IBM

IBM zSeries and System z9

(S12)

SSL Support mit Linux für zSeries und System z9

GSE Herbsttagung 2005 in Garmisch-Partenkirchen

24.-26.Oktober 2005

Dr. Manfred Gnirss
gnirss@de.ibm.com
TMCC Böblingen
IBM Deutschland Entwicklung GmbH

Arthur Winterling
winterling@de.ibm.com
zServer Software System Evaluation
IBM Deutschland Entwicklung GmbH

ON DEMAND BUSINESS™

© 2005 IBM Corporation

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

AIX*
CICS*
DB2*
e-business logo*
Enterprise Storage Server*
ESCON*
FICON
FlashCopy*
GDPS*
HyperSwap
IBM*
IBM eServer*
IBM logo*
NetView*
OS/390*
Parallel Sysplex*
S/390*
Sysplex Timer*
Tivoli*
TotalStorage*
z/OS*
z/VM*
zSeries*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Intel is a trademark of the Intel Corporation in the United States and other countries.
Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
UNIX is a registered trademark of The Open Group in the United States and other countries.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

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.

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

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. 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.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.

S12 – SSL Support mit Linux für zSeries

Dr. Manfred Gnirss, Arthur Winterling, IBM Böblingen

Der Vortrag beschreibt die Vorgehensweise der Installation sowie des Customizing eines SSL-Server mit Linux für zSeries bzw. für System z9. Hierbei wird insbesondere auf die Verwendung von Crypto Prozessoren / SSL Hardware Verschlüsselung eingegangen (siehe auch Vortrag S13 und G12)

Agenda

- Introduction
- Secure Socket Layer
- zSeries and System z9 Crypto Hardware
- Linux for zSeries – Cryptographic APIs
- SSL Support for z/VM
- Example: Apache with HW Crypto
- Check HW Crypto
- Misc.
- Summay

Introduction

Encryption of Data –
A Business Imperative

- Businesses are proactively focusing on securing customer and business data
  - Increasing regulatory requirements driving need for security of data for audit and compliance
  - Recent events highlight impact of loss/theft of removable data
  - Requirements for tighter security driving need for encryption of data

The Power of Mainframe Encryption

Helping to reduce risk across your value-net

Helping to protect data over the Internet

Helping to protect data leaving your enterprise

Customer objectives:

- Only intended party is allowed to decrypt
- Availability of the keys and decryption services when you need them

Protect archived data

Secure Socket Layer

SSL Linux

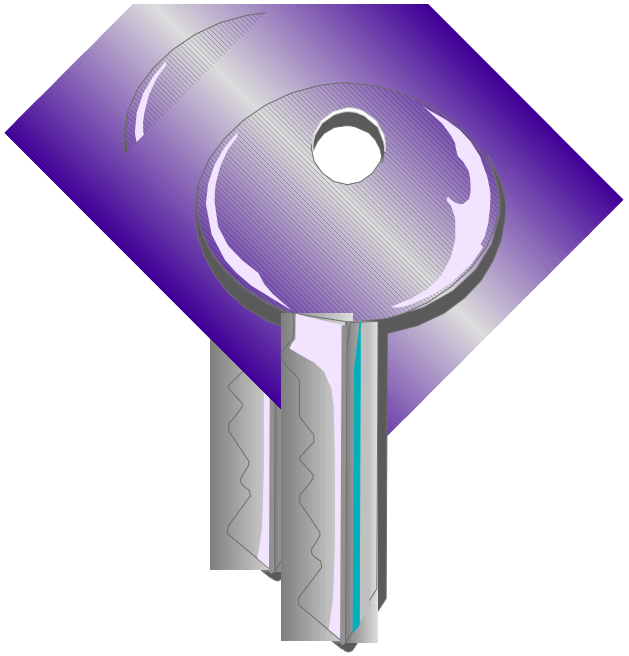
- Symmetric encryption (same key for encrypt and decrypt)
  - Problem is key exchange!
  - Relatively fast algorithms
- Asymmetic encryption (key-pair, one key for encrypt and one for decrypt – Public-Private Key)
  - No exchange of secret key via unsecure methods necessary!
  - Relatively slow algorithms (expensive, high CPU load)

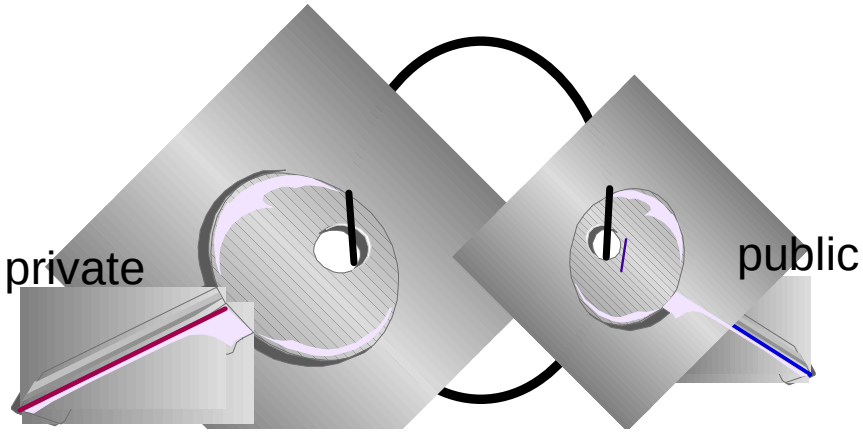
SSL Linux . . .

- Crypto in Software: Performance depends on CPU capacity.
- Crypto Support with specialized hardware:
  - Benefit better performance/throughput
  - Faster specialized HW and/or Off-loading from CPU
  - CPACF: DES, TDES, SHA-1, SHA-256, AES
  - PCI-Cryptofeatures, PCICC, PCICA, PCIXCC, CEX2x: RSA (and other ...)

Cryptographic Techniques for SSL

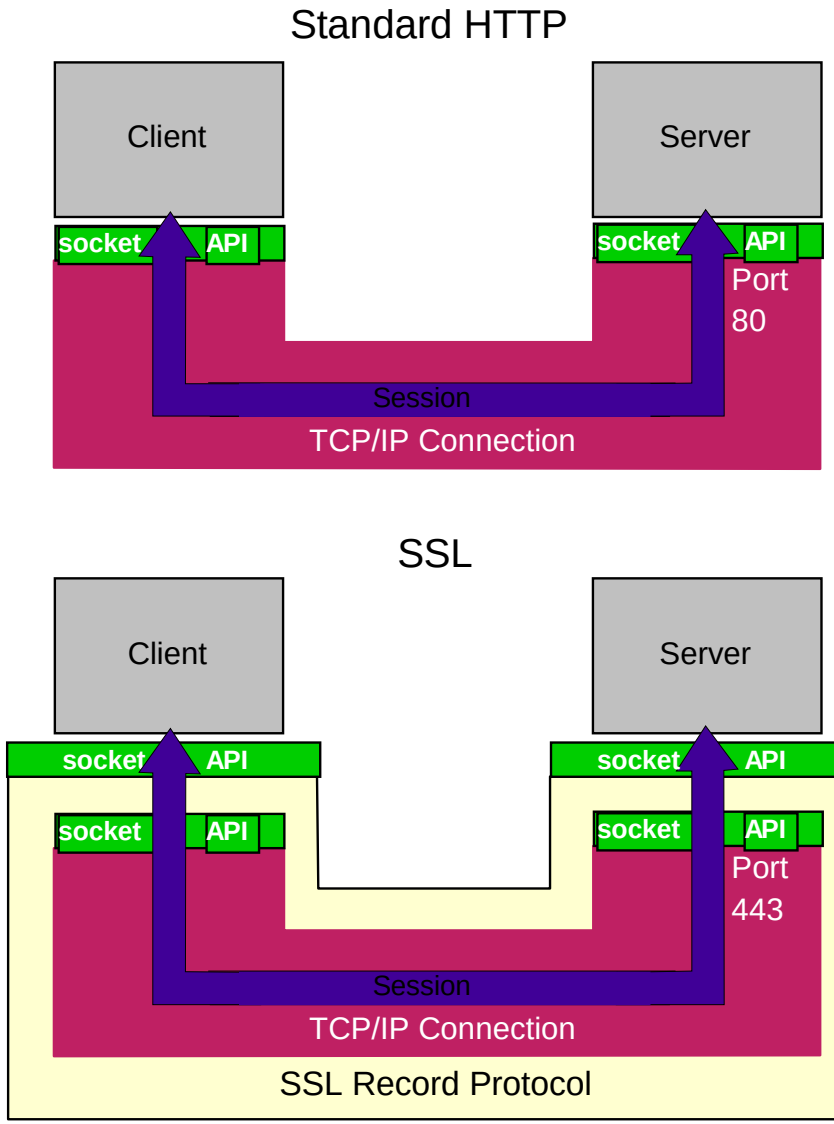
- **Symmetric Encryption**
  - Used to encrypt the data
  - DES, 3DES (Triple DES)
  - RC2, RC4
- **Asymmetric Encryption**
  - Used for key exchange and digital signatures
  - RSA
- **Message Digest/ Hashing**
  - For message integrity (message authentication)
  - MD5, SHA-1





The Secure Socket Layer Protocol

- **Creates secure channel**
  - **Encryption, Integrity, Authentication**
  - **Entire session is encrypted**
- **SSL request indicated by URL with https://**
  - **Triggers SSL handshake**
  - **SSL over HTTP uses different port number**
  - **Default port number: 443**
- **Secure channel can be used for other protocols**
  - **TN3270E server, LDAP, Java and FTP support SSL**

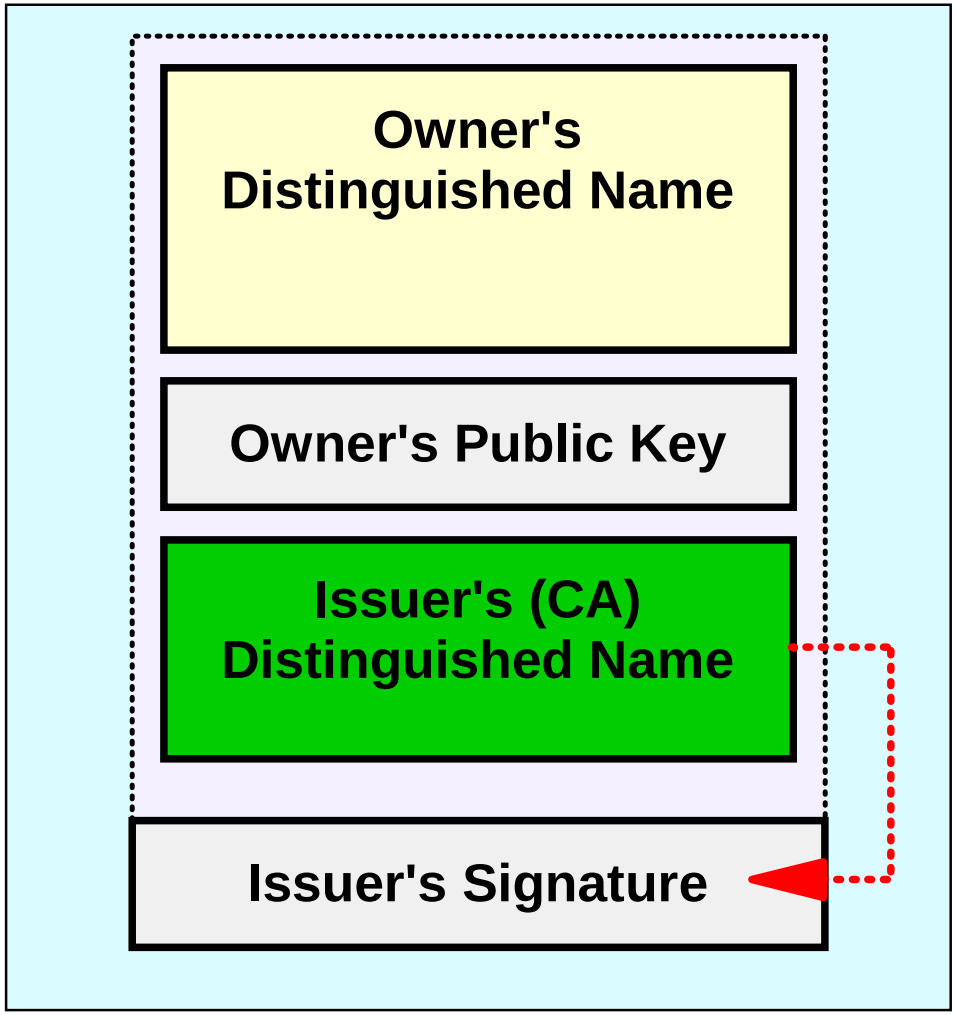


Cryptographic Techniques for SSL

- **Session concept: all packets have sequence numbers with Message Authentication**
  - **MD5 or SHA-1 used for MAC, algorithm similar to HMAC**
  - **TLS uses HMAC-MD5 or HMAC-SHA**
- **All data encrypted with symmetric cipher**
  - **DES, TDES, RC4 and RC2**
  - **Symmetric cipher used for performance reasons**
  - **1000 times faster than RSA for same no. of bytes**
- **Key exchange for encryption and MAC keys with asymmetric cipher**
  - **SSL uses RSA only**
  - **TLS additionally supports Diffie-Hellman**

Digital Certificates

- Certificate identifies its owner
- Main purpose is to publish the owner's public key
- Issuer is a Certification Authority (CA)
- Issuer's digital signature certifies the owner's identity and public key
  - Allows anyone who has CA certificate to verify the validity of the certificate



A Self-Signed Certificate

A Self-Signed Certificate . . .

- **Certificates are called "self-signed" if the certificate hierarchy does not end in a trusted root CA certificate**
  - **Trusted root certificates are delivered with the web browser or imported from a file (Microsoft IE)**
- **With self-signed certificates, it is impossible to assure the authenticity of the SSL session partner**
  - **Anybody can create a certificate with arbitrary content and sign it with their own CA**
- **Self-signed certificates are usually appropriate for intranet applications**
- **Web servers on the Internet should not use self-signed certificates**

SSL Cipher Suites

SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSL_RSA_WITH_DES_CBC_SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
SSL_RSA_WITH_NULL_MD5
SSL_RSA_WITH_NULL_SHA
SSL_RSA_WITH_IDEA_CBC_SHA
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA

Notes:

red - "exportable" cipher suites

green italicized - not supported by IBM web servers and most browsers

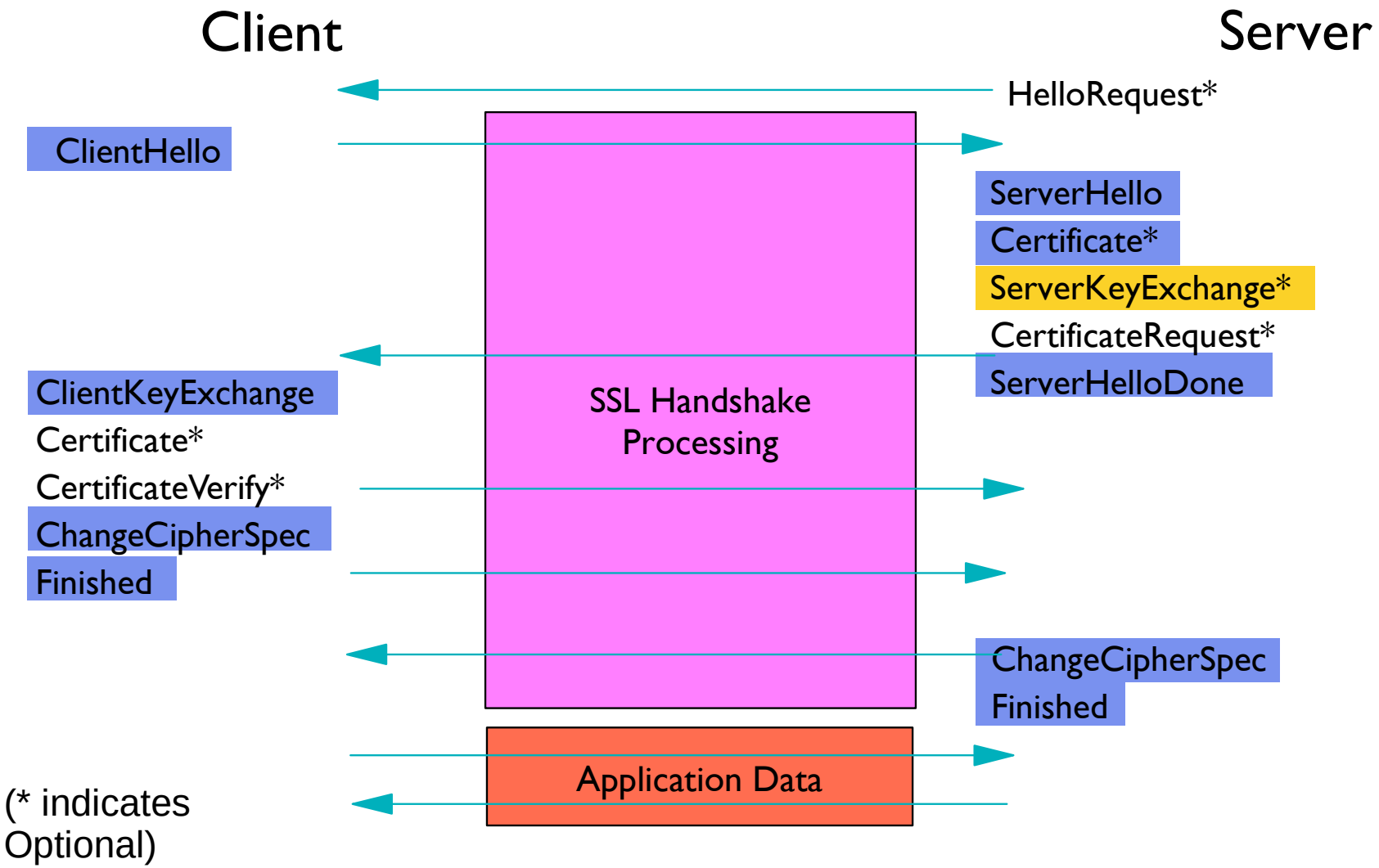
Most web browsers support only a subset of these cipher suites. The cipher suites shown are for SSL V.3

Netscape and Microsoft created proprietary cipher suites not in SSL spec.

SSL Cipher Suites

SSL_RSA_WITH_3DES_EDE_CBC_SHA	Triple-DES SHA (168-bit)
SSL_RSA_EXPORT_WITH_RC4_40_MD5	RC4 SHA (40-bit)
SSL_RSA_WITH_RC4_128_MD5	RC4 MD5 (128-bit)
SSL_RSA_WITH_DES_CBC_SHA	DES SHA (56-bit)
SSL_RSA_WITH_RC4_128_SHA	RC4 SHA (128-bit)
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	RC2 MD5 (40-bit)
SSL_RSA_WITH_NULL_SHA	
SSL_RSA_WITH_NULL_MD5	
SSL_NULL_WITH_NULL_NULL	
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA	RC4 SHA Export 1024 (56-bit)
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA	DES SHA Export 1024 (56-bit)

SSL Version 3 Handshake



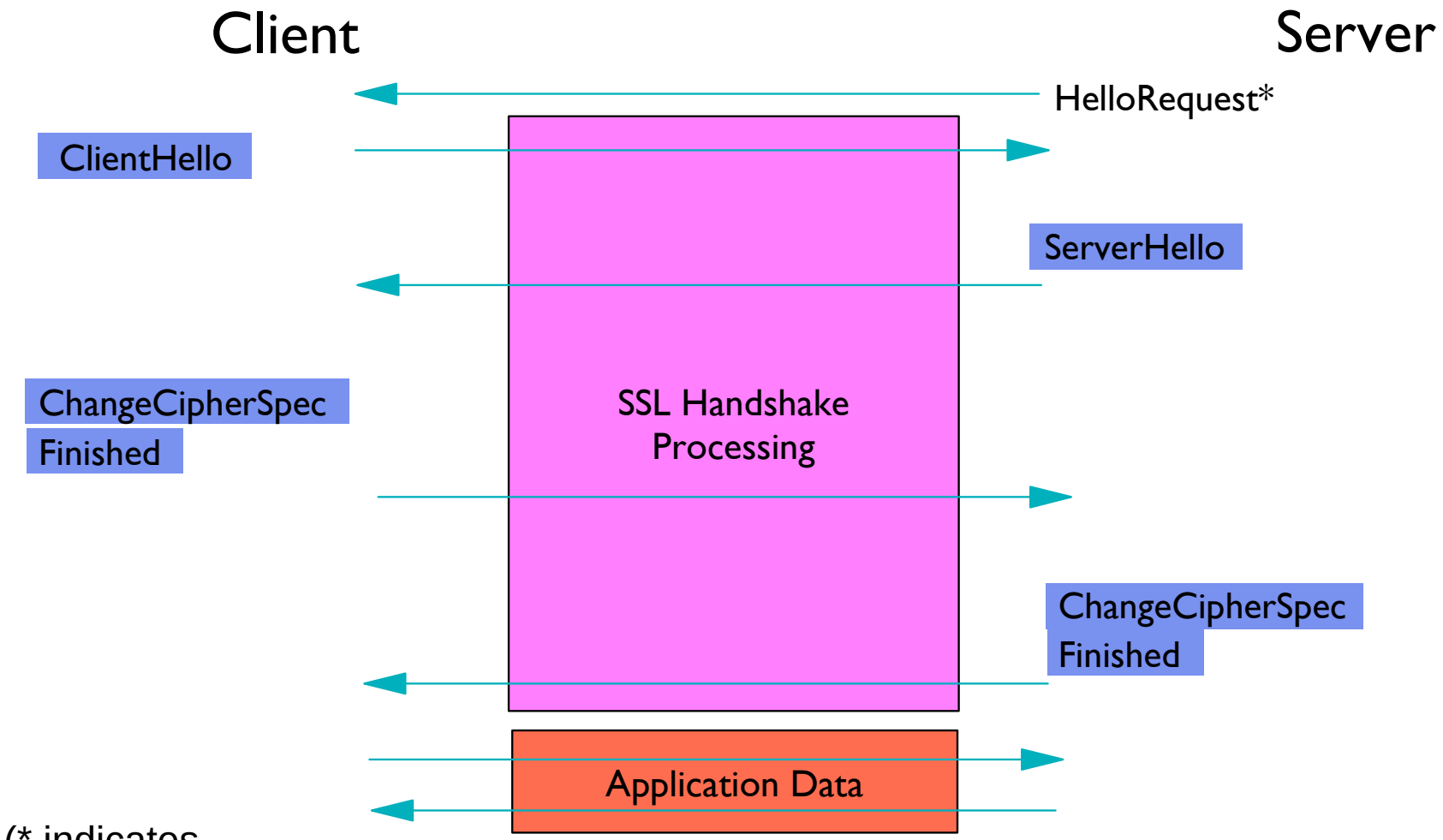
SSL Version 3 Handshake (no Client Authentication)

- **"Client hello" message**
  - **Browser sends list of supported cipher suites in preference order**
  - **Nonce (Number used once, random number)**
- **"Server hello" message**
  - **Cipher suite selected by server**
  - **Session ID (16 byte value) and nonce**
- **"Server hello done" message**
  - **Indicates end of server generated messages**
- **"Client key exchange" message**
  - **Client creates random "Pre Master Secret" (48 bytes), encrypts with server's public key**

SSL Version 3 Handshake (no Client Authentication) . . .

- **Client and Server generate keys**
  - **MAC secrets, write keys, IVs for client and server**
  - **From pre-master secret, client nonce, and server nonce**
- **"Change cipher spec" messages**
  - **Sent by both client and server to switch to new cipher suite**
- **"Finished" messages**
  - **MAC of all previous messages in handshake**
  - **Sent by both client and server**
  - **First message encrypted and MAC'd with the new cipher keys**
  - **Provide message integrity for the entire handshake**

Resuming an SSL Session



(* indicates Optional)

Resuming an SSL Session . . .

- **"Client hello" message**
  - **Browser includes ID of previous session in message**
- **"Server hello" message**
  - **Returns the same session ID to indicate that session will be resumed**
  - **Server caches session parameters until timeout reached**
- **No new encryption parameters for resumed session**
  - **Saves costly RSA decryption of "Pre-master secret"**
  - **New session keys are generated**
    - **Old Pre-master-secret, new client nonce and server nonce**
    - **No "Perfect Forward Secrecy": attacker who compromises Pre-master secret can generate keys of all resumed sessions**

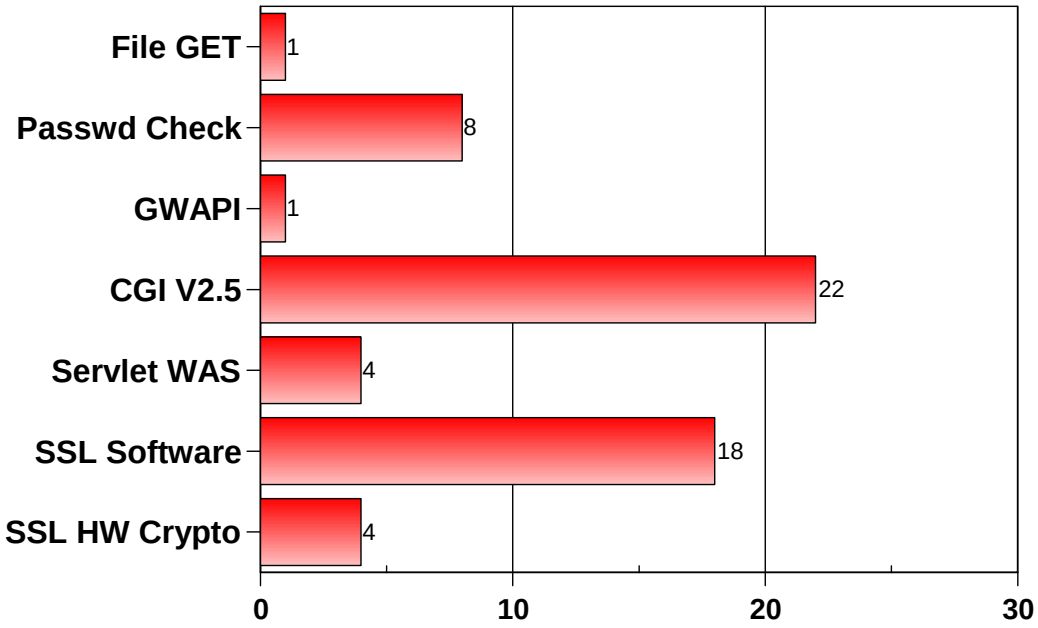
Resuming an SSL Session . . .

- **"Change cipher spec" messages**
  - **Sent by both client and server to switch to new cipher suite**
- **"Finished" messages**
  - **MAC of all previous messages in session resume**
  - **Sent by both client and server**
  - **First message encrypted and MACed with the new cipher keys**
  - **Provide message integrity for the entire resume**

Use of Cryptographic Hardware

- **RSA decryption is computationally very intensive (~70% of CPU usage for handshake)**
- **Use of crypto hardware can increase max. no. of handshakes per second dramatically**
  - **Up to 2000/sec. on z/900**
- **Crypto hardware can improve data encryption performance**
  - **If 3DES is used**
  - **RC4 always in software**

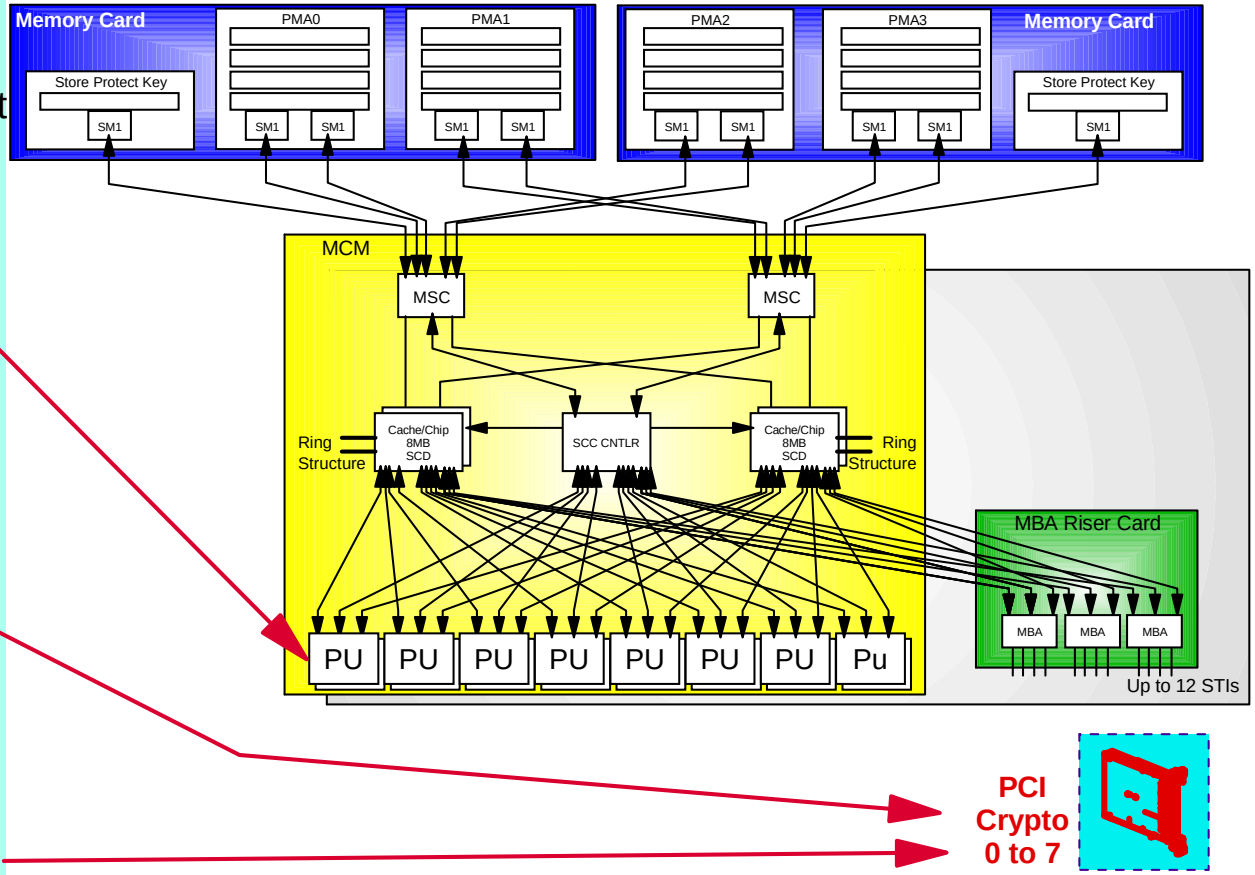
Rough Idea of Relative CPU Costs for some Web Options



zSeries and System z9 Crypto Hardware

z890, z990 Hardware Cryptographic Coprocessors

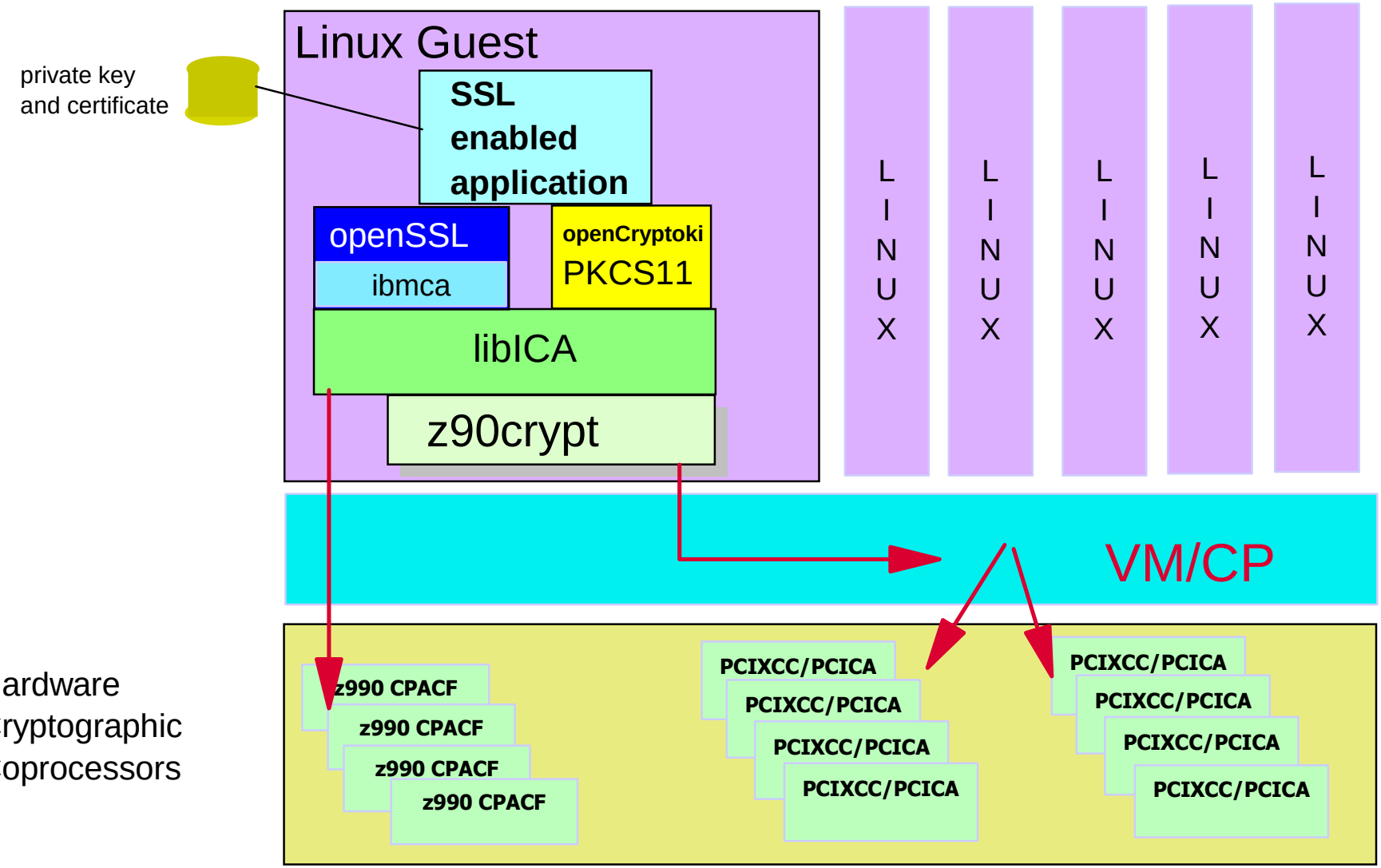
- **CP Assist for Cryptographic Functions (CPACF)**
  - One CPACF per processing unit
  - standard orderable feature
  - 5 new published crypto instructions or through ICSF
  - non-secure (clear keys only)
- **PCI Cryptographic Accelerator (PCICA)**
  - priced feature
  - High performance SSL assist
  - 0 to 6 features in a system
- **PCIX Cryptographic Coprocessor (PCIXCC)**
  - priced feature
  - hardware tamperproof
  - 0 to 4 features in a system
- **Crypto Express2**



- CPACF and PCICA exploitable by Linux and z/VSE
- up to 8 features total (PCIXCC and PCICA mix)

© 2003 IBM Corporation

Linux access to cryptographic hardware support



Hardware
Cryptographic
Coprocessors

Crypto HW- Performance

www.ibm.com/servers/eserver/zseries/security/cryptography.html

z990 Figures:

One CPACF: 400+MB/sec DES, 160+MB/secT-DES, 350+MB/sec SHA-1

Crypto Express 2 (Coprocessor Mode – CEX2C)

DES – 4KB blocks = 5.2 MB/sec for one CEX2C feature
T-DES – 4KB blocks = 4.8 MB/sec
MAC – 4KB blocks = 4.8 MB/sec
DSG (CRT – 1024-bit) = 2200/sec

SSL handshakes
PKD-CRT 1024-bit = 2100/sec

System z9 Figures:

Crypto Express 2 (Accelerator Mode – CEX2A)

SSL handshakes
PKD-CRT 1024-bit = 6000/sec for one feature with two CEX2A

Crypto Performance with Linux for zSeries

For all Linux Open SSL measurements the following applies:
•Linux Kernel Level: 2.4.19
•Open SSL Code Level: 0.9.6E
•z90Crypt Level: 1.1.2
•No Client Authentication

Linux native in LPAR

Caching SID	Handshake	# of CPs	ETR	Utilization %
no	Software	4	208	99.9
no	8 Cryp.Acc.Cards	4	6,703	99.5
no	12 Cryp.Acc.Cards	16	13,068	55.1

For all Linux Open SSL measurements the following applies:
- Linux running native on 2084-304
- Linux System Level: SLES8 SP4
- Linux Kernel Level: 2.4.21-266
- Open SSL Code Level: 0.9.7a
- z990Crypt Level: 1.3.2
- No Client Authentication

Caching SID	Handshake	Cipher	ETR	CPU Util. %
no	Software	RC4,MD5	202	99.99
no	4 PCIXCC Feat.	RC4,MD5	4,630	83.55
no	2 CEX2C Feat.	RC4,MD5	4,320	80.06
no	4 CEX2C Feat.	RC4,MD5	5,697	99.89

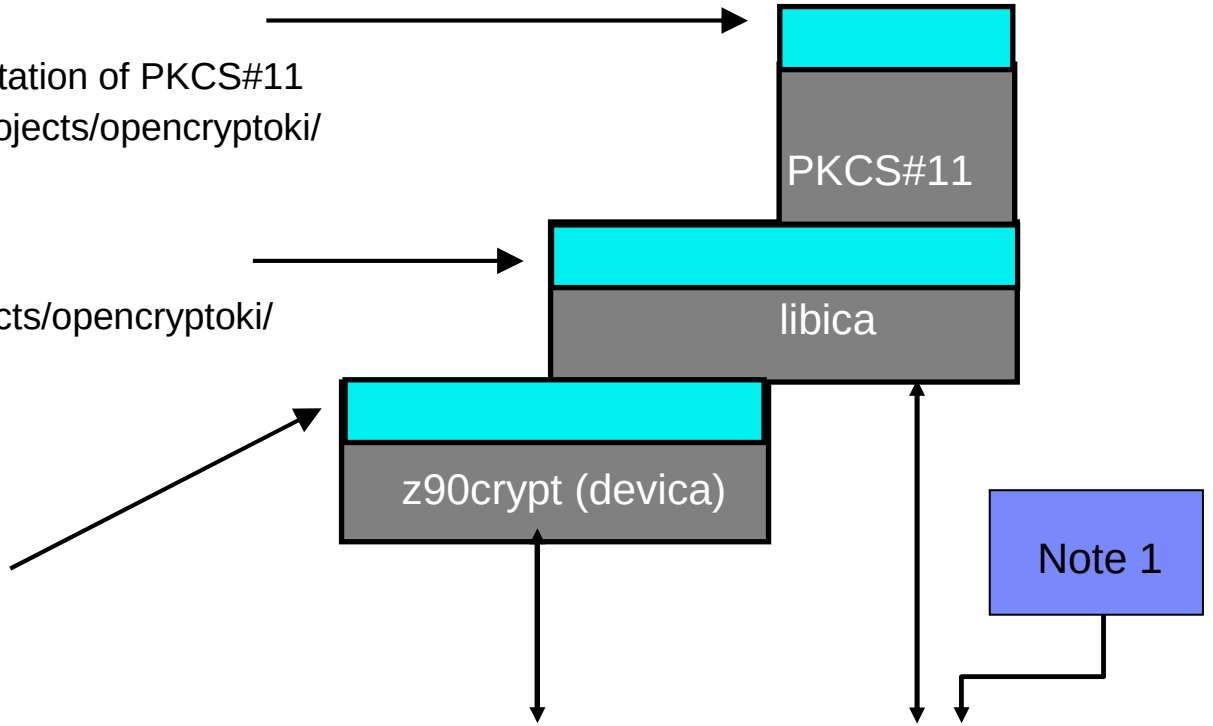
z990 Crypto performance figures at www.ibm.com/servers/eserver/zseries/security/cryptography.html

Linux For zSeries Cryptographic APIs

Crypto APIs With Linux For zSeries and System z9

OpenCryptoki is openSource implematation of PKCS#11
Information: http://sourceforge.net/projects/opencryptoki/

Information: http://sourceforge.net/projects/opencryptoki/



z90crypt API
- hdw status check
- random number generation
- RSA decryption (modular/CRT exponentiation)
- quiescing

PCI hardware crypto
(PCIXCC, PCICA, Crypto Express2)

CPACF
(z990, z890, System z9)

Known exploiters
- Libica: openSSL, …
- PKCS#11: IBM WebSphere Application Server, Tivoli Access manager, IBM JDK 1.4, …

Note 1: Kernel 2.6 invokes directly the CPACF for IPSec VPN, …

The z90crypt API

- Required structures provided in z90crypt.h
- All interactions require to get a device handle first
  - dh=open("/dev/z90crypt",0_RDWR)

- Functions
  - Hardware status check

    rc=ioctl(dh,**ICAZ90STATUS**,my_z90crypt_status);

    provides a mask of installed cards and type (PCICC, PCICA or Crypto Express2)

  - Pseudo Random Number generation

    rc =**read** (dh,my_buffer,number_of_bytes)

    get a string of pseudo random bytes (software generated)

  - Quiescing

    rc =ioctl(dh,**ICAZ90QUIESCE**)

    the driver completes the outstanding work, refuses new requests

The z90crypt API . . .

- Functions (cont'd)
  - RSA exponentiation (calls PCICA, PCIXCC or CryptoExpress2)
    - The RSA clear key can be provided in two formats
      - Modulus-exponent
      - CRT (Chinese Remainder Theorem) format (faster processing)
    - Processed respectively by
      - rc=ioctl(dh,**ICARSAMODEXPO**,<structure_name>) or
      - rc=ioctl(dh,**ICARSACRT**,<structure_name>)
- See key structures in appendix
- RSA key pairs expected to be initially generated using a utility program such as OpenSSL
- z90crypt API test program testcrtde.c provided in redbook SC24-6870

The libica API

IBM eServer Cryptographic Accelerator Device Interface Library (libica) for the Linux Operating System

"IBM is contributing to the open source community the Linux operating system implementation of the
ICA interface library. This library is used to provide a generalized abstraction to the ICA device driver interface. It is the fundamental building block of all the function that IBM will make available
to interface to the device.
This API is also common across the AIX, and Windows platform device support for the device, allowing for portablility of applications. "

- icaGetAdapterMask
- icaGet AdapterID
- icaGetVPD
- icaOpenAdapter
- icaCloseAdapter
- icaRsaModExpo *
- icaRsaCrt *
- icaRsaKeyGenerateModExpo *
- icaRsaKeyReGenerateCrt *
- icaDesEncrypt **
- icaDesDecrypt **
- icaTDesEncrypt **
- icaTDesDecrypt **
- icaDesMac
- icaTDesMac
- icaSha1**
- icaRandomNumberGenerate

* PCICA , PCIXCC or Crypto Express2

* * CPACF

Driving the coprocessors with Linux for zSeries

IBM PKCS#11 API
Project

Open Source implementation of the PKCS#11 API (aka Cryptoki). providing support for the IBM eServer Cryptographic Accelerator and the IBM 4758 Cryptographic Coprocessor

PKCS #11 - Cryptographic Token Interface Standard

This standard specifies an API, called Cryptoki, to devices which hold cryptographic information and perform cryptographic functions. Cryptoki, pronounced crypto-key and short for cryptographic token interface, follows a simple object-based approach, addressing the goals of technology independence (any kind of device) and resource sharing (multiple applications accessing multiple devices), presenting to applications a common, logical view of the device called a cryptographic token. "

The specifications of PKCS#11 are at
http://rsasecurity.com/rsalabs/pkcs

SSL Support for z/VM

Secure Socket Layer (SSL) Support in z/VM

SSL support between a remote client and z/VM TCP/IP server is provided via a SSL server.

The SSL server is a special Linux machine only for this purpose.

The SSL server manages the certificate database and handles encryption and decryption of data

3 Steps:
- Install and configure SSL server (Linux)
- Configure TCP/IP
- Test configuration

Secure Socket Layer (SSL) Support in z/VM . . .

TCP/IP for VM Secure Socket Layer (SSL) Server Configuration Information and Requirements

http://www.vm.ibm.com/related/tcpip/vmsslinf.html

For z/VM 4.4 to z/VM – z/VM 5.1:

The z/VM SSL server implementation is supported on **specific Linux distributions**, for which **specific Linux IUCV driver support** is also required.
Supported distributions, requisite IUCV patches, and the z/VM-supplied SSL RPM packages for each distribution are listed in the table that follows.

SUSE Kernel Version	Required Patches	Linux RPM Package File	z/VM-Supplied RPM File
2.4.19 (**SLES-8**) †	Not Applicable	vmssld-1.24.19-1.rpm	VMSLDSB RPMBIN

(†) 31-bit version **only**

File transfer and installation instructions: see RPM Fackage File.

Check also for detailed information about service updates for the z/VM SSL Server (APARs/PTFs)

Secure Socket Layer (SSL) Support in z/VM . . .

Configure sslserv virtual machine

z/VM 5.1 TCP/IP Planning and cutomization - SC24-6125
Chapter 23: Configuring the SSL Server

SSL Server Configuration Steps

1. Install the appropriate VMSSL Linux Red Hat Package Manager (RPM) package
2. Update the PROFILE TCPIP file
3. Update the DTCPARMS file for the SSL server
4. Update the ETC SERVICES file

Set up the certificate database

Secure Socket Layer (SSL) Support in z/VM . . .

TCP/IP SSL Server – Configuration
Certificate Testing Examples, Hints and Tips

http://www.vm.ibm.com/related/tcpip/tcsslcfx.html

- SSL capable client, like
  - IBM Personal Communications (commonly referred to as "PCOM") Telnet client
  - BlueZone Telnet client
  - BlueZone FTP client
  - Netscape browser (HTTP and FTP protocols)

- Certificat (useful command: SSLADMIN)
  - Self-signed certificate
  - Certificate signed by a CA (free test certificates available)

- Send and store also certificate to client

Example: Apache With HW Crypto

Set up of Apache with Hardware Crypography

Prepare Hardware and z/VM for HW-Crypto Support

Hardware crypto enablement

- z990/z890 Crypto (CPACF) requires feature code 3863 to be enabled in the system LIC (PoR). Install PCI crpto cards.

Virtualization considerations

- The logical partition requires PCICA/PCIXCC/Crypto Express2 to be specified in the PCI Cryptographic candidate and online lists in the LPAR image profile, and a domain index to be selected

- The **VM USER DIRECTORY** for the Linux guest machine has the APVIRT (shared AP queues) operand for the CRYPTO statement. Example: **CRYPTO APVIRT**
- A linux guest directory can also have the APDED (dedicated AP Q) keyword instead of APVIRT in the CRYPTO statement

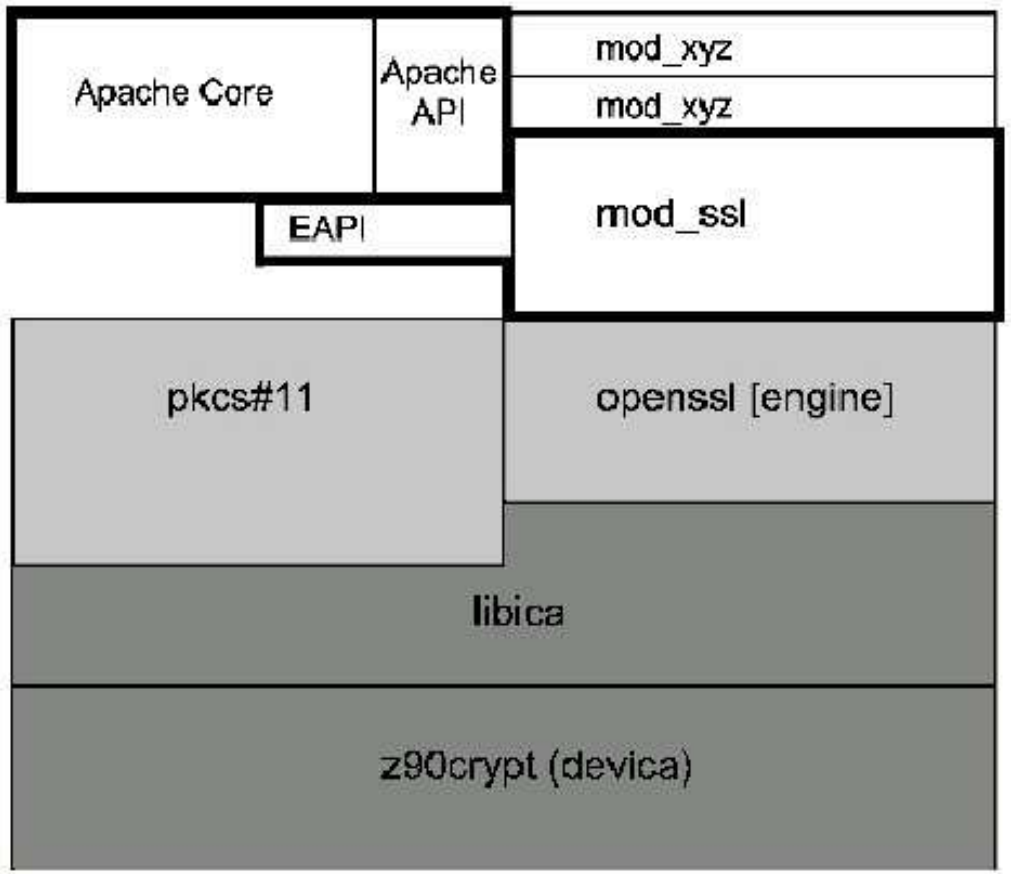
Notes:
Documentation: zSeries Crypto Guide Update (SG24-687)
Crypto domain concept is irrelevant for Linux on zSeries (as of Clear key only), but Domain number has to be specified!
If mixture of accelerator and coprocessor cards, z90crypt uses accelerator cards only

Set up of Apache with Hardware Crypography . . .

Classical way

Download all necessary parts from the web

Check:
http://sourceforge.net/projects/opencryptoki
http://www.apache.com
http://oss.software.ibm.com/developerworks/

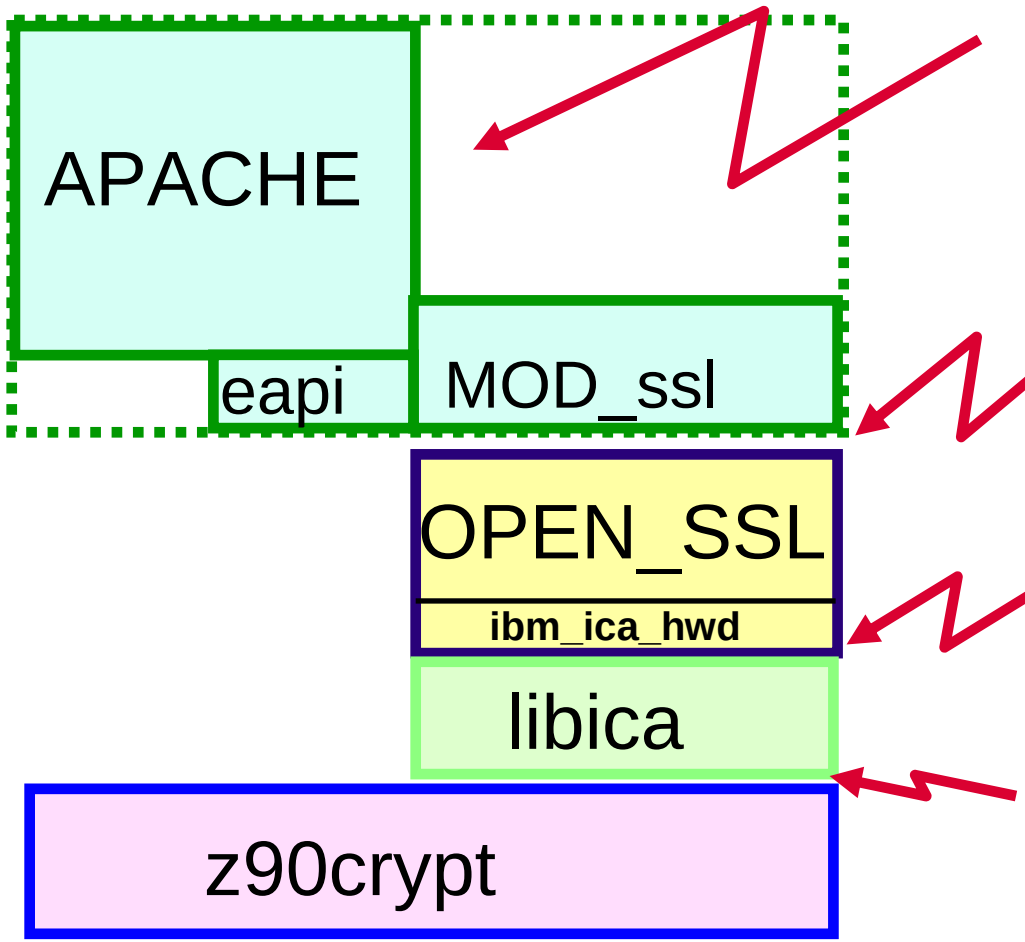


Set up of Apache with Hardware Crypography . . .

Classical way

Verify and test your installation

Check for guideance: zSeries Crypto Update Redbook



Test 4

- modify httpd.conf and restart
- use https request
- verify in httpd logs
- check device status

Test 3

- try "speed" from openssl command prompt with -engine option

Test 2

- run low-level ica test: icacrtde*

*materials from redbook

Test 1

- check device status after loading
- run low-level driver test tstcrtde*

Set up of Apache with Hardware Crypography . . .

Classical way

1. Install the crypto device driver z90crypt
2. Load z90crypt and define the crypto device node
   - z90crypt queries the status of the installed PCI coprocessors at startup time - can be retrieved by displaying from pseudo directory /proc/driver/z90crypt
3. Low level testing

   C tool programs: tstcrtde and icacrtde - Program source can be found in redbook SG24-6870
4. Installation of the libica crypto interface library
   - API for the IBM eServer Cryptographic Accelerator Device Driver
   - intended to be used by middleware such as openSSL
5. Build openSSL with the ibmca engine
   - test at low level using the provided icatest program
   - openSSL can be used to test execution speed w/ and w/o ibmca
6. Install the Apache web server
   - the mod_ssl module is configured to be build into Apache with crypto
   - the SSLCryptoDevice directive in httpd.conf defines the ibmca crypto device
7. Create Cetrificate
8. Test installation

Set up of Apache with Hardware Crypography . . .

„Prepared“ Linux Distribution (Example SuSE SLES9)

Modern Linux Distributions (for example SuSE SLES 9) contain already all required components

- Load **z90crypt** driver
  - In */etc/rc.config*: *START_Z90CRYPT=yes* (loading during start)
  - *rcz90crypt start* (loading at any time)
    Note: This script is available after libica is installed
- Install (if not already done) **libica** via Yast or rpm
- Install (if not already done) **openCryptoki** via Yast or rpm
- Install (if not already done) **OpenSSL** via Yast or rpm
  - Check if OpenSSL is patched with correct engine path

    This can be done simply with:
    run: *openssl speed rsa1024 –engine ibmca –elapsed*
    message „*can‘t use that engine*“ indicates a problem . . .

Set up of Apache with Hardware Crypography . . .

„Prepared“ Linux Distribution (Example SuSE SLES9)

- Install (if not already done) **apache V2** via Yast or rpm
- You will need a *index.html* page later
  (example is in apache2-example-pages package)
- (run: *SuSEconfig –module apache2*)
- Edit */etc/sysconfig/apache2*
  change: *APACHE_START_TIMEOUT=2* to *20*
  add to *APACHE_SERVER_FLAGS*: *SSL*
  add: *ServerName* and *ServerAdmin*
- Check */etc/sysconfig/apache2*
  contains *APACHE_MODULES ssl*
- Check in */etc/apache2/listen.conf*
  check if port 443 is defined
- **Edit */etc/apache2/ssl-global.conf*: Add the Crypto Device Directive**
  add on section *<IfModule mod_ssl.c>*: *SSLCryptoDevice ibmca*

Set up of Apache with Hardware Crypography . . .

„Prepared“ Linux Distribution (Example SuSE SLES9)

- Create a virtual host file
  - *cd /etc/apache2*
  - cp vhosts.d/vhost-ssl.template vhosts.d/vhosts-ssl.conf
  - edit *vhosts-ssl.conf*:

    add ServerName and ServerAdmin
    uncomment *SSLCertificateChainFile /etc/apache2/ssl.crt/ca.crt*
  - edit *httpd.conf*:

    add *vhosts-ssl.conf* to the include statement in section Virtual server configuration:
    Include /etc/apache2/vhosts.d/vhost-ssl.conf
- run: *SuSEconfig –module apache2*
- Create index page in */srv/www/htdocs*
- Check apache configuration
  - *acapache2 configtest*
  - *httpd2 -S*

Set up of Apache with Hardware Crypography . . .

„Prepared“ Linux Distribution (Example SuSE SLES9)

- Create a certificate

  Two methods available:

  - gensslcert
  - certificate.sh (Snake Oil)
- Check Logs in case of problems in */var/log/apache2*
  - *access_log*
  - *error_log*
  - *ssl_request_log* (exists only if SSL is configured correctly)
- Start Apache 2 with SSL:
  - Run: *rcapache2 start*
    - „*start*“ is sufficient, as *APACHE_SERVER_FLAGS* is set to *SSL*
    - Provide Pass Phrase, if prompted

Verify Installation for HW Crypto Support

Load Linux z90crypt device driver

Load the Linux z90crypt device driver with rcz90crypt if not already done.

```
t29lp40:~/crypto/tools # rcz90crypt start
Loading z90crypt module
```

You can check whether z90crypt is loaded using dmesg

Example for Crypto hardware support available:

```
z90crypt: Version 1.3.3 loaded, built on Oct 11 2005 16:46:19
z90crypt: z90main.o ($Revision: 1.31.2.9 $/$Revision: 1.8.2.5 $/$Revision: 1.2.2.4 $)
z90crypt: z90hardware.o ($Revision: 1.19.2.7 $/$Revision: 1.8.2.5 $/$Revision: 1.2.2.4 $)
```

Example for Crypto hardware support not available:

```
z90crypt: Version 1.3.2 loaded, built on Aug 26 2005 00:57:18
z90crypt: z90main.o ($Revision: 1.31.2.8 $/$Revision: 1.8.2.4 $/$Revision: 1.2.2.3 $)
z90crypt: z90hardware.o ($Revision: 1.19.2.6 $/$Revision: 1.8.2.4 $/$Revision: 1.2.2.3 $)
z90crypt: query_online -> Exception testing device 0
z90crypt: helper_scan_devices -> exception taken!
z90crypt: z90crypt_config_task -> Error 34 detected in refresh_z90crypt.
```

Check HW Crypto

Verify Installation for HW Crypto Support . . .

Query status of Linux z90crypt device driver

Example:Status **before** some crypto requesets have been performed

```
t291p40:~ # cat /proc/driver/z90crypt

z90crypt version: 1.3.3
Cryptographic domain: 8
Total device count: 12
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 8
CEX2A count: 4
requestq count: 0
pendingq count: 0
Total open handles: 0

Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A
           5566555556560000 0000000000000000 0000000000000000 0000000000000000

Waiting work element counts
           0000000000000000 0000000000000000 0000000000000000 0000000000000000

Per-device successfully completed request counts
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Verify Installation for HW Crypto Support . . .

Query status of Linux z90crypt device driver

Example:Status **after** some crypto requesets have been performed

```
t29lp40:~/crypto/tools # cat /proc/driver/z90crypt

z90crypt version: 1.3.3
Cryptographic domain: 8
Total device count: 12
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 8
CEX2A count: 4
requestq count: 0
pendingq count: 0
Total open handles: 0


Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A
            5566555556560000 0000000000000000 0000000000000000 0000000000000000


Waiting work element counts
            0000000000000000 0000000000000000 0000000000000000 0000000000000000


Per-device successfully completed request counts
    00000BF2 00000BDB 00000A7E 00000A4E 00000BDE 00000BD4 00000BBE 00000BB0
    00000896 00000A23 00000889 000009F2 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Verify Installation for HW Crypto Support . . .

Test OpenSSL with HW cryptography provided with engine ibmca

Example:OpenSSL with engine ibmca for RSA-1024

```
t29lp40:~/crypto/tools # openssl speed rsa1024 -engine ibmca
engine "ibmca" set.
Doing 1024 bit private rsa's for 10s: 1000 1024 bit private RSA's in 0.00s
Doing 1024 bit public rsa's for 10s: 1000 1024 bit public RSA's in 0.00s
OpenSSL 0.9.7d 17 Mar 2004
built on: Tue Oct 11 15:42:57 UTC 2005
options:bn(64,64) md2(int) rc4(ptr,int) des(idx,cisc,4,long) aes(partial) blowfish(idx)
compiler: gcc -fPIC -DOPENSSL_THREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -DOPENSSL_NO_KRB5 -DB_ENDIAN -DNO_ASM -DMD32_REG_T=int -
DOPENSSL_NO_RC5 -DOPENSSL_NO_IDEA -O2 -fsigned-char -fmessage-length=0 -Wall -fomit-frame-pointer -fno-strict-aliasing -DTERMIO -Wall
-fbranch-probabilities
available timing options: TIMES TIMEB HZ=100 [sysconf value]
timing function used: times
                  sign    verify    sign/s verify/s
rsa 1024 bits   0.0000s   0.0000s 1000000.0 1000000.0
t29lp40:~/crypto/tools #
```

Verify Installation for HW Crypto Support . . .

Test OpenSSL with HW cryptography provided with engine ibmca

Example:OpenSSL with engine ibmca for SHA

```
t29ip40:~/crypto/tools # openssl speed sha -engine ibmca
engine "ibmca" set.
Doing sha1 for 3s on 16 size blocks: 1074895 sha1's in 3.00s
Doing sha1 for 3s on 64 size blocks: 972592 sha1's in 3.00s
Doing sha1 for 3s on 256 size blocks: 878587 sha1's in 3.00s
Doing sha1 for 3s on 1024 size blocks: 643746 sha1's in 3.00s
Doing sha1 for 3s on 8192 size blocks: 185033 sha1's in 3.00s
OpenSSL 0.9.7d 17 Mar 2004
built on: Tue Oct 11 15:42:57 UTC 2005
options:bn(64,64) md2(int) rc4(ptr,int) des(idx,cisc,4,long) aes(partial) blowfish(idx)
compiler: gcc -fPIC -DOPENSSL_THREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -DOPENSSL_NO_KRB5 -DB_ENDIAN -DNO_ASM -DMD32_REG_T=int -
DOPENSSL_NO_RC5 -DOPENSSL_NO_IDEA -O2 -fsigned-char -fmessage-length=0 -Wall -fomit-frame-pointer -fno-strict-aliasing -DTERMIO -Wall
-fbranch-probabilities
available timing options: TIMES TIMEB HZ=100 (sysconf value)
timing function used: times
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
sha1              5732.77k    20748.63k    74972.76k   219731.97k   505263.45k
```

Verify Installation for HW Crypto Support . . .

Test OpenSSL with HW cryptography provided with engine ibmca

Example:OpenSSL with engine ibmca for DES

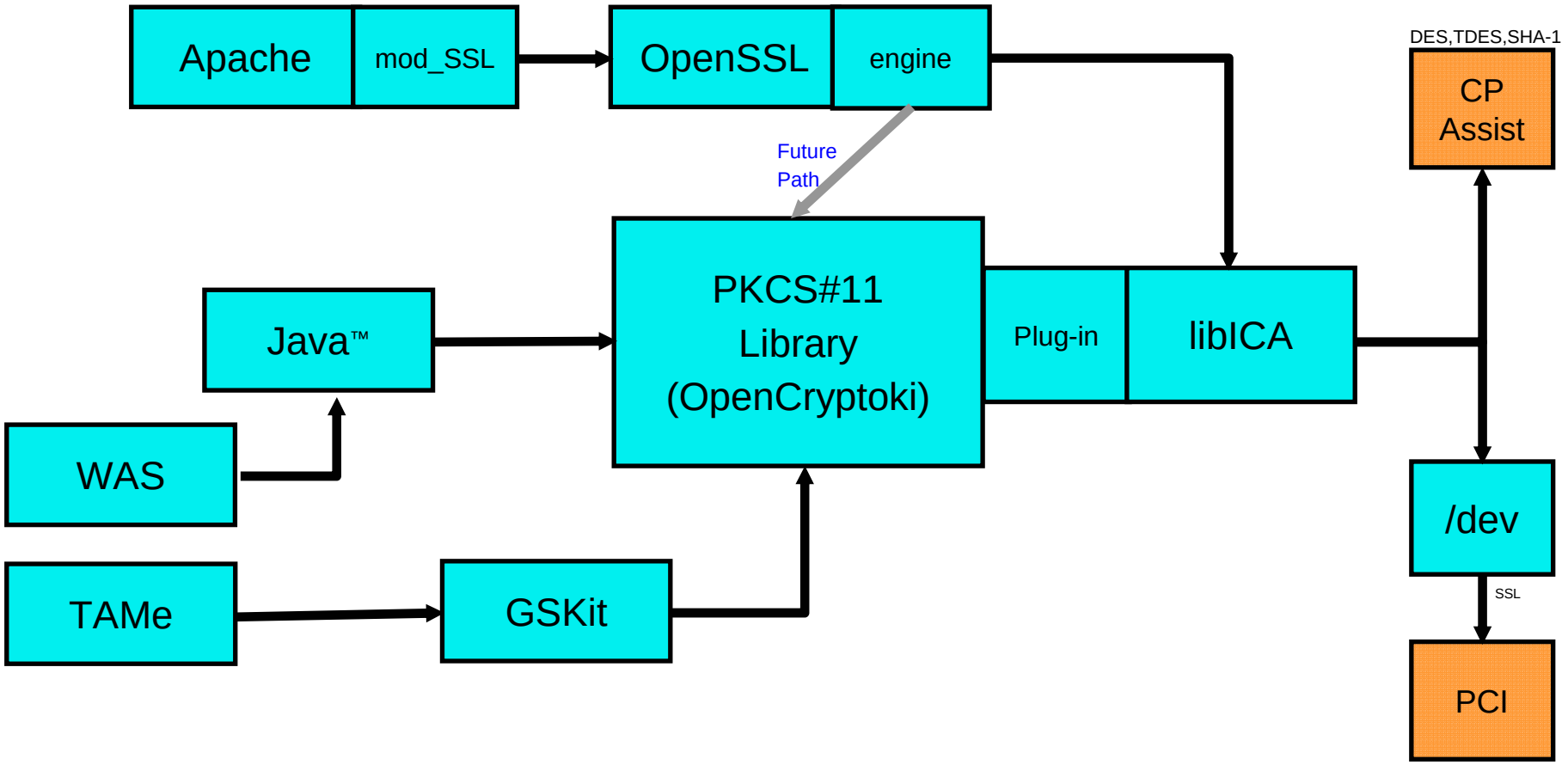
```
t29ip40:~/crypto/tools # openssl speed des -engine ibmca
engine "ibmca" set.
Doing des cbc for 3s on 16 size blocks: 3548696 des cbc's in 3.00s
Doing des cbc for 3s on 64 size blocks: 967139 des cbc's in 3.00s
Doing des cbc for 3s on 256 size blocks: 246998 des cbc's in 3.00s
Doing des cbc for 3s on 1024 size blocks: 61997 des cbc's in 3.00s
Doing des cbc for 3s on 8192 size blocks: 7785 des cbc's in 3.00s
Doing des ede3 for 3s on 16 size blocks: 1359777 des ede3's in 3.00s
Doing des ede3 for 3s on 64 size blocks: 350043 des ede3's in 3.00s
Doing des ede3 for 3s on 256 size blocks: 88161 des ede3's in 3.00s
Doing des ede3 for 3s on 1024 size blocks: 22071 des ede3's in 3.00s
Doing des ede3 for 3s on 8192 size blocks: 2755 des ede3's in 3.00s
OpenSSL 0.9.7d 17 Mar 2004
built on: Tue Oct 11 15:42:57 UTC 2005
options:bn(64,64) md2(int) rc4(ptr,int) des(idx,cisc,4,long) aes(partial) blowfish(idx)
compiler: gcc -fPIC -DOPENSSL_THREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -DOPENSSL_NO_KRB5 -DB_ENDIAN -DNO_ASM -DMD32_REG_T=int -
DOPENSSL_NO_RC5 -DOPENSSL_NO_IDEA -O2 -fsigned-char -fmessage-length=0 -Wall -fomit-frame-pointer -fno-strict-aliasing -DTERMIO -Wall
-fbranch-probabilities
available timing options: TIMES TIMEB HZ=100 (sysconf value)
timing function used: times
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
des cbc          18926.38k    20632.30k    21077.16k    21161.64k    21258.24k
des ede3          7252.14k     7467.58k     7523.07k     7533.57k     7522.99k
```

Misc.

Linux for zSeries and System z9 - Software libraries with RSA encryption

Software	Remark	Hardware Support
mod_ssl	Uses OpenSSL	Yes (if engine ibmca is used)
OpenSSL	Uses libica	Yes (if engine ibmca is used)
Gskit Gskit V7 together with SusE SLES8 Gskit V7together with Red Hat Linux 3.0 AE	Uses pkcs#11, libica PCICA, CEX2A is supported	Yes
openCryptoki	Open source implementation of PKCS#11	Yes

Hardware Crypto Exploitation (Summary)



Key:

Hardware

Software

Summary

Recommendation

- For data transport and data exchange, please consider seriously usage of cryptographic methods!
- If you have any chance to access crypto hardware support, benefit from performance and throughput increase.
- If you have already some crypto hardware installed, enable it also for usage with Linux for zSeries and System z9
  (If you are using a z890, z990, or System z9 then you can enable the CPACF in any case.)

Questions ?

Linux for zSeries Integrated Cryptography Appendices

z90crypt Key Structures

The ica_rsa_modexpo_t structure

- The ica_rsa_modexpo_t structure can be found in z90crypt header file 'z90crypt.h', along with the other structures for use with z90crypt. The (private) key consists of exponent in *b_key and the modulus in *n_modulus. Both of these are hexadecimal representations of large numbers. The length L of *n_modulus must be in the range 64 – 256.

- The input data and the exponent b_key must both be of length L. The output data must be of length L or greater. In all these cases, padding on the left with zeroes is allowed.

- There are mathematical rules for the construction of public/private key-pairs, constraining the modulus and exponent in each member of a pair, but we omit these, as z90crypt will not generate key-pairs anyway. See http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/ for a summary of the mathematics.

z90crypt Key Structures

The ica_rsa_modexpo_crt_t structure

- The only formal difference between this structure and the previous one is in the definition of the key. This is defined so that the Chinese Remainder Theorem can be used in decryption/encryption. z90crypt decrypts about 4 times faster if the CRT definition is used. The key-definition fields are all in hexadecimal representation. They have these meanings and limitations:

  - v *bp_key, *bq_key: Prime factors of the modulus. In z90crypt the modulus is (*bp_key) * (*bq_key). The resulting length L of the modulus, in hexadecimal representation, must be found before these fields are defined.
  - v *np_prime, *nq_prime: Exponents used for *bp_key and *bq_key respectively.
  - v *u_mult_inv: A coefficient used in the z90crypt implementation of decryption by CRT.
  - v *bp_key, *np_prime, *u_mult_inv must all be of length 8 +L/2
  - v *bq_key, *nq_prime must both be of length L/2

- The input data length must be L, and the output data length must be at least L, as in ica_rsa_modexpo_t.

Disabling/Enabling Crypto

Disabling crypto

For test or trouble shooting purposes, you might want to disable a cryptographic device. You can do this by editing the /proc/driver/z90crypt file with the vi editor. Proceed like this to disable a cryptographic device:

1. Open /proc/driver/z90cryp with vi. You will see several lines including two lines like this:

```
Mask of online devices: 1 means PCICA, 2 means PCICC
    2200000000000000 0000000000000000 0000000000000000 0000000000000000
```

The lower line represents the physical arrangement of the cryptographic devices with digits 1 and 2 representing PCICA and PCICC cards, respectively.

2. Overwrite the digit that represents the card you want to disable with a character d. To disable the card in the second position or our example overwrite the second 2:

```
Mask of online devices: 1 means PCICA, 2 means PCICC
    2d00000000000000 0000000000000000 0000000000000000 0000000000000000
```

3. Close and save /proc/driver/z90cryp. Confirm that you want to save your changes even if the content of the file has changed since you opened it.

Disabling/Enabling Crypto

To enable a disabled device proceed like this:

1. Open /proc/driver/z90cryp with vi. You will see two lines like this:

```
Mask of online devices: 1 means PCICA, 2 means PCICC
    2d000000000000000 0000000000000000 0000000000000000 0000000000000000
```

   Each d in the second line represents the disabled device. In our example, the device in the second position has been disabled.

2. Overwrite the d that represents the device you want to enable with an e:

```
Mask of online devices: 1 means PCICA, 2 means PCICC
    2e000000000000000 0000000000000000 0000000000000000 0000000000000000
```

3. Close and save /proc/driver/z90cryp. Confirm that you want to save your changes even if the content of the file has changed since you opened it. The device driver replaces the e with the digit for the actual device.

Crypto Support For 31-bit Emulation

31-bit applications can access the 64-bit z90crypt driver by using the 31-bit emulation feature.

The best way to build a 31-bit emulation support for hardware cryptography is to install a 31-bit Linux system and a second with a 64-bit Linux system and the necessary emulation layer.

1 -Compile all necessary programs (like libica, pkcs11, openssl and apache) under the 31-bit system.

2 -Copy the 31-bit versions of the following files to the directory for 31-bit emulation in your distribution. You can find both files on your 31-bit system in the /usr/lib directory. The files are:

libica.so

libcrypto.so

3 -Copy the rest of your applications (like openssl, apache, ...) to your target 64-bit system (including the 31-bit emulation layer).

4 -Now you should be able to run the test programs from OpenSSL, such as:

openssl speed rsa -engine ibmca

The Public Key Cryptography Standards (PKCS)

- 'Informal standards' developped by RSA Laboratories. Used today in many different standards and products

general syntax of structures exchanged between entities as encrypted data, digested data, signed data, ...

general syntax for certification request : distinguished name, public key, set of attributes. Collectively signed by the requesting entity

syntax for transfer of personal identity information, private keys, certificates, ...

PKCS #1: RSA Cryptography Standard
PKCS #2: obsoleted
PKCS #3: Diffie-Hellman Key Agreement Standard
PKCS #4: obsoleted
PKCS #5: Password-Based Cryptography Standard
PKCS #6: Extended-Certificate Syntax Standard
PKCS #7: Cryptographic Message Syntax Standard
PKCS #8: Private-Key Information Syntax Standard
PKCS #9: Selected Attribute Types
PKCS #10: Certification Request Syntax Standard
PKCS #11: Cryptographic Token Interface Standard
PKCS #12: Personal Information Exchange Syntax
PKCS #13: Elliptic Curve Cryptography Standard
PKCS #15: Cryptographic Token Information Format

http://www.rsasecurity.com/rsalabs/pkcs

Linux for zSeries Cryptography Bibliography

Linux for zSeries Device Drivers and Installation Commands –
Linux kernel 2.6 – October 2005 stream

ftp://www6.software.ibm.com/software/developer/linux390/docu/l26cdd00.pdf

openCryptoki docs
http://sourceforge.net/projects/opencryptoki/
http://www-128.ibm.com/developerworks/security/library/s-pkcs/index.html

IBM zSeries 990 Cryptographic Coprocessor Configuration
http://www.redbooks.ibm.com/redbooks/pdfs/sg246310.pdf

zSeries Crypto Guide Update
http://www.redbooks.ibm.com/redbooks/pdfs/sg246870.pdf

Processor Resource/Systems Manager Planning Guide - SB10-7036

z/VM CP Planning and Administration Version 5 Release 1.0 - SC24-6083

z/VM Directory Maintenance Facility Tailoring and Administration Guide - SC24-6084

z/OS Integrated Cryptographic Service Facility Overview - SA22-7519

Linux for zSeries Cryptography Bibliography

Linux on IBM zSeries and S/390: Best Security Practices - SG24-7023

z/VM 5.1 TCP/IP Planning and cutomization - SC24-6125