Websphere Application Server V8 for Linux on System z
SSL Setup & Performance Study

Thomas Weber, IBM Germany R&D
System z Performance Analyst
Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

The following are trademarks or registered trademarks of other companies.

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- SUSE is a registered trademark of Novell, Inc. in the United States and other countries.
- Red Hat, Red Hat Enterprise Linux, the Shadowman logo and JBoss are registered trademarks of Red Hat, Inc. in the U.S. and other countries.
- Oracle and Java are registered trademarks of Oracle and/or its affiliates in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.
Overview of cryptographic terms used

**SSL/TLS** – SSL (Secure Socket Layer) and its successor TLS (Transport Layer Security) are protocols for encrypting data transfers over a network.

**RSA** - asymmetric algorithm used for public-key cryptography. The RSA key size (e.g. 2048 bits) defines the security strength of the algorithm.

**AES** – symmetric algorithm used for data encryption/decryption during network data transmission after the SSL connection is established.

**CPACF** – System z **CP** Assist for Cryptographic Function is a feature on the Central Processor unit to accelerate symmetric cryptographic and hash functions.

**CEX** – System z **Crypto Express** feature provides support for asymmetric cryptographic operations in secure and clear key mode in case of SSL and TLS.
Linux on System z end-to-end project: Websphere Application Server (WAS) V8 SSL setup & performance study

Setup study:
• **Scenario 1:** WAS V8 SSL setup for securing network communications
• **Scenario 2:** IBM HTTP Server (IHS) SSL + WAS setup for securing network communications
• usage of System z cryptographic hardware features (CPACF, CEX)

Performance study:
• SSL cryptographic operations in software mode only
• SSL cryptographic operations supported by CPACF
• SSL cryptographic operations supported by CPACF and CEX
  • CEX configured as SSL Accelerator (CEX3A)
  • CEX configured as cryptographic Co-Processor (CEX3C)
• results for different SSL RSA key sizes (2048 and 4096 bits)
# Overview of the used System z cryptographic hardware

<table>
<thead>
<tr>
<th>System z cryptographic hardware feature</th>
<th>symmetric crypto operations</th>
<th>asymmetric crypto operations</th>
<th>hash functions MACs</th>
<th>random number generation</th>
<th>secure key crypto</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPACF</td>
<td>DES</td>
<td>SHA-1</td>
<td>pseudo RNG</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3DES</td>
<td>SHA-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AES</td>
<td>CMAC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crypto Express Accelerator</td>
<td></td>
<td>RSA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crypto Express Coprocessor</td>
<td>RSA</td>
<td>true RNG</td>
<td>via CCA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ECC (via CCA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* green – used in this project
DayTrader benchmark application

- Open Source benchmark application
- emulates an Online Stock Trading System
- end-to-end Java EE web application
- IBM WAS is a Java EE application server

http://geronimo.apache.org/GMOxDOC30/daytrader-a-more-complex-application.html
Scenario 1:
IBM WAS with internal HTTP transport – setup overview
IBM WAS with internal HTTP transport – cryptographic overview

WebSphere Application Server 8

- IBMPKCS#11Impl Provider

Source:
- If applicable, describe source origin
  - System z CP
  - CPACF
  - Crypto Express feature (CEX)

Application

- libICA library
- ICA token
- PKCS #11 API
- openCryptoki

shared libraries

Linux kernel

- zcrypt device driver

cryptographic hardware

- requires setup
- requires setup

© 2014 IBM Corporation
Configure Linux on System z cryptographic hardware support for IBM WAS V8 SSL (1)

Enable System z cryptographic hardware

- **CPACF**
  - CP Assist for Cryptographic Functions is available to the IBM System z Processor Unit (PU)
  - must be enabled per feature code
  - accessible from all LPARs

- **CEX**
  - additional crypto feature
  - Crypto Express cards can be shared among selected LPARs
  - LPARs must be assigned to CEX cards using the SE or HMC customize image profiles task
Configure Linux on System z cryptographic hardware support for IBM WAS V8 SSL (2)

Linux packages required:

- **openCryptoki**
- **openCryptoki-64bit**
  - PKCS #11 API implementation for Linux
  - interface between cryptographic hardware and user space applications

- **libica**
  - library for IBM Cryptographic Architecture (libICA)
  - provides interface library routines used by modules to interface with IBM cryptographic hardware
Configure Linux on System z cryptographic hardware support for IBM WAS V8 SSL (3)

IBM Linux on System z zcrypt device driver

• required when one or more System z Crypto Express (CEX) features are accessible in a LPAR or z/VM guest
• zcrypt device driver must be loaded (SLES11: rcz90crypt start)
- lszcrypt* command shows the status of the available CEX features

```
# lszcrypt -V
card02: CEX3A online
card03: CEX3A online
```

- chzcrypt* command controls any available CEX features

```
# chzcrypt -d 02
# lszcrypt -V
card02: CEX3A offline
card03: CEX3A online
```

* s390-tools package
Configure Linux on System z cryptographic hardware support for IBM WAS V8 SSL (4)

CP Assist for Cryptographic Function (CPACF) support
• IBM WAS and IHS use the openCryptoki and libICA library on behalf to access System z cryptographic hardware
• icainfo (libica package) command lists supported CPACF ciphers

# icainfo
The following CP Assist for Cryptographic Function (CPACF) operations are supported by libica on this system:
SHA-1: yes
SHA-256: yes
SHA-512: yes
DES: yes
TDES-128: yes
TDES-192: yes
AES-128: yes
AES-192: yes
AES-256: yes
PRNG: yes
CCM-AES-128: yes
CMAC-AES-128: yes
CMAC-AES-192: yes
CMAC-AES-256: yes
Configure Linux on System z cryptographic hardware support for IBM WAS V8 SSL (5)

Slot manager daemon for openCryptoki (pkcsslotd) – ICA token

- daemon must be running (e.g. rcpkcsslotd start)
- 'PKCS#11 cryptographic ICA token' must be initialized using the `pkcsconf` command (openCryptoki package)
- display token info shows a not yet initialized token below

```
# pkcsconf -t
Token #0 Info:
Label: IBM ICA PKCS #11
Manufacturer: IBM Corp.
Model: IBM ICA
Serial Number: 123
Flags: 0x880045 (RNG|LOGIN_REQUIRED|CLOCK_ON_TOKEN|
USER_PIN_TO_BE_CHANGED|SO_PIN_TO_BE_CHANGED)
Sessions: 0/2
R/W Sessions: -1/-2
PIN Length: 4-8
Public Memory: 0xFFFFFFFF/0xFFFFFFFF
Private Memory: 0xFFFFFFFF/0xFFFFFFFF
Hardware Version: 1.0
Firmware Version: 1.0
Time: 13:32:27
```
Configure Linux on System z cryptographic hardware support for IBM WAS V8 SSL (6)

slot manager daemon for openCryptoki (pkcsslotd) – ICA token

- command sequence to initialize the 'PKCS#11 cryptographic ICA token'

- initialize the ICA token (-c specifies the ICA token slot)
  # pkcsconf -c 0 -I

- set a new Security Officer (SO) PIN
  # pkcsconf -c 0 -P

- initialize and set a new User PIN
  # pkcsconf -c 0 -u
  # pkcsconf -c 0 -p
Configure Linux on System z cryptographic hardware support for IBM WAS V8 SSL (7)

slot manager daemon for openCryptoki – pkcsslotd

• list the fully initialized 'PKCS#11 cryptographic ICA token'

    # pkcsconf -t
    Token #0 Info:
    Label: IBMICATOK
    Manufacturer: IBM Corp.
    Model: IBM ICA
    Serial Number: 123
    Flags: 0x44D (RNG|LOGIN_REQUIRED|USER_PIN_INITIALIZED|CLOCK_ON_TOKEN|TOKEN_INITIALIZED)
    Sessions: 0/-2
    R/W Sessions: -1/-2
    PIN Length: 4-8
    Public Memory: 0xFFFFFFFF/0xFFFFFFFF
    Private Memory: 0xFFFFFFFF/0xFFFFFFFF
    Hardware Version: 1.0
    Firmware Version: 1.0
    Time: 15:37:35
IBM WAS with internal HTTP transport – cryptographic overview

**WebSphere Application Server 8**

- IBMPKCS#11Impl Provider

**Source:**
- If applicable, describe source origin
  - System z CP
  - CPACF
  - Crypto Express feature (CEX)

**LibIAC library**

- ICA token
- PKCS #11 API
- openCryptoki

**Zcrypt device driver**

- done
- requires setup

**Application**

- shared libraries

**Linux kernel**

- cryptographic hardware
Configure IBMPKCS11Impl Provider for IBM WAS V8 SSL support (1)

• update the IBMPKCS11Impl Provider *Java Security properties file* ( {WAS home dir}/java/jre/lib/security/java.security )
  - add the IBMPKCS11Impl Provider at the top of list provider list
  - attach the path to the file holding the PKCS#11 token information

```
# # List of providers and their preference orders (see above):
#
security.provider.1=com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl /etc/cex3config.cfg
security.provider.2=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.3=com.ibm.crypto.provider.IBMJCE
security.provider.4=com.ibm.jsse.IBMJSSEProvider
security.provider.5=com.ibm.jsse2.IBMJSSEProvider2
security.provider.6=com.ibm.security.jgss.IBMJGSSProvider
security.provider.7=com.ibm.security.cert.IBMCertPath
security.provider.8=com.ibm.security.cmskeystore.CMSProvider
...```
Configure IBMPKCS11Impl Provider for IBM WAS V8 SSL support (2)

- sample PKCS#11 token configuration file

```bash
# cat /etc/cex3config.cfg
name = IBMICATOK ← name of the ICA token label
description = config for IBM Crypto Express 3 (configured as an ICA token)
library = /usr/lib/pkcs11/PKCS11_API.so64 ← path to PKCS#11 library
SlotListIndex = 0 ← number of the PKCS#11 ICA token slot
disabledMechanisms = {
  CKM_MD5
  CKM_SHA_1
  CKM_MD5_HMAC
  CKM_SHA_1_HMAC
  CKM_SSL3_MASTER_KEY_DERIVE
  CKM_SSL3_KEY_AND_MAC_DERIVE
  CKM_SSL3_PRE_MASTER_KEY_GEN ← list of PKCS#11 mechanisms to disable
}
```
IBM WAS V8 with internal HTTP transport - SSL setup (1)

Adding a user to the PKCS#11 group

- non-root users running WAS using the PKCS#11 API must belong to the pkcs11 group
- for example WAS running under a non-root user (e.g. wasadmin)
- root user is automatically added when `pkcs11_startup` command is called for the first time (SLES11: done in the in pkcsslotd startup script)

- sample: add the 'wasadmin' user to the pkcs11 group

```
# grep pkcs11 /etc/group
pkcs11:!:64:root
# usermod -G pkcs11 wasadmin
```
Update the Java JCE policy files

• IBM WAS ships its own Java environment (JRE) with strong but limited Java Cryptography Extension (JCE) policy files
  • limited RSA key sizes
  • limited cipher support (e.g. AES-128 vs AES-256)

• requires replacement of JAR files placed in the JRE's directory
  jre/lib/security/

For further details about JCE policy files, see:

Select your Java version and search for IBM SDK Policy files.
IBM WAS V8 with internal HTTP transport - SSL setup (3)

Select a supported cipher suite for hardware acceleration
Example: AES-256 + RSA

• Hardware support depends on System z machine and Linux distribution level
- check that AES-256 is supported by CPACF (*icainfo* command)
  → AES-256: yes
- RSA is supported with the CEX feature

<table>
<thead>
<tr>
<th>WAS V8 cipher suite</th>
<th>IBM System z cryptographic stack support (SLES11 SP2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL_RSA_WITH_AES_256_CBC_SHA</td>
<td>full support</td>
</tr>
</tbody>
</table>
| SSL_DHE_RSA_WITH_AES_256_CBC_SHA | partially supported
DHE-RSA in software; AES in hardware |
| SSL_RSA_WITH_AES_256_CBC_SHA256 | not supported
currently no support for SHA-256 in opencryptoki |
| SSL_ECDH_RSA_WITH_AES_256_CBC_SHA | ECDH-RSA not supported |
| SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA | ECDHE-RSA not supported |
Select a supported cipher suite for hardware acceleration

SSL certificate and key management

SSL certificate and key management | SSL configurations | NodeDefaultSSLSettings | Quality of protection (QoP) settings

Specifies the security level, ciphers, and mutual authentication settings.

General Properties

Client authentication
None

Protocol
SSL_TLS

Provider

Predefined JSSE provider
Select provider
IBM/JSSE

Custom JSSE provider
Custom provider

Cipher suite settings

Cipher suite groups
Custom

Update selected ciphers

Cipher suites

SSL_RSA_WITH_NULL_MD5
SSL_RSA_WITH_NULL_SHA
SSL_RSA_WITH_NULL_SHA256
SSL_ECDH_ECDSA_WITH_NULL_SHA
SSL_ECDH_RSA_WITH_NULL_SHA

Selected ciphers
SSL_RSA_WITH_AES_256_CBC_SHA

Add
<< Remove
IBM WAS V8 with internal HTTP transport - SSL setup

Verify the SSL setup (1)

• try to access your application via SSL

```bash
# curl -k -v https://wasnode1.net:9443/daytrader ← int. WAS SSL port
About to connect() to wasnode1.net port 9443 (#0)
Trying 10.x.x.x... connected
Connected to wasnode1.net (10.x.x.x) port 9443 (#0)
successfully set certificate verify locations:
CAfile: none
CApath: /etc/ssl/certs/
SSLv3, TLS handshake, Client hello (1):
SSLv3, TLS handshake, Server hello (2):
SSLv3, TLS handshake, CERT (11):
SSLv3, TLS handshake, Server finished (14):
SSLv3, TLS handshake, Client key exchange (16):
SSLv3, TLS change cipher, Client hello (1):
SSLv3, TLS handshake, Finished (20):
SSLv3, TLS change cipher, Client hello (1):
SSLv3, TLS handshake, Finished (20):
SSL connection using AES256-SHA ← used cipher suite
```
IBM WAS V8 with internal HTTP transport - SSL setup

Verify the SSL setup (2)

• Do we really use cryptographic hardware?
  - *icastats* command shows libICA statistics *during* application execution

<table>
<thead>
<tr>
<th># icastats function</th>
<th># hardware</th>
<th># software</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>12</td>
<td>0</td>
<td>supported by CPACF</td>
</tr>
<tr>
<td>SHA-224</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SHA-256</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SHA-384</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SHA-512</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RANDOM</td>
<td>36</td>
<td>0</td>
<td>supported by CPACF(pseudo) or CEX3C(true)</td>
</tr>
<tr>
<td>MOD EXPO</td>
<td>7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RSA CRT</td>
<td>62</td>
<td>0</td>
<td>supported by CEX3A/C</td>
</tr>
<tr>
<td>DES ENC</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DES DEC</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3DES ENC</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3DES DEC</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>AES ENC</td>
<td>94</td>
<td>0</td>
<td>supported by CPACF</td>
</tr>
<tr>
<td>AES DEC</td>
<td>93</td>
<td>0</td>
<td>supported by CPACF</td>
</tr>
<tr>
<td>CMAC GEN</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CMAC VER</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
IBM WAS V8 with internal HTTP transport - SSL setup

Verify the SSL setup (3)

- `lszcrypt` command shows statistics for any available Crypto Express cards
  - no parameters given shows available CEX cards
    
    ```bash
    # lszcrypt
    card02: CEX3A
    card03: CEX3A
    ```
  
  - verbose level 1 shows status for the CEX cards
    
    ```bash
    # lszcrypt -V
    card02: CEX3A   online
    card03: CEX3A   online
    ```
  
  - verbose level 2 shows request count for the CEX cards
    
    ```bash
    # lszcrypt -VV
    card02: CEX3A  online  hwtype=9  depth=8  request_count=369228
    card03: CEX3A  online  hwtype=9  depth=8  request_count=373015
    ```
• transaction throughput nearly doubles with CEX3 cards
• more than 20% throughput increase with CPACF
• CPUs almost fully utilized for all test cases
• all CEX3 test cases include CPACF feature
Result for scenario 1: WAS only - RSA key 4096 bits
SSL transaction throughput (normalized)

- Transaction throughput increases up to 3x with one CEX3 card
- Transaction throughput increases 5x with two CEX3 cards
- CPUs not fully utilized when CEX3 processors are used
• use of System z cryptographic features reduces CPU costs at higher throughput rates
• pure software cryptographic operations are extremely CPU cost expensive
average response times for the SSL transactions (RSA key 4096 bits)

• avg response time below 5 ms for the CEX3 card setups
• transactions are processed faster with System z cryptographic hardware enabled on the application server
Scenario 2: IBM WAS with IBM HTTP server – setup overview

**Tier one**
- **System x445**
  - 8 CPU 2.7GHz
  - 8 GB memory
- WebSphere Studio Workload Simulator

10 GbE network

SSL conn.

**Tier two**
- **LPAR1**
  - 4 CPU
  - 8GB memory
  - 2x CEX3-2P
  - 10GbE OSA
  - SLES11 SP2

- **IBM HTTP Server 8**
  - Websphere Application Server V8
  - DayTrader application

**Tier three**
- **LPAR2**
  - 4 CPU
  - 8GB memory
  - SLES11 SP2

- **Database Server**
  - DB2 V9.7

- **DayTrader database**

**Hipersocket MTU 8k**

**DS8800**

- Ficon
- Hyper PAV
IBM WAS with IBM HTTP server – cryptographic overview

WebSphere Application Server 8

IBM HTTP Server 8

- ICC
- GSKit 8

libICA library

ICA token

PKCS #11 API
openCryptoki

* CMS keystore

Application

shared libraries

zcrypt device driver

Linux kernel

cryptographic hardware

System z CP

CPACF

Crypto Express3 feature (CEX3)

libICA library

ICAC token

PKCS #11 API
openCryptoki

* CMS keystore

Application

shared libraries

zcrypt device driver

Linux kernel

cryptographic hardware

System z CP

CPACF

Crypto Express3 feature (CEX3)
Differences when using IBM HTTP Server instead of internal WAS HTTP transport

• zcrypt device driver handling is the same
• PKCS#11 (opencryptoki) ICA token configuration is the same

Differences:
• IHS uses the Global Secure ToolKit API (GSKit) instead of the IBMPKCS11ImplProvider

• SSL definitions are added to the IHS configuration file /opt/IBM/HTTPServer/conf/httpd.conf

• SSL certificates stored in the ICA token (PKCS12 keystore for WAS)
IBM WAS V8 with IBM HTTP server - SSL setup

Sample common SSL/TLS configuration for IHS version 8 (1)

# Example SSL(TLS) configuration
#
# added due to conflicting GSKit8 and openSSL libraries
LoadFile /usr/lib64/libcrypto.so ← added to prevent SSL init failures
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
Listen 443
<VirtualHost wasnode1.net:443>
ServerName wasnode1.net
SSLEnable
SSLProtocolDisable SSLv2 ← disable SSL protocol versions
SSLProtocolDisable SSLv3 to force TLS usage
# cipher suite TLS_RSA_WITH_AES_256_CBC_SHA(35b)
# remove all ciphers first
SSLCipherSpec ALL NONE ← reset cipher suites list
SSLCipherSpec ALL +TLS_RSA_WITH_AES_256_CBC_SHA ← add cipher suites
</VirtualHost>
...
IBM WAS V8 with IBM HTTP server - SSL setup

Sample SSL/TLS PKCS#11 configuration for IHS version 8 (2)

...  
# PKCS#11 configuration  
KeyFile /opt/IBM/HTTPServer/ssl/key.kdb ← CMS keystore for signer certificates

SSLServerCert IBMICATOK:ihsCert ← use server certificate stored in PKCS#11 ICA token

SSLStashfile /opt/IBM/HTTPServer/ssl/ibmicatok.sth ← password file with stashed PKCS#11 ICA token user PIN

SSLPKCSDriver /usr/lib/pkcs11/PKCS11_API.so64 ← Fully qualified name of the PKCS#11 library module

SSLDisable

SSLCachePortFilename /opt/IBM/HTTPServer/logs/siddport  
# End of SSL configuration
Questions?

- **Further information**
  - More detailed description is in the available White Paper (covers also the IBM HTTP Server setup)
    “IBM Websphere Application Server Version 8 for Linux on IBM System z - SSL Setup and Performance Study”
    http://www.ibm.com/developerworks/linux/linux390/perf/tuning_security.html#ssl
  - Linux on System z – Tuning hints and tips
  - Live Virtual Classes for z/VM and Linux
    http://www.vm.ibm.com/education/lvc/

---

**Thomas Weber**  
*Linux on IBM System z  
Performance Analyst*  

**IBM Deutschland  
Research & Development**  
*Schoenaicher Strasse 220  
71032 Boeblingen, Germany*

E-mail:  
thomas.weber@de.ibm.com