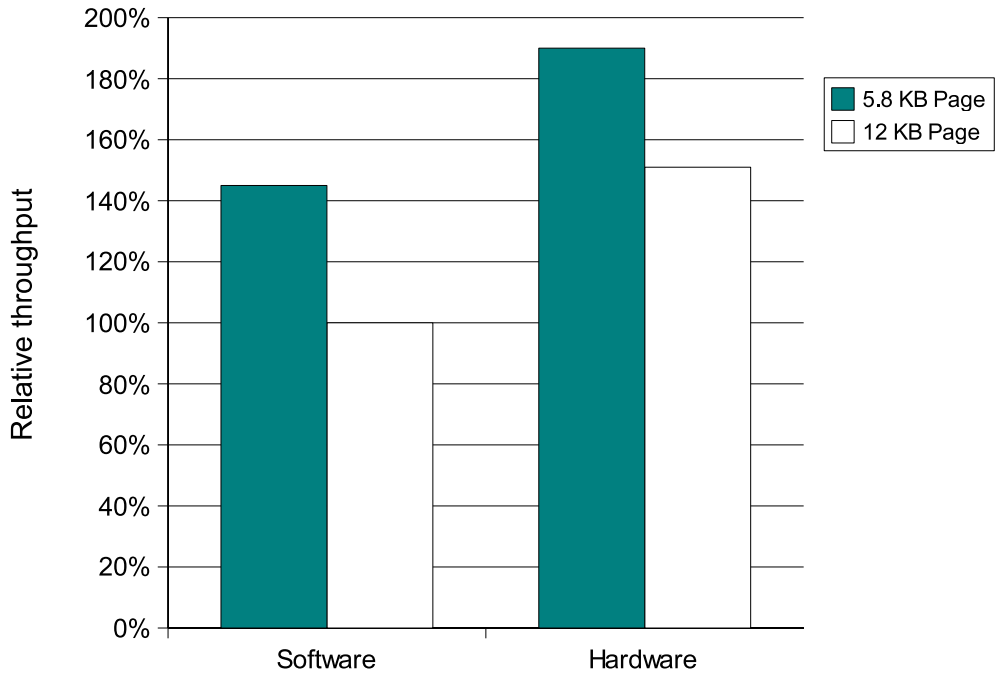


*Figure 7. 5.8 KB versus 12 KB SSL access through a TCP junction*

**Observations**

The larger the page size, the lower the throughput in terms of pages per second.

**Conclusion**

This is expected because the larger page size needs more time to be transferred from the application server to the client. For the impact of the hardware cryptography support, see the conclusion of "Using a page size of 2.9 MB."

## Using a page size of 2.9 MB

In this test case the client has an SSL connection to the WebSEAL server, which requires authentication and encryption, and accesses the Web page on the back end WebSphere Application Server via an unencrypted connection (TCP junction). In order to better show the benefits of the symmetric encryption acceleration, the page size was increased from 12 KB to 2.9 MB.

**Encryption algorithm**: AES-128

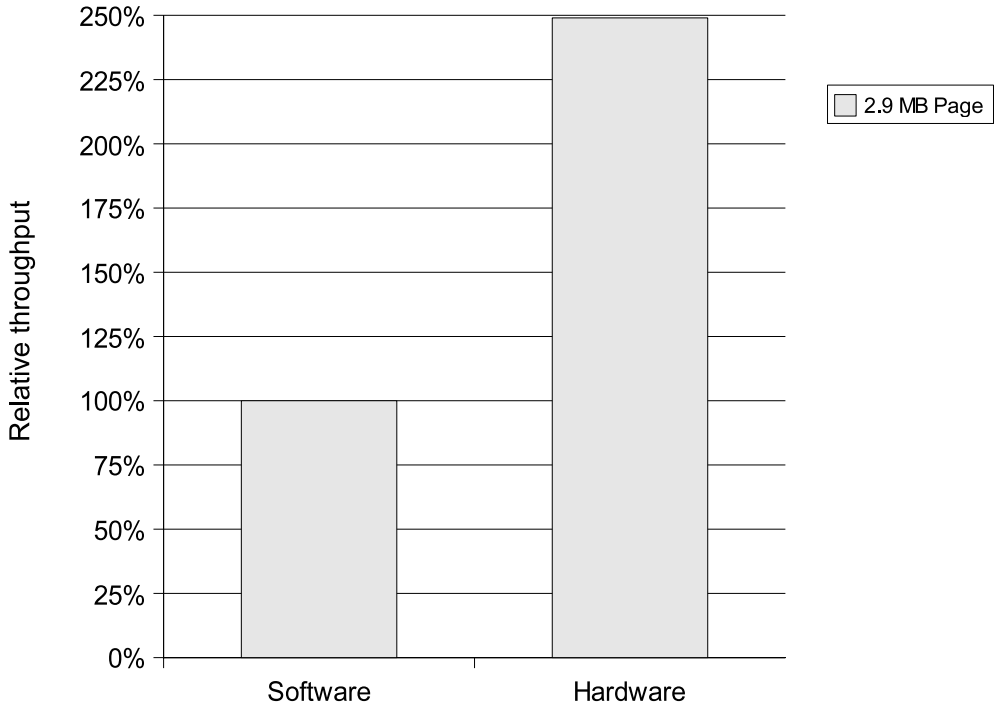


*Figure 8. 2.9 MB page SSL access through a TCP junction*

## Impact of cryptographic hardware

For a better comparison, the throughput was normalized to throughput per client. Figure 9 shows the improvement using the cryptographic hardware for encryption for each request size.

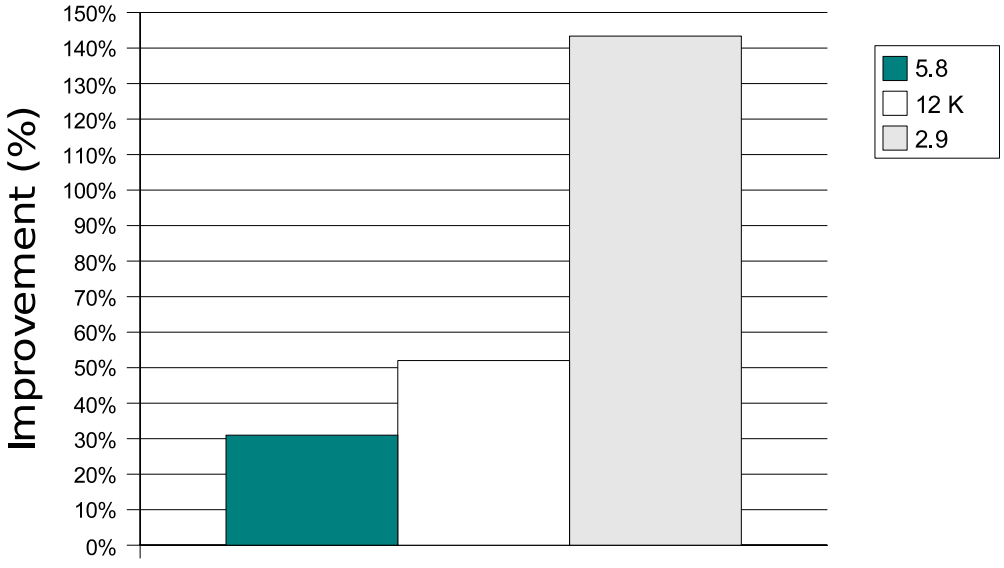


*Figure 9. Improvement by hardware cryptographic support versus software encryption for an increasing request size*

### Observations

Comparing Figure 6 on page 14 in "SSL page access through a TCP junction" on page 13 with Figure 3 on page 11 in "Non-SSL page access through a TCP junction, unauthenticated" on page 10 shows that using SSL encryption between the client and WebSEAL server degraded the throughput by about 35% for a page size of 5.8 KB without hardware support. Using hardware support increased the throughput by 30%, so the throughput was now only 12% below the unencrypted case. Figure 9 on page 16 shows that the advantage of the hardware encryption increases with the page size as expected. For the 2.9 MB page size, the improvement was up to 143% which is more than twice what software encryption gets.

**Conclusion**

The additional effort for the SSL encryption and decryption requires CPU resources, which, in the case of a fully utilized system, leads to a degradation of the throughput. Using the CPACF hardware support reduces the cost to a minimum. For large pages we saw an improvement of a factor of 2.4 compared with software encryption.

**Related information**

Tivoli WebSEAL - hardware equipment and software environment
To perform our Tivoli WebSEAL performance tests, we created a customer-like environment. We configured the hardware and software, the network, the storage server, and the z/VM guests.

Tivoli WebSEAL - test case setup
To perform our Tivoli WebSEAL performance tests, we identified and set up the software and hardware for various test case scenarios. We set up our DB2 server, set up three levels of security, created a WebSEAL junction, put our Web pages in the correct directories, created ACL descriptions, and used the System z9 cryptographic hardware functionality.

Tivoli WebSEAL - detailed set up examples
The detailed setup we performed to enable WebSEAL with cryptographic hardware support is provided here.

Tivoli WebSEAL - other sources of information
Additional resources to provide information on the products, hardware, and software discussed in this paper can be found in various books and at various Web sites.

# SSL page access through a TCP junction with "-c all" option

In this test case the client has an SSL connection to the WebSEAL server, which requires authentication and encryption, and accesses the Web page on the back end WebSphere Application server via an unencrypted connection (TCP junction).

## Using a page size of 5.8 KB

This test involves SSL access that compares software encryption against hardware encryption. Encryption occurs between the client and WebSEAL. The AES-128 encryption algorithm was used for encryption operations between the client and WebSEAL. In this test, only one handshake occurred. The -c option allows you to insert Tivoli Access Manager-specific client identity and group membership information into the HTTP headers of requests destined for junctioned third-party servers. The HTTP header information enables applications on junctioned third-party servers to perform user-specific actions (such as SSO) based on the client's Tivoli Access Manager identity.

HTTP header information must be transformed by the back end server to environment variable format for use by a service on the back end server. Header information is transformed into a CGI environment variable format by replacing all dashes (-) with under bars (_) and adding ″HTTP″ to the beginning of the string. The value of the HTTP header becomes the value of the new environment variable.

The Tivoli Access Manager-specific HTTP header entries are available to CGI programs as the environment variables HTTP_IV_USER, HTTP_IV_GROUPS and HTTP_IV_CREDS. For other application framework products, see the product's documentation for instructions on extracting headers from HTTP requests.

The -c all option inserts all three types of identity information into the HTTP header. The addition of the -c option places additional processing overhead on WebSEAL.

**Encryption algorithm**: AES-128

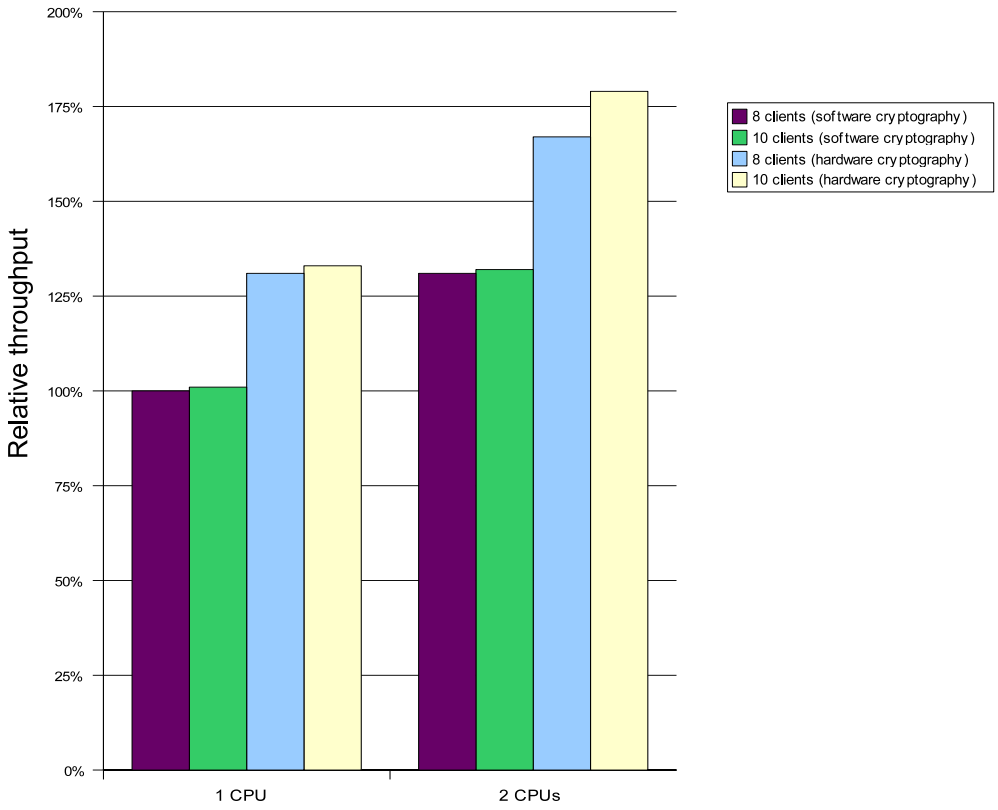


*Figure 10. 5.8 KB SSL page access through a TCP junction with -c all*

Figure 11 on page 19 is a comparison between 5.8KB SSL page access through a TCP junction with the -c option and 5.8KB SSL page access through a TCP junction without the -c option.

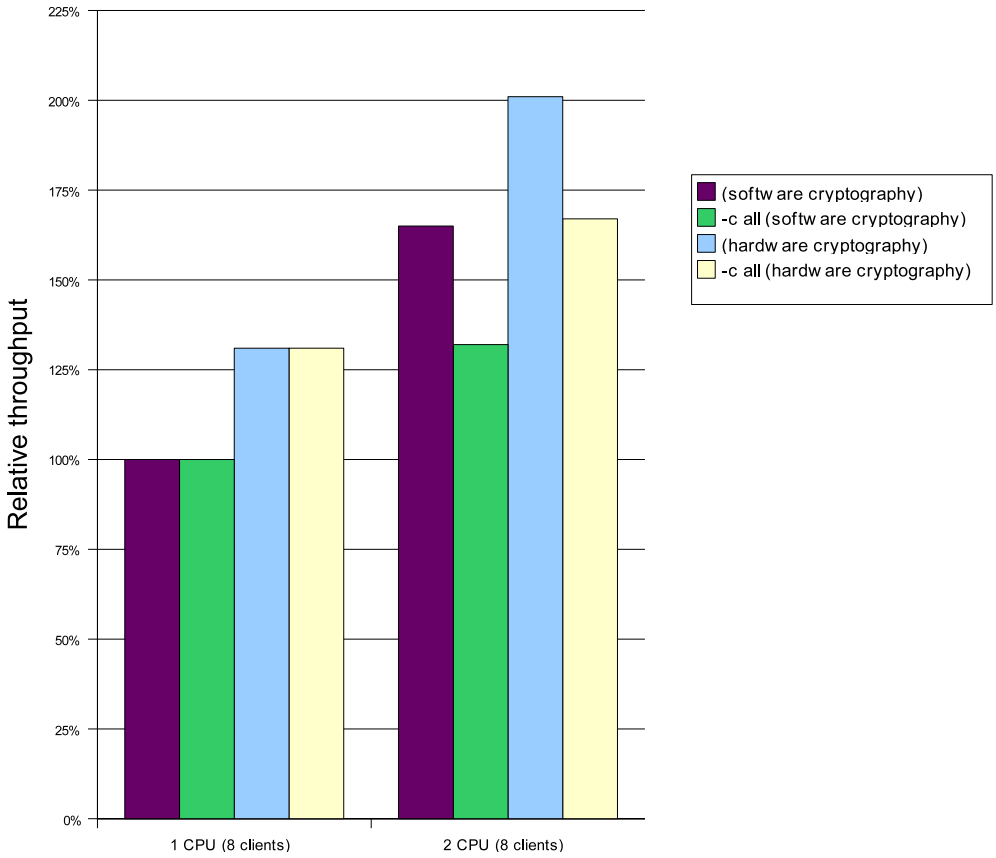


*Figure 11. 5.8 KB SSL page access through a TCP junction versus 5.8 KB SSL page access through a TCP with -c all*

**Observations**

The case with a single CPU shows only the improvement of using hardware cryptography. The additional workload had nearly no impact on the scenarios with one CPU as without -c all option because these runs are CPU constrained. The more interesting case is when two CPUs are used. In this case, the throughput degraded by 17% using cryptographic hardware even though there is some CPU capacity available. This indicated that the access to the Tivoli Access Manager introduced some latencies.

**Conclusions**

In the one CPU case, the -c all option does not have an impact on the throughput because WebSEAL is CPU constrained. In the two CPU case, WebSEAL is not CPU constrained and there is a small degradation in throughput as would be expected due to the additional processing required in WebSEAL to support this feature.

**Related information**

Tivoli WebSEAL - hardware equipment and software environment
To perform our Tivoli WebSEAL performance tests, we created a customer-like environment. We configured the hardware and software, the network, the storage server, and the z/VM guests.

Tivoli WebSEAL - test case setup
To perform our Tivoli WebSEAL performance tests, we identified and set up the software and hardware for various test case scenarios. We set up our DB2 server, set up three levels of security, created a WebSEAL junction, put our Web pages in the correct directories, created ACL descriptions, and used the System z9 cryptographic hardware functionality.

Tivoli WebSEAL - detailed set up examples
The detailed setup we performed to enable WebSEAL with cryptographic hardware support is provided here.

Tivoli WebSEAL - other sources of information
Additional resources to provide information on the products, hardware, and software discussed in this paper can be found in various books and at various Web sites.

# SSL page access through an SSL junction

In this test case the client has an SSL connection to the WebSEAL server and accessed the Web page on the back end WebSphere Application Server via an SSL connection (SSL junction).

## Using page sizes of 5.8 KB and 12 KB

This test involves two SSL accesses comparing software encryption against hardware encryption. Encryption occurs between the client and WebSEAL on the front end and between WebSEAL and the back end server. We initially tried WebSphere Application Server running on z/OS as the back end server, however, we were not able to get the WebSphere Application Server to select DES-168 as the cipher suite. To get around this problem, we used the Apache HTTP server under Linux running on a separate LPAR as the back end server. The DES-168 encryption algorithm was used for encryption operations between the client and WebSEAL and WebSEAL and Apache Web server. In this test only one handshake occurs.

**Encryption algorithm**: DES-168

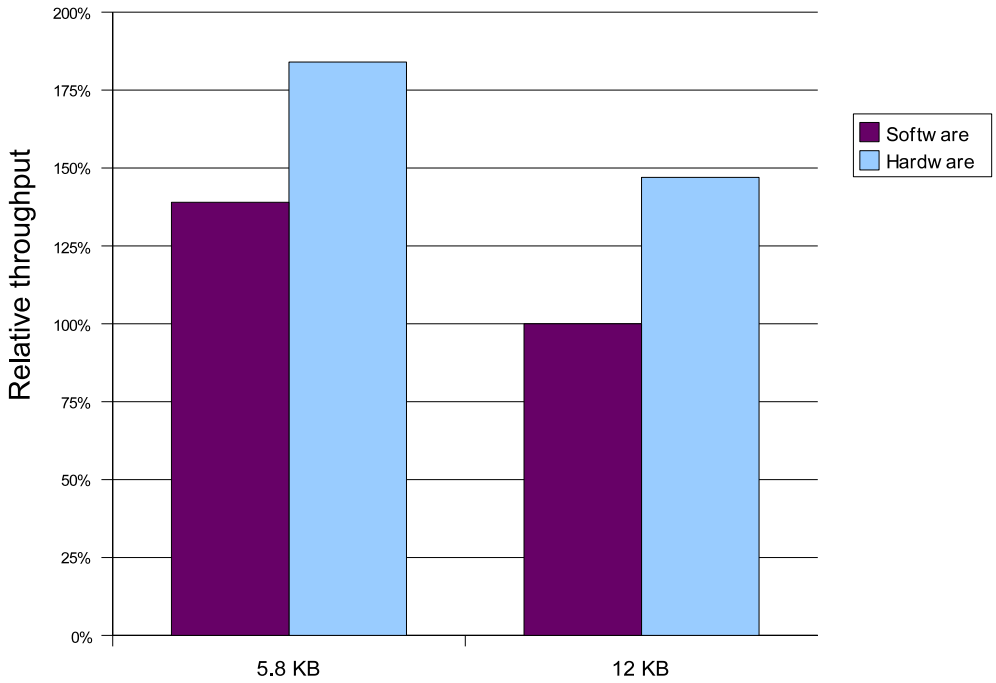


*Figure 12. 5.8 KB and 12 KB SSL page access through an SSL junction*

**Observations**

The improvement of hardware encryption was, again, between 30% and 50%. The degradation of the throughput compared with SSL through a TCP junction (see Figure 10 on page 18) was about 60%. The CPU load on the WebSEAL system was high, between 80% and 95%.

**Conclusion**

The WebSEAL server has much more work to do in this scenario, the encrypted packages have to be decrypted, analyzed, and encrypted again. This leads to a significant degradation of the throughput, which could be reduced by using hardware support for encryption. Be aware that the alternative to this overhead, like an isolated network using a HiperSockets connection or a z/VM guest LAN, can be easily implemented on System z and produces the same grade of security without this overhead.

**Related information**

Tivoli WebSEAL - hardware equipment and software environment
To perform our Tivoli WebSEAL performance tests, we created a customer-like environment. We configured the hardware and software, the network, the storage server, and the z/VM guests.

Tivoli WebSEAL - test case setup
To perform our Tivoli WebSEAL performance tests, we identified and set up the software and hardware for various test case scenarios. We set up our DB2 server, set up three levels of security, created a WebSEAL junction, put our Web pages in the correct directories, created ACL descriptions, and used the System z9 cryptographic hardware functionality.

Tivoli WebSEAL - detailed set up examples
The detailed setup we performed to enable WebSEAL with cryptographic hardware support is provided here.

Tivoli WebSEAL - other sources of information
Additional resources to provide information on the products, hardware, and software discussed in this paper can be found in various books and at various Web sites.

## SSL page access with new authentication and new SSL session every request

In this test case the client has an SSL connection to the WebSEAL server, which requires authentication and encryption, and accesses the Web page on the back end WebSphere Application Server via an unencrypted connection (TCP junction). These transactions involve a handshake for every transaction and a transfer on 100 (encrypted) bytes.

### Using a page size of 100 B

**Encryption algorithm**: AES-128

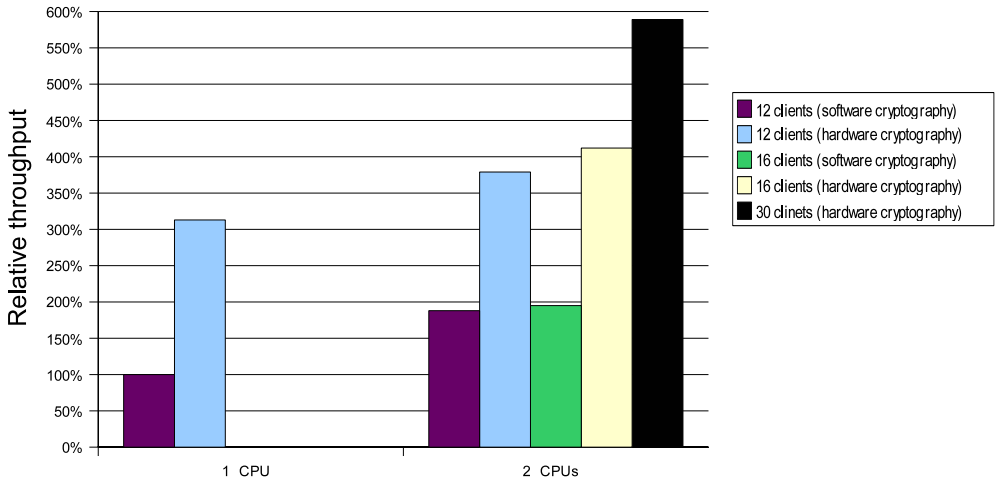


*Figure 13. 100 B SSL page access BA*

This test case puts a large load on the client machines. The client encryption and decryption is done in software and as the number of clients on a single machine increases, a single client becomes a bottleneck. For one of our test cases, four client machines were used. Performance Toolkit for z/VM showed the utilization on the cryptographic accelerator cards as follows:

- Total operations: 119589
- Utilization: 18.7%

Table 6 shows the results of collecting statistics on the cryptographic card every 5 seconds during the workload execution. We were interested in the length of the pending operations queue.

*Table 6. Collection statistics on the cryptographic card*

| **Pending Queue Length** | | | | | | | | | | | **Total Operations** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Average | Card 1 | Card 2 |
| Frequency | 4 | 13 | 16 | 10 | 3 | 7 | 2 | 1 | 4 | 2.88 | 63949 | 55640 |

**Observations**

Using software encryption leads to a complete utilization of the WebSEAL server with one CPU. Even with hardware encryption, the system is almost fully utilized, therefore, we did not increase the load further. Increasing the number of CPUs from one to two resulted in an improvement of 88% for the same workload. Here we have the biggest improvement when using hardware encryption support. It is between a factor of two and a factor of three for the same workloads. The average queue length of the cryptographic cards of 2.88 shows that the cards are under a significant load.

**Conclusion**

This test does only authentication with very small pages to generate the maximum overhead. Both hardware cryptographic features, the CPACF from the CPU for decryption and encryption of the requests and the CEX2C cryptographic adapter feature for the authentication are used. This enables the workload to be increased beyond what is possible using software encryption. In the last test we were able to improve the total throughput of the system by about a factor of three by using hardware cryptographic support.

**Related information**

Tivoli WebSEAL - hardware equipment and software environment
To perform our Tivoli WebSEAL performance tests, we created a customer-like environment. We configured the hardware and software, the network, the storage server, and the z/VM guests.

Tivoli WebSEAL - test case setup
To perform our Tivoli WebSEAL performance tests, we identified and set up the software and hardware for various test case scenarios. We set up our DB2 server, set up three levels of security, created a WebSEAL junction, put our Web pages in the correct directories, created ACL descriptions, and used the System z9 cryptographic hardware functionality.

Tivoli WebSEAL - detailed set up examples
The detailed setup we performed to enable WebSEAL with cryptographic hardware support is provided here.

Tivoli WebSEAL - other sources of information
Additional resources to provide information on the products, hardware, and software discussed in this paper can be found in various books and at various Web sites.

# Detailed set up examples for the Tivoli WebSEAL performance tests

The detailed setup we performed to enable WebSEAL with cryptographic hardware support is provided here.

## Enable and run WebSEAL with cryptographic hardware support

Use the following steps to enable and run WebSEAL with cryptographic hardware support.

1. Load the System z9 crypto device driver using the following command (performed at each system startup):

   ```
   /etc/init.d/z90crypt start
   ```

2. Start the PKCS11 subsystem using the following command (performed at each system startup):

   ```
   /etc/init.d/pkcsslotd start
   ```

3. Configure the PKCS#11 device by doing the following (this is performed once)
   a. Initialize the token using the following command:

      ```
      /usr/lib/pkcs11/methods/pkcsconf -c 0 -I
      ```

   b. When prompted, enter the default Security Officer (SO) PIN (87654321) and a unique token label as shown below:

      ```
      /usr/lib/pkcs11/methods/pkcsconf -c 0 -I
      Enter the SO PIN: ********
      Enter a unique token label: websealserver
      ```

      For our example, 87654321 was the SO PIN default password and a token label name of "websealserver" was used.

   c. Set the SO PIN to a value different from the default (87654321) as show below:

```
/usr/lib/pkcs11/methods/pkcsconf -c 0 -P
Enter the SO PIN: ********         87654321
Enter the new SO PIN: ********     76543210
Re-enter the new SO PIN: ********  76543210
```

"76543210" is just an example. You can choose any number you wish.

d. Set the user PIN as shown below:

```
/usr/lib/pkcs11/methods/pkcsconf -c 0 –u
Enter the SO PIN: ********            76543210
Enter the new user PIN: ********      12345678
Re-enter the new user PIN: ********   12345678
```

e. Change the user PIN as shown below:

```
/usr/lib/pkcs11/methods/pkcsconf -c 0 -p
Enter user PIN: ********              12345678
Enter the new user PIN: ********      01234567
Re-enter the new user PIN: ********   01234567
```

4. Create a self-signed certificate using the PKCS#11 token, to be used by WebSEAL by doing the following (this is performed once). Use the gsk7ikm utility to add a self-signed certificate to the CMS Cryptographic Token.

   **Note:** Before starting the gsk7ikm utility, set the JAVA_HOME environment variable to point to your Java™ JRE and ensure Java is in the PATH as shown below:

   export JAVA_HOME=/opt/IBMJava2-s390-142/jre/
   export PATH=/opt/IBMJava2-s390-142/jre/bin:$PATH
   gsk7ikm

   a. Run gsk7ikm.

      The following screen appears.

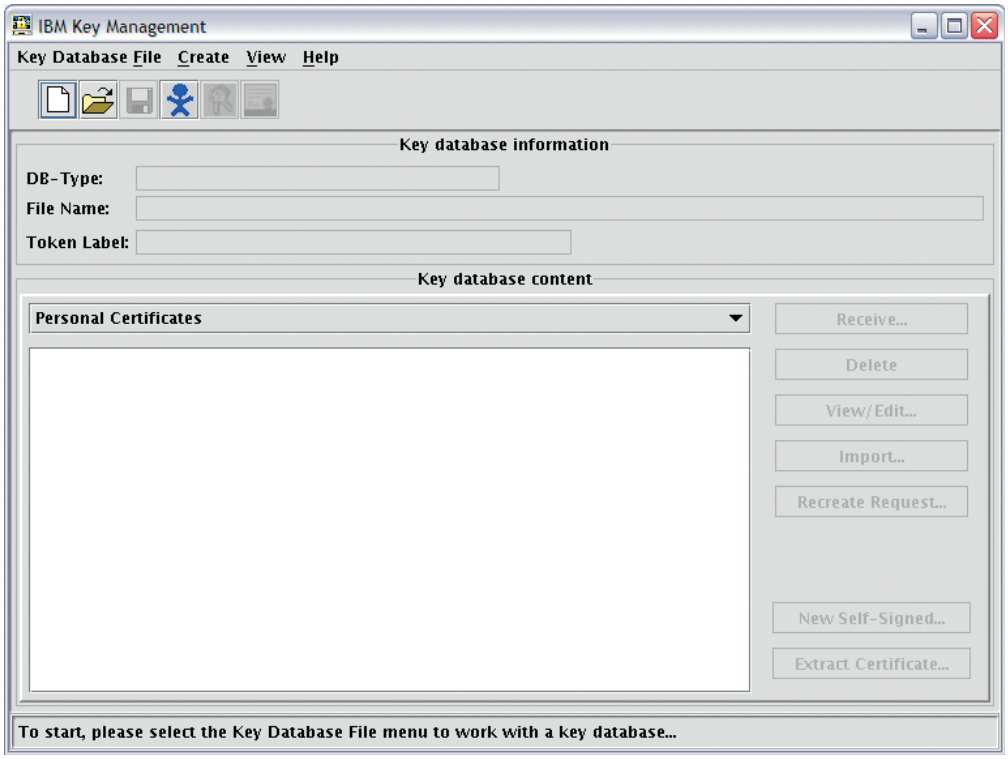


*Figure 14. IBM key management screen*

1) From Figure 14, select **Key Database File** –> **Open**.
2) The following box opens.

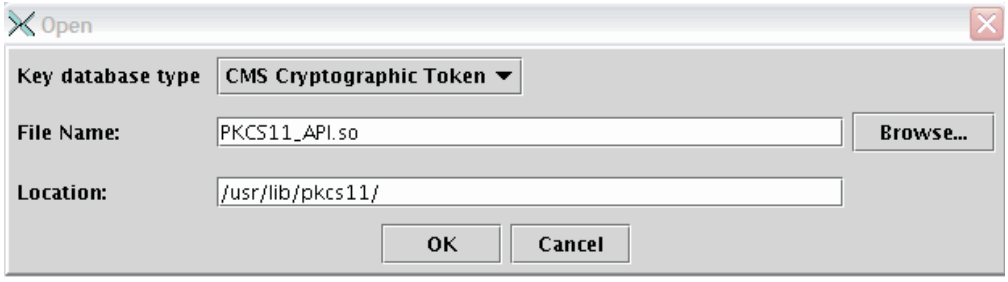


*Figure 15. Key database file entry box*

1) Select the Key Database Type of CMS Cryptographic Token.
2) Enter PKCS11_API.so for the file name.
3) Enter "/usr/lib/pkcs11" for the location.
4) Click the OK button and the following screen appears:

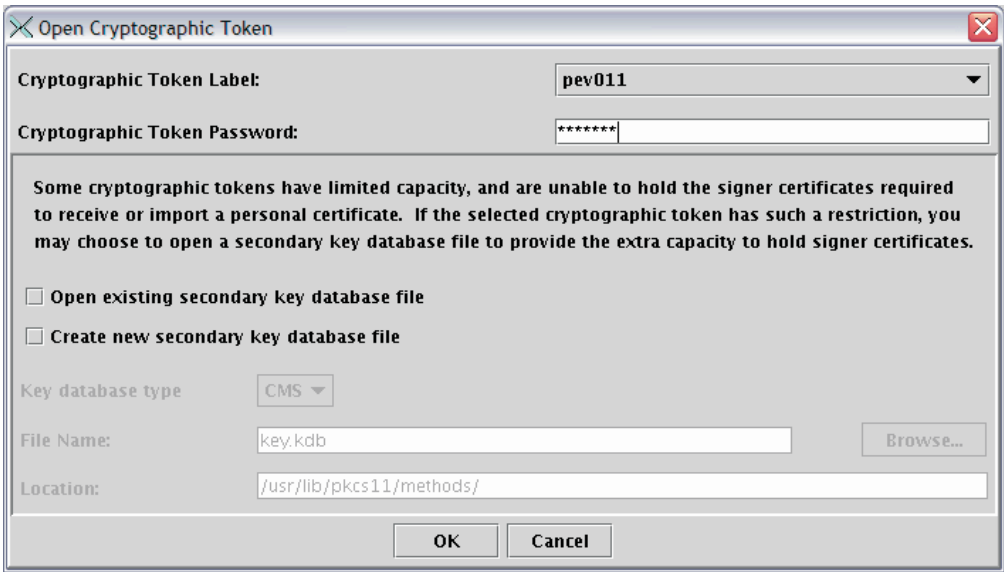


*Figure 16. Open cryptographic token screen*

1) Enter the cryptographic token password that you set before (01234567).
2) Deselect the Open existing secondary key database file box.
3) Click OK.
4) On the IBM key management screen (see Figure 14 on page 26), click on New Self-Signed...

The following screen is displayed:

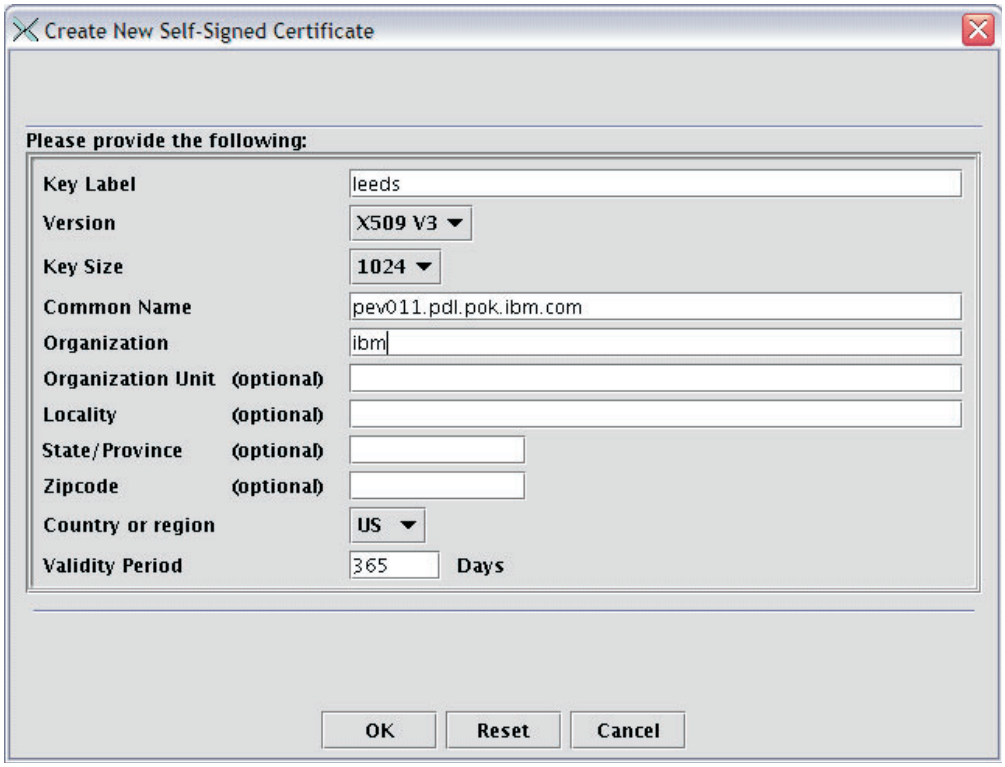


*Figure 17. Create new self-signed certificate screen*

1) Enter the appropriate information to create the certificate.
2) Click OK.

The self-signed certificate you just created will now be visible on the IBM Key Management screen as show below.

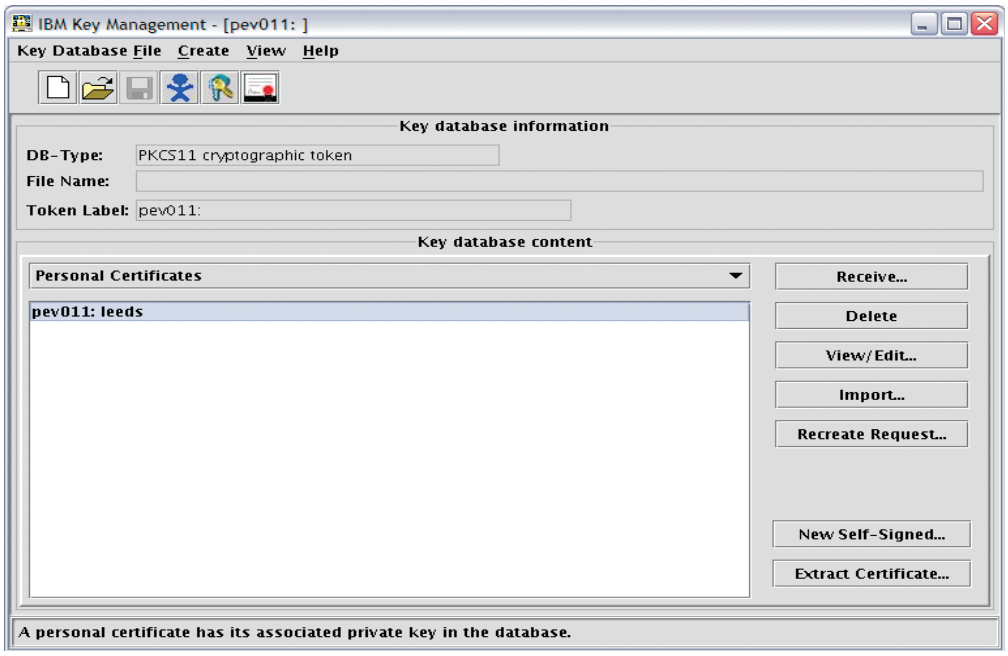


*Figure 18. IBM Key Management screen showing self-signed certificate*

Exit the gsk7ikm application.

5. Configure WebSEAl to use the PKCS#11 devices for cryptographic operations as detailed below (this is performed once):
   - Add ivmgr to group pkcs11 by issuing the following:

     usermod -G ivmgr,tivoli,pkcs11 ivmgr
   - Update the following parameters in the webseald-xxx.conf file (in our case webseald-default.conf):

     unix-group = pkcs11
     webseal-cert-keyfile-label = websealserver:leeds
     [The value for the webseal-cert-keyfile-label directive has the form – token_label:certificate_label]
     [ssl]
     pkcs11-driver-path = /usr/lib/pkcs11/PKCS11_API.so
     pkcs11-token-label = websealserver
     pkcs11-token-pwd = 01234567 (user PIN set during token initialization)
     pkcs11-symmetric-cipher-support = yes (this directive enables the use of CPACF)

     [ssl-qop]

     ssl-qop-mgmt = yes

```
[ssl-qop-mgmt-default]
default = AES-128 or
   DES-168
```

6. Start or re-start WebSEAL using one of the following commands:

```
pdweb start      or
pdweb restart
```

## Other sources of information for the Tivoli WebSEAL performance tests

Additional resources to provide information on the products, hardware, and software discussed in this paper can be found in various books and at various Web sites.

- For information on WebSphere Application Server see:
  http://www.ibm.com/software/info1/websphere/index.jsp?tab=products/apptransaction
- For information on Linux on System z see:
  www.ibm.com/systems/z/os/linux
- For information on z/VM see:
  www.vm.ibm.com
- For information on installing WebSEAL see:
  http://www-03.ibm.com/systems/z/os/linux/utilities/pdf/lu26st00.pdf
- For information about the Linux utilities for IBM System z see:
  http://www-03.ibm.com/systems/z/os/linux/utilities/
- For information on IBM open source projects see:
  www.ibm.com/developerworks/opensource/index.html
- For information on hardware encryption see the IBM Red Book *Using Cryptographic Adapters for Web Servers with Linux on IBM System z9 and zSeries*® at:
  http://www.redbooks.ibm.com/redpapers/pdfs/redp4131.pdf
- For general information on the software products used in this test, see the IBM red paper *IBM Tivoli Access Manager for e-business* at:
  http://www.redbooks.ibm.com/redpapers/pdfs/redp3677.pdf

## Notices for the Tivoli WebSEAL performance tests

Additional resources to provide information on the products, hardware, and software discussed in this paper can be found in various books and various Web sites.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Intel and Xeon are trademarks of Intel Corporation in the United States, other countries or both.

Linux is a registered trademark of Linus Torvalds in the United States and other countries.

Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product and service names may be trademarks or service marks of others.

Information concerning non-IBM products was obtained from the suppliers of their products or their published announcements. Questions on the capabilities of the non-IBM products should be addressed with the suppliers.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.