IBM

Technical Marketing Competence Center

Cryptography with Linux for System z
Clear key vs. secure key cryptography

1st European IBM /GSE conference 2007
for z/VSE, z/VM and Linux on System z

Session SD21, Tuesday 16th October 2007

Dr. Manfred Gnirss
Technical Marketing Competence Center Europe
gnirss@de.ibm.com

Trademarks

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

APPN*	IBM*	Resource Link
CICS*	IBM eServer	S/390*
DB2*	IBM logo	S/390 Parallel Enterprise Server
DB2 Connect	IMS	Sysplex Timer*
e-business on demand	iSeries	System z
e-business logo*	Multiprise*	TotalStorage
Enterprise Storage Server	NetView*	VM/ESA*
ESCON*	OS/2*	VSE/ESA
FICON	OS/390*	WebSphere*
FICON Express	Parallel Sysplex*	z/Architecture
GDPS*	PR/SM	z/OS*
Geographically Dispersed Parallel Sysplex	Processor Resource/Systems Manager	z/VM*
HiperSockets	pSeries	zSeries*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

UNIX is a registered trademark of The Open Group 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.

* 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.

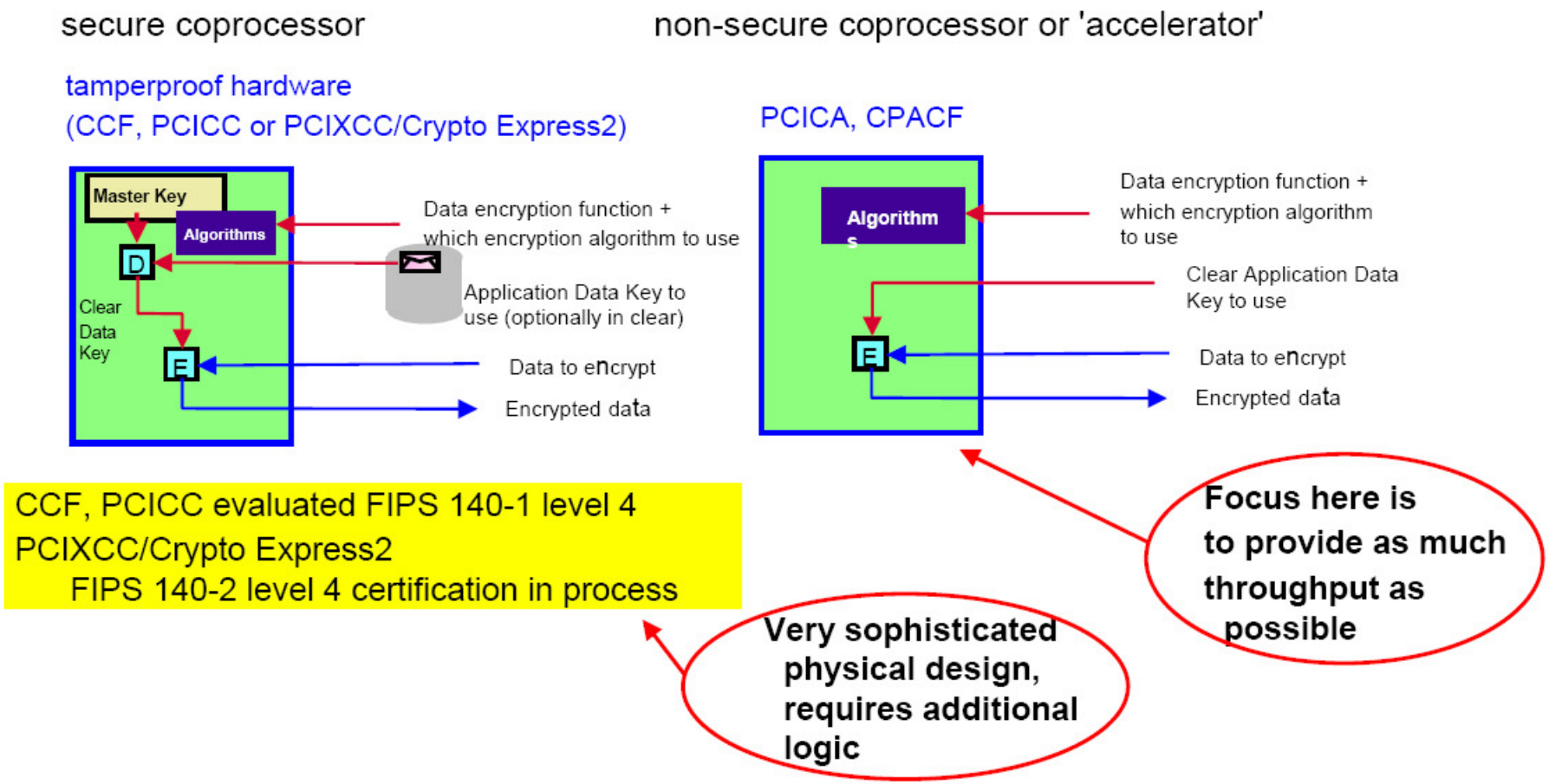
Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products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

Agenda

- Introduction
- System z9 hardware setup and configuration
- z/VM considerations
- Hardware Cryptography with Linux for System z
  - Software and hardware crypto access
  - Clear key cryptography
    - In-kernel crypto
    - z90crypt
    - OpenSSL
    - PKCS#11 – openCryptoki
    - Java
  - Secure key cryptography
- Summary
- Appendix

Clear key and secure key support



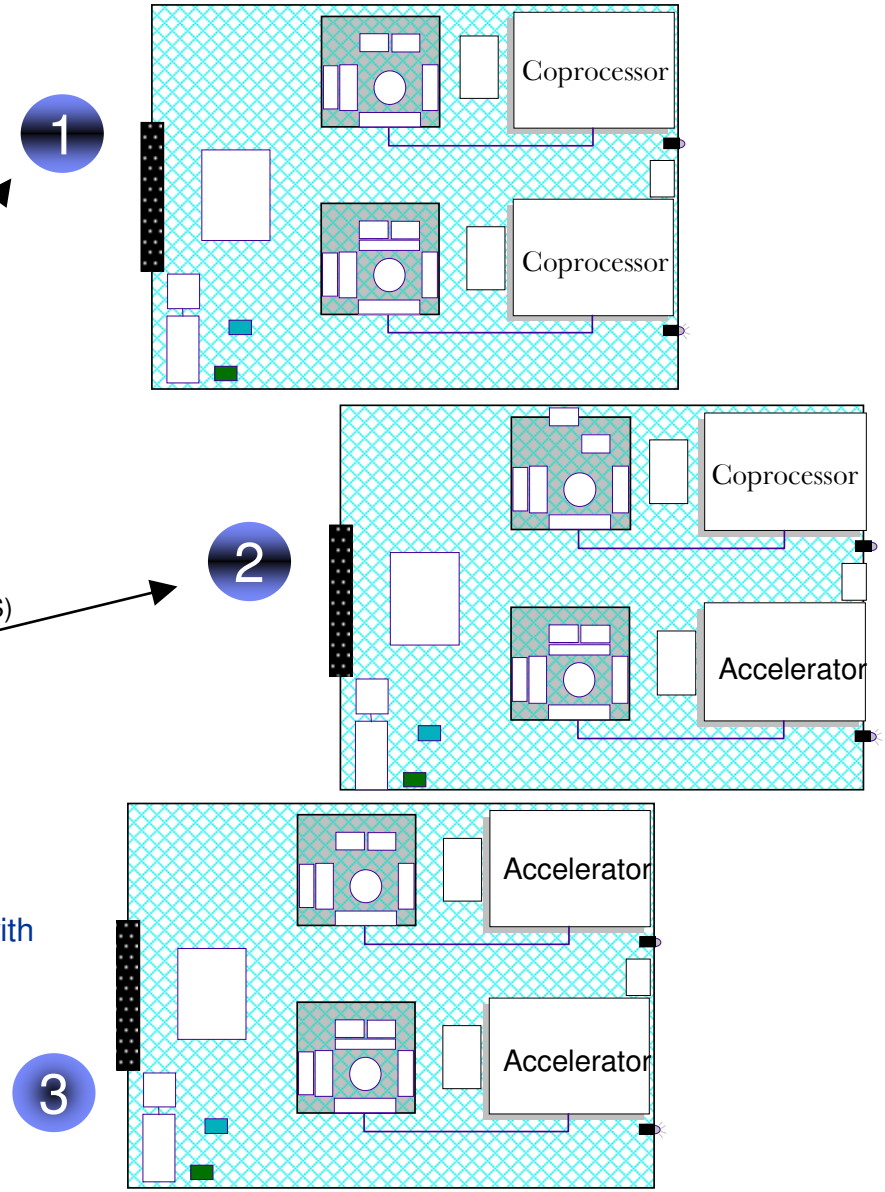
PCIXCC has a two Master Keys: one to protect symmetric keys and another one to protect asymmetric keys

Agenda

- Introduction
- **System z9 hardware setup and configuration**
- z/VM considerations
- Hardware Cryptography with Linux for System z
  - Software and hardware crypto access
  - Clear key cryptography
    - In-kernel crypto
    - z90crypt
    - OpenSSL
    - PKCS#11 – openCryptoki
    - Java
  - Secure key cryptography
- Summary
- Appendix

System z9 hardware crypto

- **CP Assist for Cryptographic Function (CPACF)**
  - Standard on every CP and IFL (since z990, z890)
  - Supports DES, TDES and SHA-1
  - New to System z9 EC and BC:
    - Advanced Encryption Standard (AES-128)
    - Secure Hash Algorithm – 256 (SHA-256)
    - Pseudo Random Number Generation (PRNG)
- **Crypto Express2**
  - Contains 2 PCI-X adapters per feature
  - Two configuration modes
    - Coprocessor (default)
      - Designed for Federal Information Processing Standard (FIPS) 140-2 Level 4 certification
    - Accelerator (configured from the HMC)
  - Three configuration options
    - Default set to Coprocessor
  - Capacity per adapter
    - Configured as coprocessor: ~1000 SSL handshakes/sec
    - Configured as accelerator: ~3000 SSL handshakes/sec
  - April 2007: Announcement of Crypto Express2-1P feature with only 1 PCI-X adapter for IBM System z9 BC
- **TKE workstation with 5.0 level of LIC**
  - Supports configurable Crypto Express2 feature
  - New Graphical User Interface (GUI)
  - Smart Card Reader

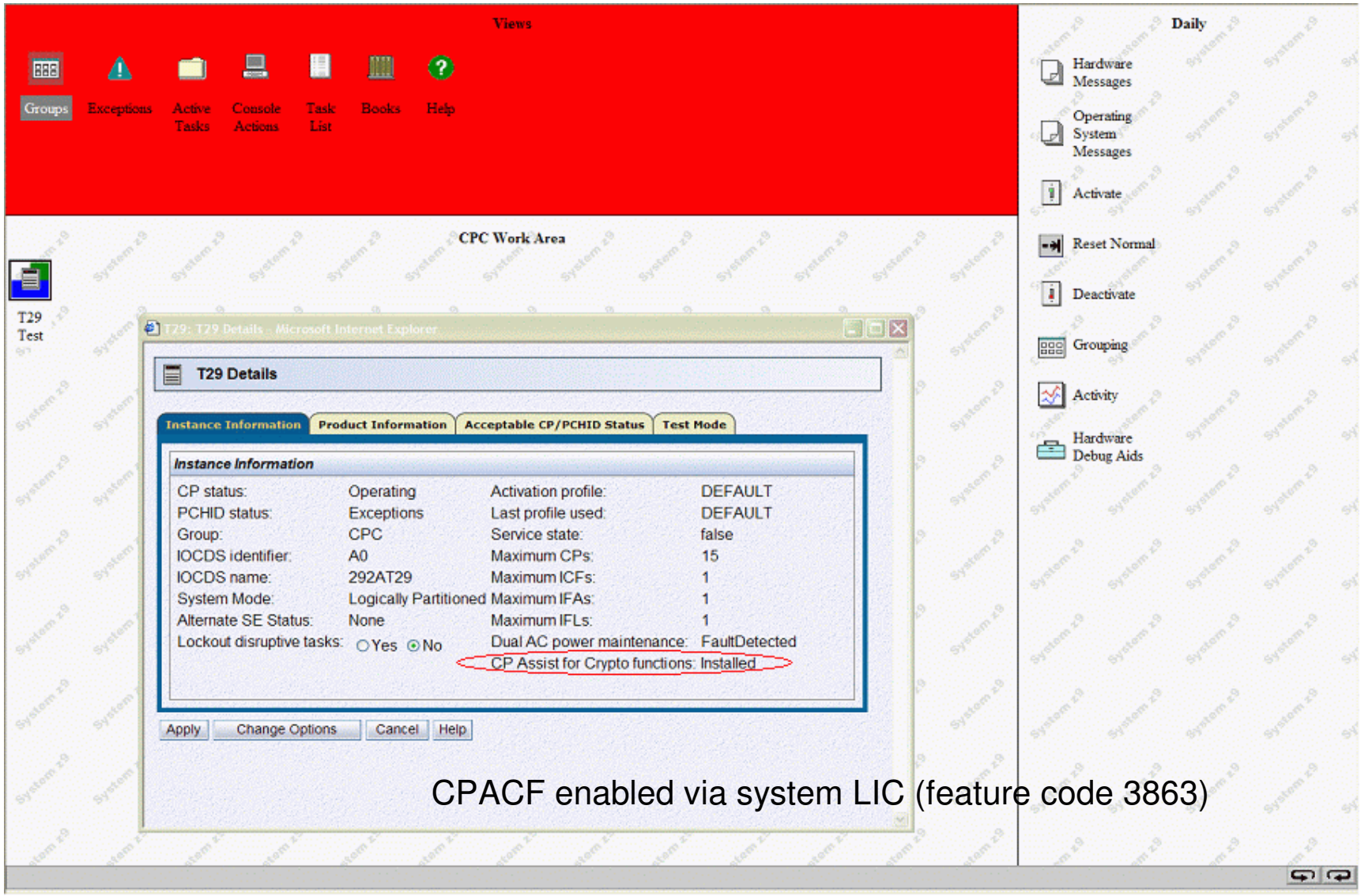


System z9 crypto hardware setup

Careful planning

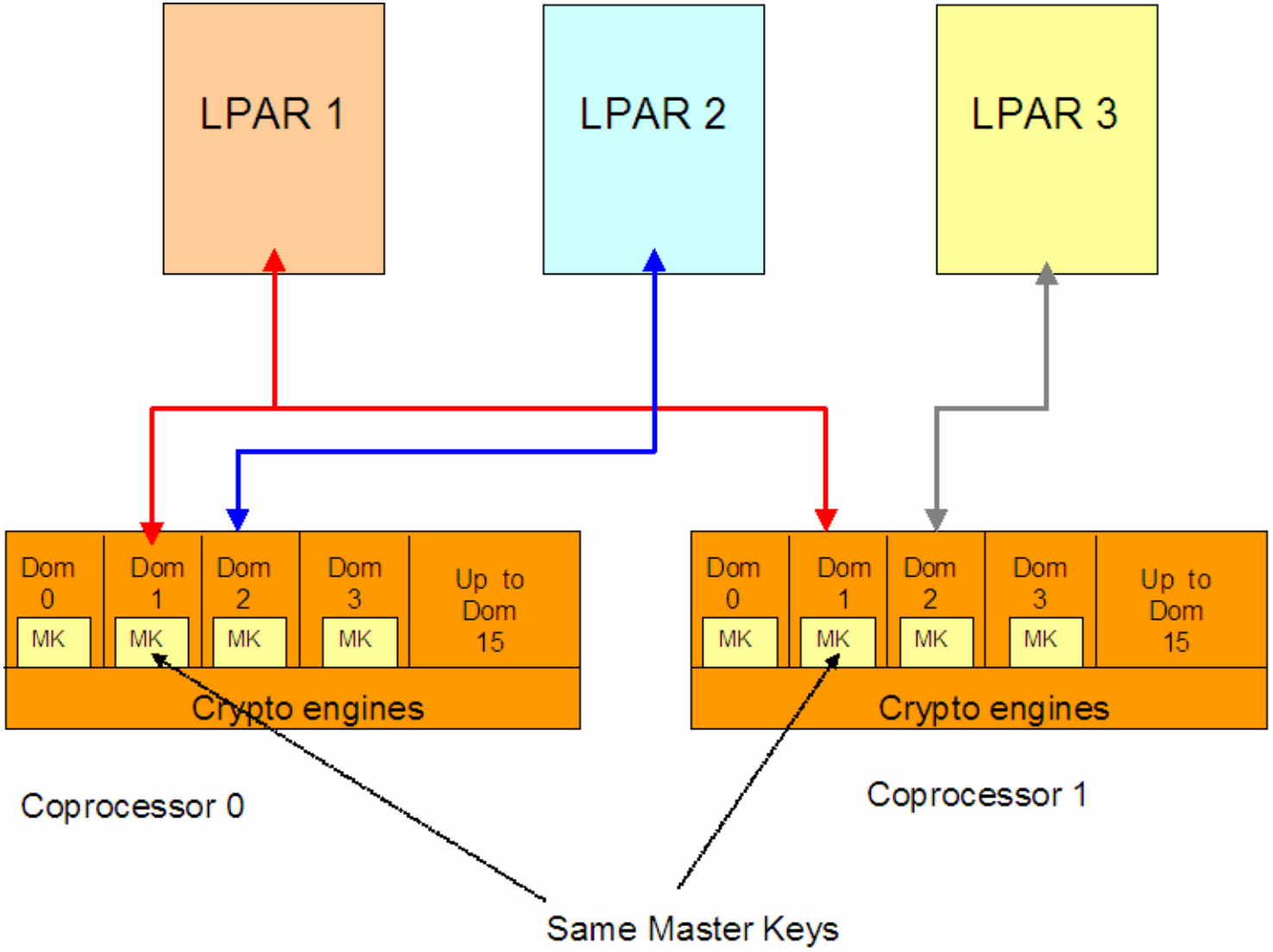
- Esp. if you do not want too often perform LPAR Activate and Deactivate
- Which adapter / domain to which LPAR
- Which LPAR for cyrpto configuration via TKE
- (Master-) key
- Up to 8 features with 2 PCI-X adapters (cards, processors)
  (1 PCI-X adapter per Crypto Express2-1P)
- How many coprocessors, how many accelerators,
- Sharing, redundancy
- You need LIC internal feature 3863 (Crypto Enablement feature)
  - By default: System z9 is delivered without this feature!
  - Installation is non-disruptive.

Crypto enablement feature is installed

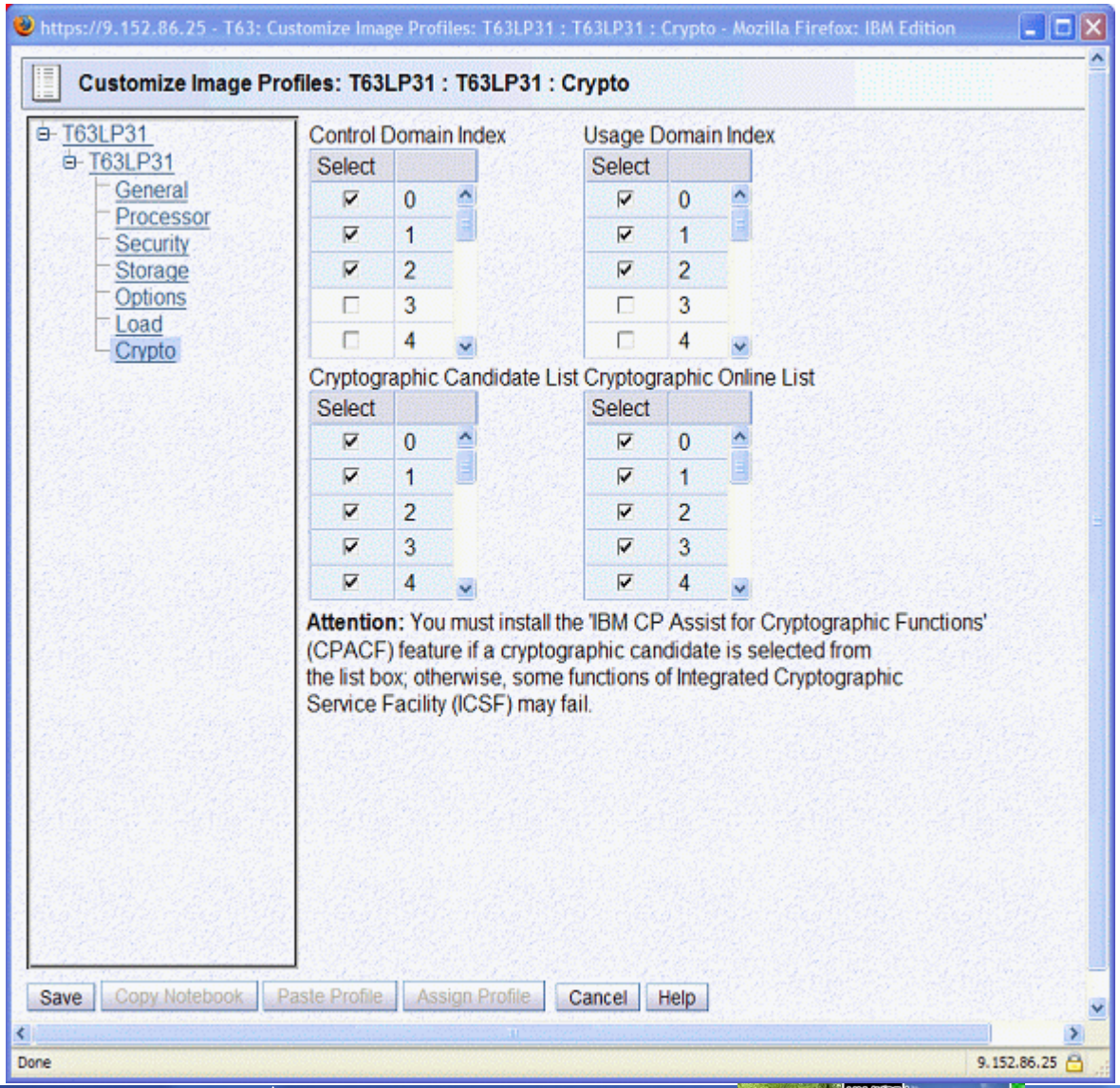


CPACF enabled via system LIC (feature code 3863)

Assign Crypto Domain to LPARs

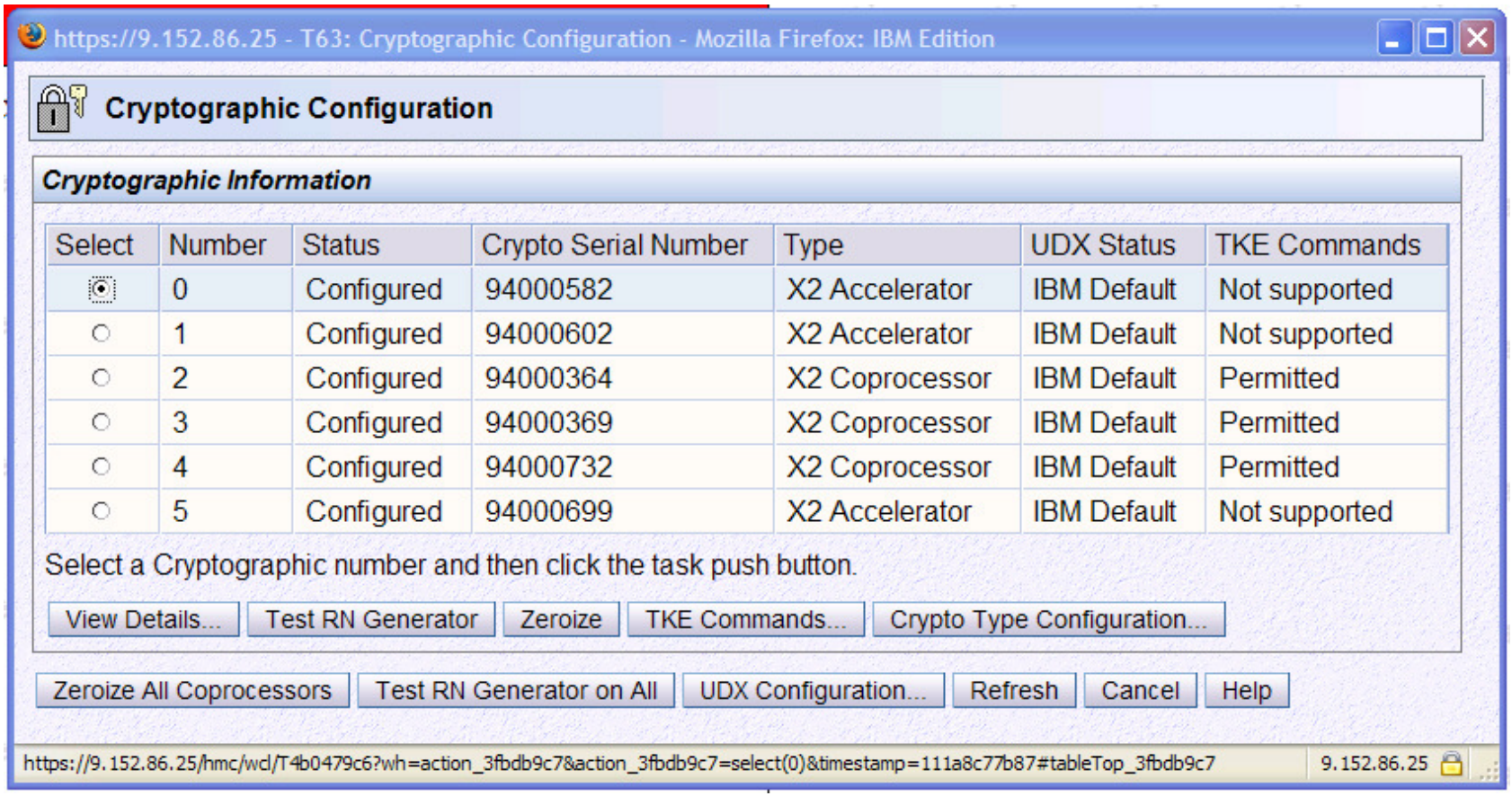


Customize Image Profile



- Combination of Usage Domain Index and PCI-X adapter number must be unique across all active partitions! (exception for backup configurations).
- To newly installed crypto coprocessors numbers are assigned sequentially (during power-on-reset).
- For non-disruptive concurrent installation of a Crypto Express2 feature, out-of-sequence number (from unused range) can be assigned (please inform IBM installation team).
- To dynamically enable a PCI-X adapter to a partition, you need
  - at least 1 usage domain index
  - and coprocessor number must be in the candidate list.
- Changes need partition deactivate-activate!

Crypto Express2: Coprocessor or accelerator



Select	Number	Status	Crypto Serial Number	Type	UDX Status	TKE Commands
◉	0	Configured	94000582	X2 Accelerator	IBM Default	Not supported
○	1	Configured	94000602	X2 Accelerator	IBM Default	Not supported
○	2	Configured	94000364	X2 Coprocessor	IBM Default	Permitted
○	3	Configured	94000369	X2 Coprocessor	IBM Default	Permitted
○	4	Configured	94000732	X2 Coprocessor	IBM Default	Permitted
○	5	Configured	94000699	X2 Accelerator	IBM Default	Not supported

Planning for LPARs, domains and PCI-X adapter numbers

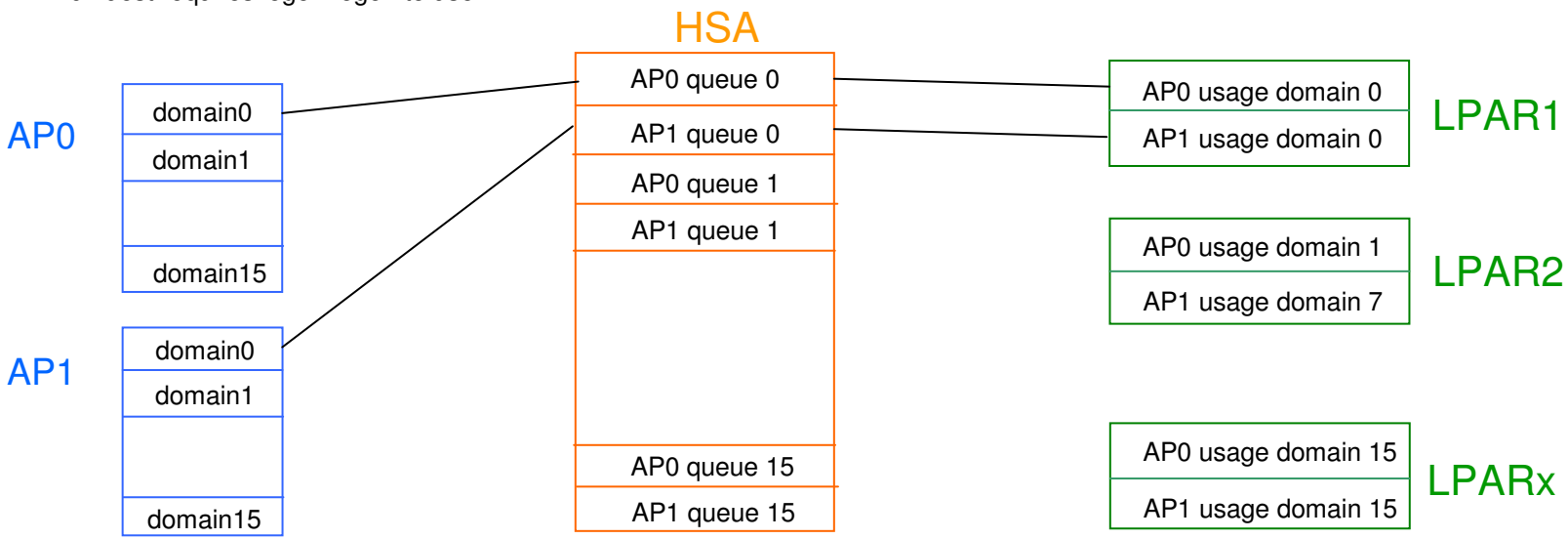
	Adapter Type	Doamin Index 0	Doamin Index 1	Doamin Index 2	…/…	Doamin Index 14	Doamin Index 15
PCI-X Adapter 0	CEX2C/A	LP00 LP02	LP05	LP04		LP04	
PCI-X Adapter 1	CEX2C/A	LP01 LP02					
PCI-X Adapter 2	CEX2C/A	LP00					
…/…							
PCI-X Adapter 14	CEX2C/A						
PCI-X Adapter 15	CEX2C/A						

Agenda

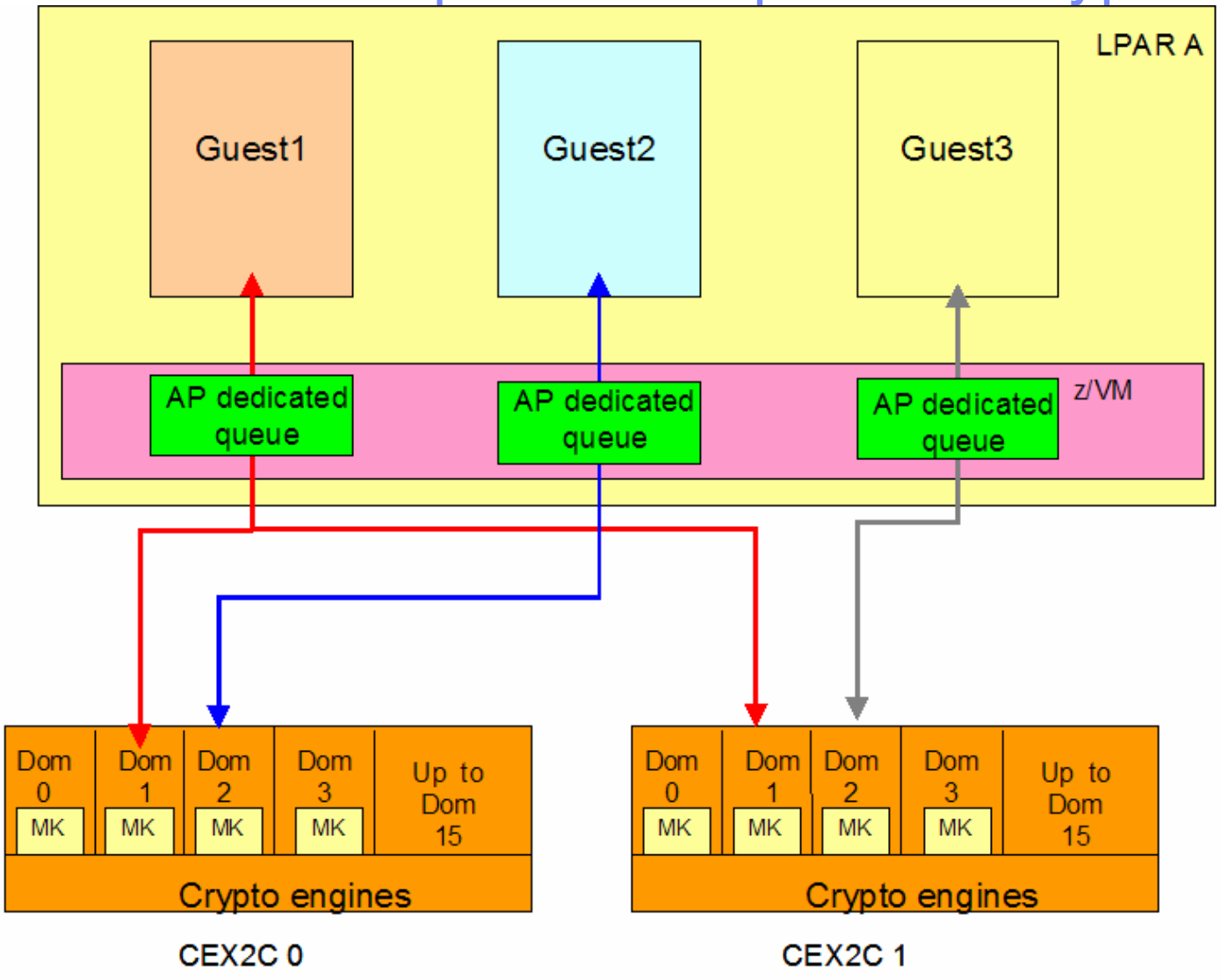
- Introduction
- System z9 hardware setup and configuration
- **z/VM considerations**
- Hardware Cryptography with Linux for System z
  - Software and hardware crypto access
  - Clear key cryptography
    - In-kernel crypto
    - z90crypt
    - OpenSSL
    - PKCS#11 – openCryptoki
    - Java
  - Secure key cryptography
- Summary
- Appendix

z/VM

- Each Adjunct Procesor (AP, coprocessor, card, adapter) can have up to 16 „usage domains“ assigned
- Each usage domain:
  - has a separate set of master key registers (for secure key)
  - is associated with a separate AP queue
- As max 16 PCI-X adapters (APs) and 16 domains there are up to 256 AP queues
- The AP queues reside in HSA (Hardware System Area) provides access to an AP
- AP numbers are assigned to a „candiate list“ or „online list“ in an LPAR activation profile
- Each LPAR is assigned at least one usage domain which apply to all of the APs configured to this LPAR
- An AP can be shared among 16 LPARs
- A usage domain - AP combination must be unique among active LPARs
- AP can be shared or dedicated
- Guests using secure key need dedicated access
- Combination of AP number and Domain should be unique across all active guests
- Hotplugged crypto cards can be used by z/VM w/o IPL (LPAR has to be prepared)
  - Linux uest requires logoff-logon to use



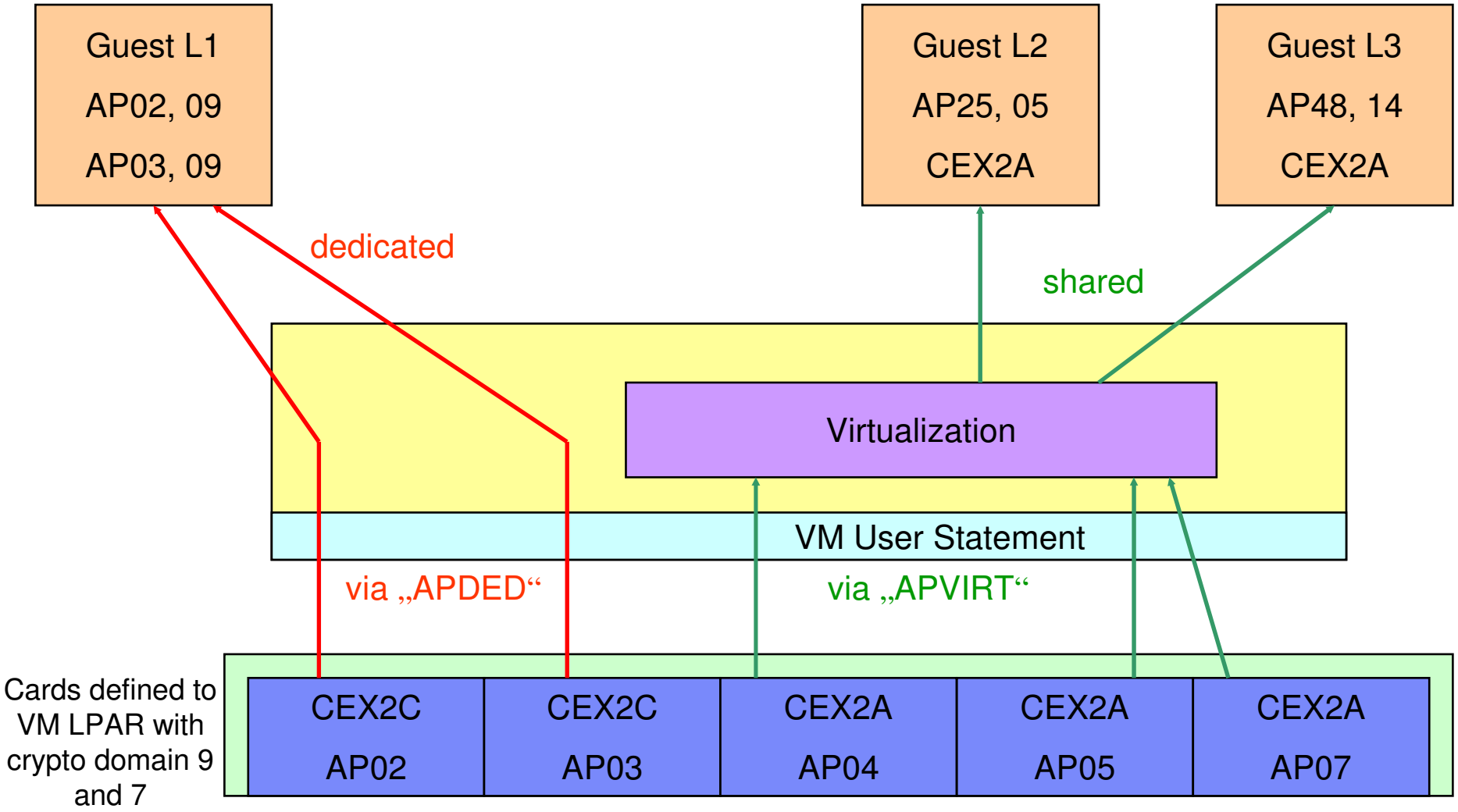
z/VM dedicated queues/adapters for crypto access



z/VM dedicated queues/adapters for crypto access . . .

```
USER GUEST1 xxxxxx 256M 1G G
   INCLUDE IBMDFLT
   IPL CMS
   MACH XA
   NICDEF C200 TYPE QDIO LAN SYSTEM VSWITCH
   CRYPTO DOMAIN 1 APDED 0 1
-- - some lines not displayed - - -
USER GUEST2 xxxxxx 256M 1G G
   INCLUDE IBMDFLT
   IPL CMS
   MACH XA
   NICDEF C200 TYPE QDIO LAN SYSTEM VSWITCH
   CRYPTO DOMAIN 2 APDED 0
- - - some lines not displayed - - -
USER GUEST3 xxxxxx 256M 1G G
   INCLUDE IBMDFLT
   IPL CMS
   MACH XA
   NICDEF C200 TYPE QDIO LAN SYSTEM VSWITCH
   CRYPTO DOMAIN 2 APDED 1
- - - some lines not displayed - - -
```


z/VM dedicated and shared queues/adapters



z/VM dedicated and shared queues/adapters . . .

```
USER GUESTL1 xxxxxx 256M 1G G
   INCLUDE IBMDFLT
   IPL CMS
   MACH XA
   NICDEF C200 TYPE QDIO LAN SYSTEM VSWITCH
   CRYPTO DOMAIN 9 APDED 2 3
-- - some lines not displayed - - -
USER GUESTL2 xxxxxx 256M 1G G
   INCLUDE IBMDFLT
   IPL CMS
   MACH XA
   NICDEF C200 TYPE QDIO LAN SYSTEM VSWITCH
   CRYPTO APVIRT
- - - some lines not displayed - - -
USER GUESTL3 xxxxxx 256M 1G G
   INCLUDE IBMDFLT
   IPL CMS
   MACH XA
   NICDEF C200 TYPE QDIO LAN SYSTEM VSWITCH
   CRYPTO APVIRT
- - - some lines not displayed - - -
```


z/VM: QUERY CRYPTO command

- Displays the status of the crypto units in the processor configuration and status of the domains and AP queues
  (Cyrpto Asyn. Messages (CAM) and Direct Attached Crypto (DAD) refers to server prior to z990, z890).
- Authorization: Privilege clas A,B,C,E

```
cp q crypto
Crypto Adjunt Processor Instructions are installed

cp q crypto ap
AP00 CEX2A Queue 11 is installed
AP01 CEX2A Queue 11 is installed
AP02 CEX2C Queue 11 is superseded by CEX2A
AP02 CEX2C Queue 11 is superseded by CEX2A

cp q crypto ap
AP00 CEX2A Queue 11 is installed
AP01 CEX2A Queue 11 is installed
AP02 CEX2C Queue 11 is reserved for dedicated use
AP02 CEX2C Queue 11 is superseded by CEX2A
```

Agenda

- Introduction
- System z9 hardware setup and configuration
- z/VM considerations
- **Hardware Cryptography with Linux for System z**
  - Software and hardware crypto access
  - Clear key cryptography
    - In-kernel crypto
    - z90crypt
    - OpenSSL
    - PKCS#11 – openCryptoki
    - Java
  - Secure key cryptography
- Summary
- Appendix

Linux

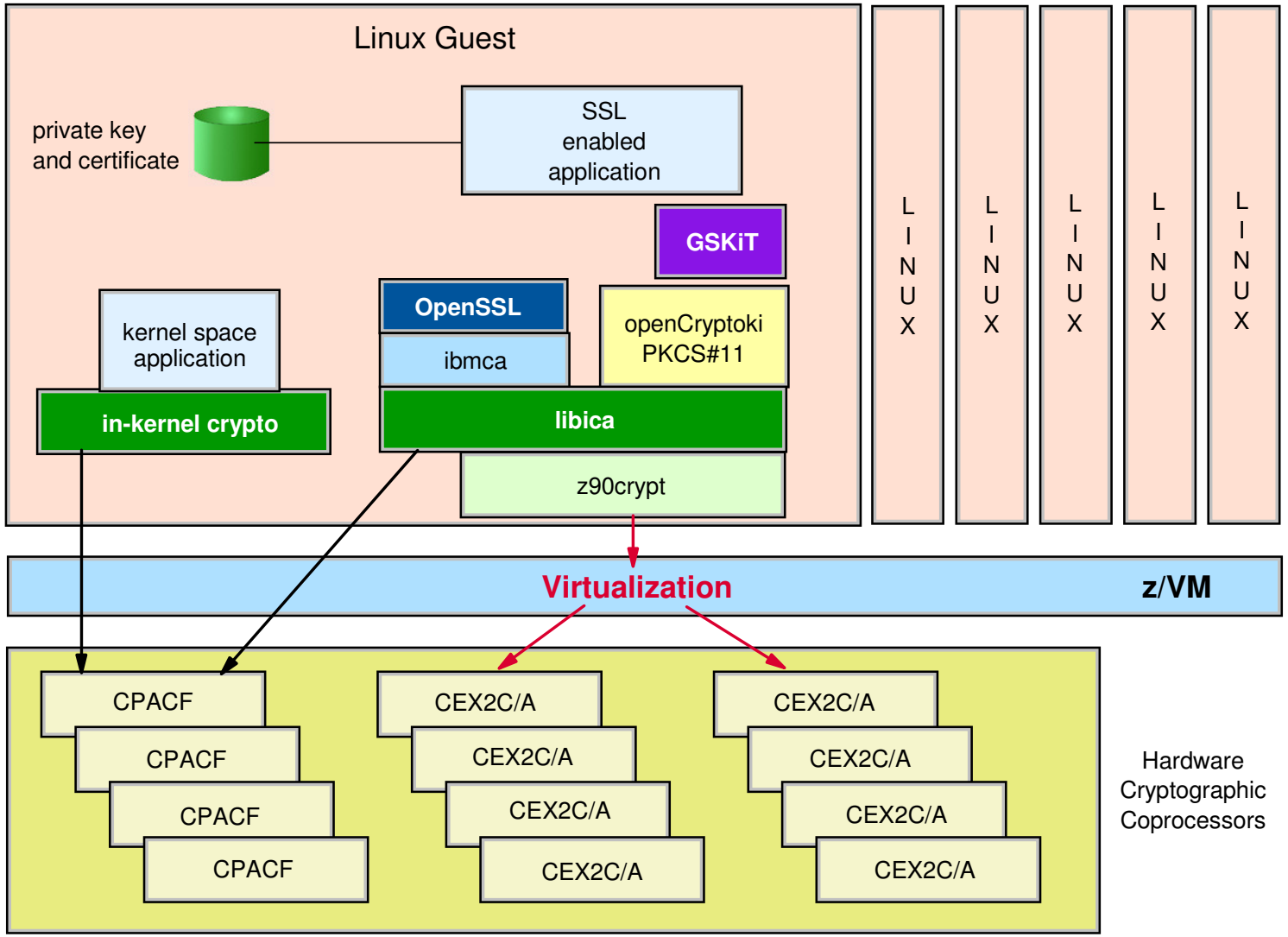
- In-kernel encryption (kernel space)
- Application (user)
- Clear-key – secure key
- Necessary software:
  - z90crypt device driver
  - OpenSSL
  - libica
  - openCryptoki
  - Ibmca engine (OpenSSL engine)
  - CCA libraries (xcryptolinzGA) – this is not an Open Source package.

- Note in the following we refer only to Novell SUSE SLESS 10 SP1

Linux software

```
tmcc-123-168:/home/gnirss # uname -a
Linux tmcc-123-168 2.6.16.46-0.12-default #1 SMP Thu May 17 14:00:09
UTC 2007 s390x s390x s390x GNU/Linux
tmcc-123-168:/home/gnirss # rpm -qa | grep openssl
compat-openssl097g-32bit-0.9.7g-13.5
openssl-devel-0.9.8a-18.15
compat-openssl097g-0.9.7g-13.5
openssl-0.9.8a-18.15
openssl-ibmca-1.0.0-7.11
openssl-32bit-0.9.8a-18.15
openssl-ibmca-32bit-1.0.0-7.11
tmcc-123-168:/home/gnirss # rpm -qa | grep openCryptoki
openCryptoki-2.2.2-24.14
openCryptoki-32bit-2.2.2-24.14
openCryptoki-devel-2.2.2-24.14
openCryptoki-64bit-2.2.2-24.14
tmcc-123-168:/home/gnirss # rpm -qa | grep libica
libica-1.3.7-0.17
libica-32bit-1.3.7-0.17
tmcc-123-168:/home/gnirss # rpm -qa | grep xcrypto
xcryptolinzGA-3.28-rc08
```

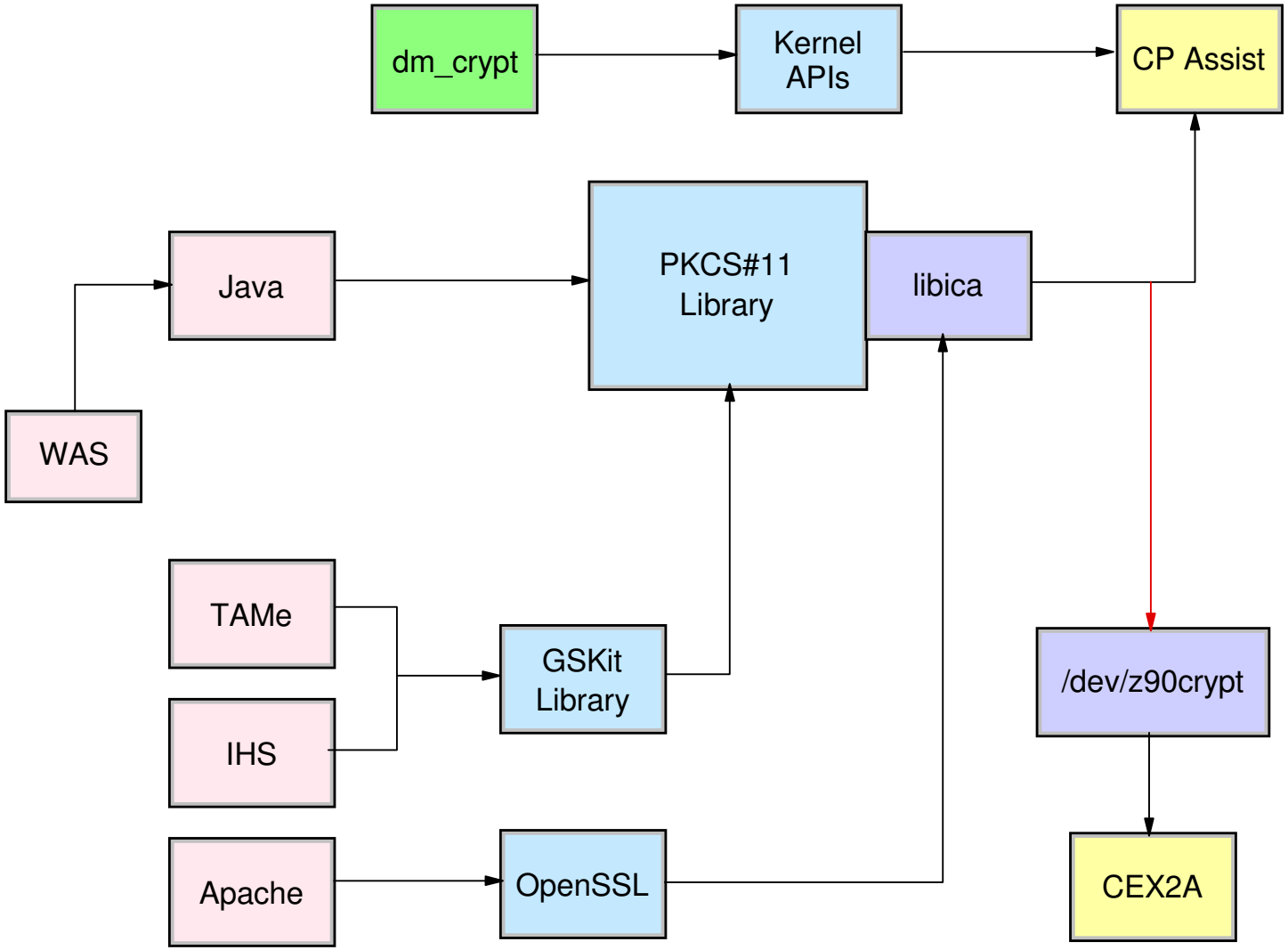

Hardware Crypto Access



Agenda

- Introduction
- System z9 hardware setup and configuration
- z/VM considerations
- Hardware Cryptography with Linux for System z
  - Software and hardware crypto access
  - **Clear key cryptography**
    - In-kernel crypto
    - z90crypt
    - OpenSSL
    - PKCS#11 – openCryptoki
    - Java
  - Secure key cryptography
- Summary
- Appendix

Clear key crypto solution



In-kernel crypto

- Linux kernel version 2.6 provides a set of modules which execute encryption functions by the kernel (kernel –space).
- These functions are built into the kernel as loadable modules.
- IBM provides modules for specific support of System z9:
  des-s390, sha1_s390, sha256_s390, aes_s390, prng
- You need CPACF enabled (feature 3863) to benefit from the support
  - IF CPACF is not enabled, then automatically fall-back into software.
  - CEX2A or CEX2C not necessary.
  - APVIRT or APDED in CRYPTO statement of z/VM Linux user not necessary.
- These modules are already shipped with the Linux distribution (SUSE SLES10 SP1)
- Usage examples:
  - IPSEC for secure communication
  - Disk encryption with dm-crypt and LUKS (Linux Unified Key Setup)

In-kernel crypto

```
gnirss@tmcc-123-168:~> ls /lib/modules/2.6.16.46-0.12-
default/kernel/crypto/
aes.ko          crc32c.ko         michael_mic.ko  tea.ko
anubis.ko       crypto_null.ko    serpent.ko      tgr192.ko
arc4.ko         deflate.ko        sha1.ko         twofish.ko
blowfish.ko     des.ko            sha256.ko       wp512.ko
cast5.ko        khazad.ko         sha512.ko
cast6.ko        md4.ko            tcrypt.ko

gnirss@tmcc-123-168:~> ls /lib/modules/2.6.16.46-0.12-
default/kernel/arch/s390/crypto/
aes_s390.ko             des_s390.ko   sha256_s390.ko
crypt_s390_query.ko     prng.ko
des_check_key.ko        sha1_s390.ko
```


In-kernel crypto

In-kernel crypto modules are loaded on request.

To use System z specific modules, add alias statements in modprobe.conf.local

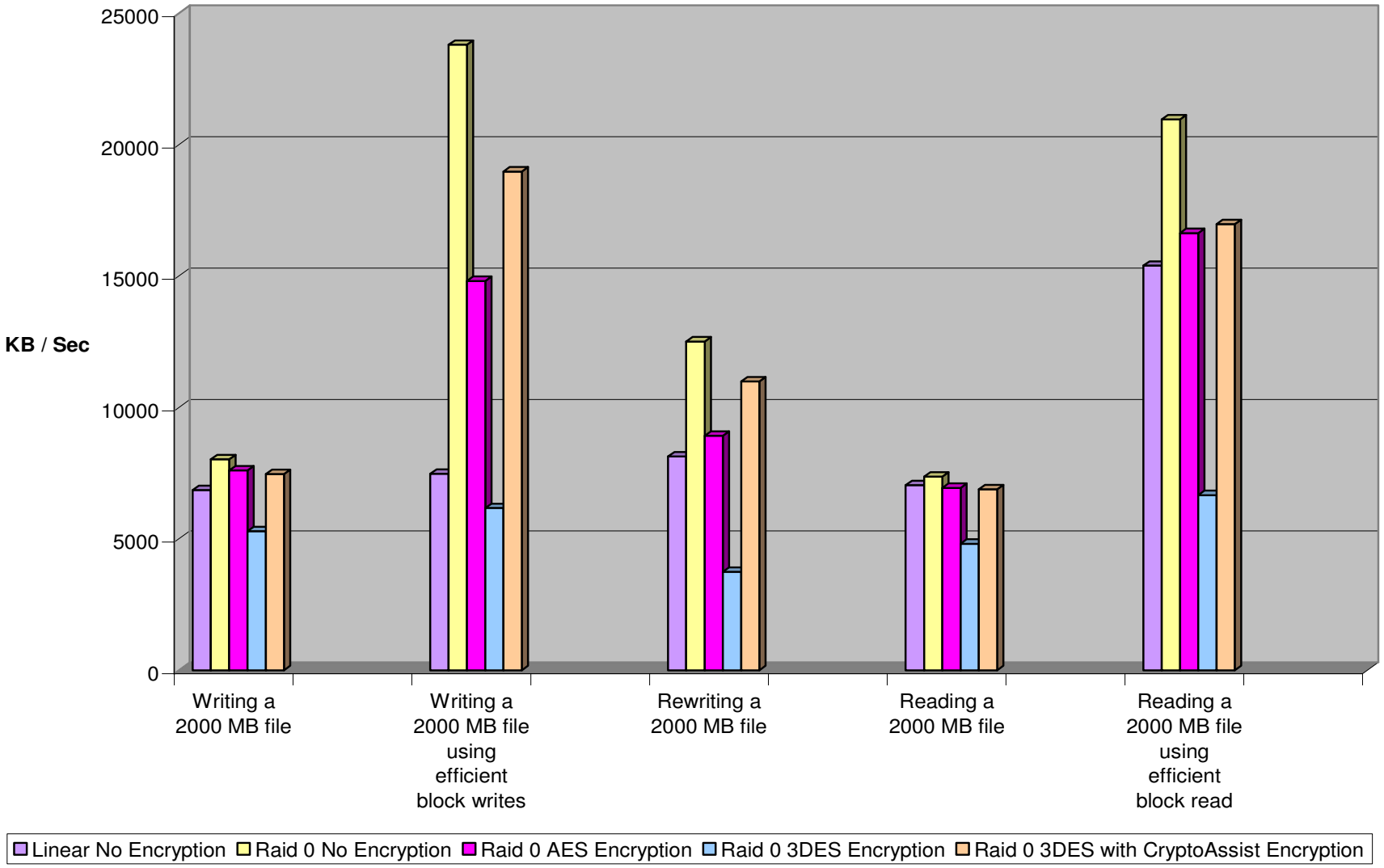
```
gnirss@tmcc-123-168:~> cat /etc/modprobe.conf.local
#
# please add local extensions to this file
# ---- use hardware support for encryption for in-kernel
modules MG 2.10.2007
alias   des     des_s390
alias   sha1    sha1_s390
alias   sha256  sha256_s390
alias   aes     aes_s390
```

To resolve dependencies and to update the definitions:

```
gnirss@tmcc-123-168:~> sudo /sbin/depmod -a
```

If general crypto modules are already loaded, use `rmmod` command for unloading.

In-kernel crypto: Performance example soft vs. CPACF



z90crypt device driver

- Acess to CEX2C and CEX2A for clear key encryption
- Acess to CEX2C for secure key encryption
- z90crypt supports only 1 domain
- z90crypt can select domain automatically
  - Not necessary to specify a domain for clear key
    Domain=-1 (this is the default) is used: Domain with highest number or AP devices (AP queues) is used. If multiple domains with identical (highest) number of AP devices, then domain with lowest number is used.
  - Specify a domain for secure key
- If multiple AP devices, then improved load balancing between devices
- Poll thread to reduce latency for an application while waiting for result of CEX2C or CEX2A execution.
- Poll_thread=1 system is polling for result while waiting (attention, this is CPU intensive)
- Specify domain and poll_thread during load or in /etc/sysconfig/z90crypt
- (modprob, insmod, or script rcz90crypt)
  Don't forget to configure load automatically of z90crypt for boot initialization (via chkconfig z90crypt on)

z90crypt device driver

```
gnirss@tmcc-123-168:~> cat /etc/sysconfig/z90crypt
# The value of -1 is used for autodetect.
Z90CRYPT_DOMAIN=-1

## Description: Turn poll thread on/off
## ## Default:     1
# When running with polling thread one CPU without
# outstanding workload is constantly polling the
# crypthographic requests. The polling thread will
# sleep when no cryptographic requests are currently
# processed. This mode utilize the cryptographic card
# as much as possible at the cost of blocking one CPU.
# Without polling thread the cryptographic cards are
# polled at a much lower rate resulting in higher
# latency and reduced throughput for cryptographic
# requests but without a notable CPU load.
Z90CRYPT_POLL=1
```


z90crypt device driver: status

```
gnirss@tmcc-123-168:~> cat /proc/driver/z90crypt
zcrypt version: 2.1.0
Cryptographic domain: 1
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 1
CEX2A count: 0
requestq count: 0
pendingq count: 0
Total open handles: 1
Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C 6=CEX2A
            050000000000000 000000000000000 000000000000000 000000000000000

Waiting work element counts
            000000000000000 000000000000000 000000000000000 000000000000000

Per-device successfully completed request counts
    00000000 00000143 00000000 00000000 00000000 00000000 00000000 00000000
. . .

gnirss@tmcc-123-168:~> cat /sys/bus/ap/devices/card01/request_count
323
```


OpenSSL

Since OpenSSL version 0.98 (in SUSE SLES 10 SP1) the OpenSSL engine interface has been changed:

- ibmca engine shipped separately (as rpm)
- Engines are automatically loaded (if not built in or already loaded from a specific directory.
- Applications ***can*** use automatically ibmca engine automatically, if support is compiled into them.
- Without dynamic support, ibmca engine ***must*** be requested explicitely.
- Enable dynamic engine support by
  - concatenating openssl.cnf.sample to /etc/ssl/openssl.cnf and
  - move statement openssl_conf=openssl_def to the top of the file.

OpenSSL – dynamic engine loading enabled

```
gnirss@tmcc-123-168:~> cat /etc/ssl/openssl.cnf
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
# - - next line inserted here (moved from above for IBMCA usage - - MG 4.10.2007
openssl_conf = openssl_def
# - - some lines not shown here - - -
# OpenSSL example configuration file. This file will load the IBMCA engine
# for all operations that the IBMCA engine implements for all apps that
# have OpenSSL config support compiled into them.
# Adding OpenSSL config support is as simple as adding the following line to
# the app:
# #define OPENSSL_LOAD_CONF     1
# - - next line put into comments here, and moved to top - - -      MG 4.10.2007
#openssl_conf = openssl_def
[openssl_def]
engines = engine_section
[engine_section]
foo = ibmca_section
[ibmca_section]
dynamic_path = /usr/lib64/engines/libibmca.so
engine_id = ibmca
default_algorithms = ALL
#default_algorithms = RAND,RSA
init = 1
```


OpenSSL speed samples

```
gnirss@tmcc-123-168:~> cat /etc/sysconfig/z90crypt
# . . .
Z90CRYPT_POLL=1
gnirss@tmcc-123-168:~> openssl speed rsa -elapsed
                  sign    verify    sign/s verify/s
rsa  512 bits 0.000953s 0.000829s   1049.7   1206.0
rsa 1024 bits 0.001231s 0.000877s    812.2   1140.9
rsa 2048 bits 0.003248s 0.001018s    307.9    982.6
```

Excellent thoughput, but high CPU usage during execution of test (as of additional polling thread).

```
gnirss@tmcc-123-168:~> cat /etc/sysconfig/z90crypt
# . . .
Z90CRYPT_POLL=0
gnirss@tmcc-123-168:~> openssl speed rsa -elapsed
                  sign    verify    sign/s verify/s
rsa  512 bits 0.010030s 0.010000s     99.7    100.0
rsa 1024 bits 0.010009s 0.010010s     99.9     99.9
rsa 2048 bits 0.010010s 0.010020s     99.9     99.8
gnirss@tmcc-123-168:~> openssl speed rsa -elapsed -multi 8
                  sign    verify    sign/s verify/s
rsa  512 bits 0.001254s 0.001251s    797.7    799.2
rsa 1024 bits 0.001252s 0.001252s    798.4    798.4
rsa 2048 bits 0.002167s 0.001253s    461.4    798.3
```

Throughput depends on test scenario and internal structure, but very low CPU usage during execution of test (as of offloading).

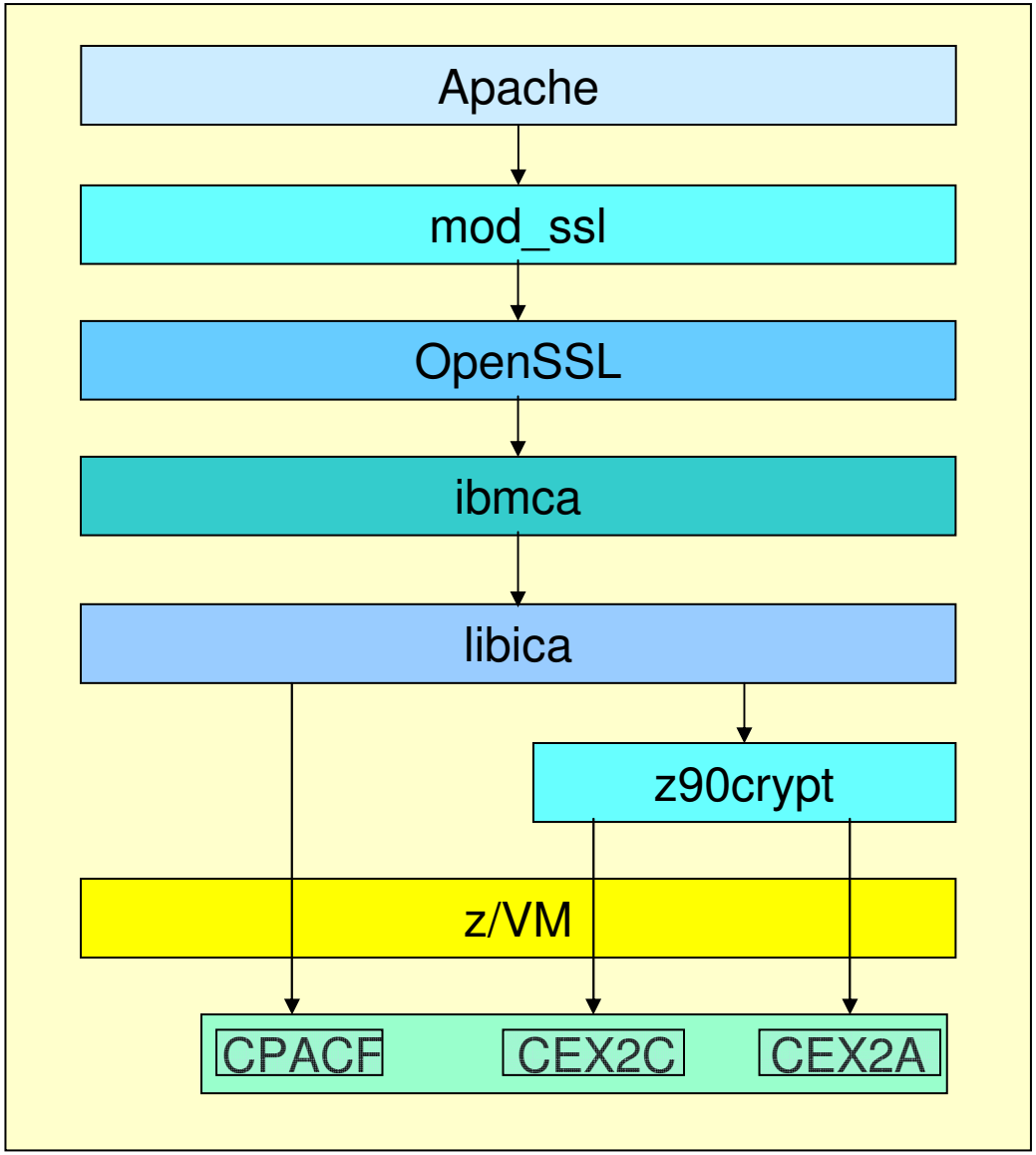
OpenSSL speed samples . . .

```
gnirss@tmcc-123-168:~> openssl speed -evp <cipher>

The 'numbers' are in 1000s of bytes per second processed.

type         sw/hw    16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes

aes-128-cbc           20402.67k    22119.95k    22694.06k    22837.93k    22815.13k
aes-128-cbc CPACF     48711.72k   136130.79k   244753.01k   308116.14k   330540.40k

des-cbc               18089.55k    20101.10k    20613.55k    20725.96k    20824.06k
des-cbc     CPACF     51182.09k   167377.35k   390725.72k   592586.76k   701044.05k

des-ede3               7105.93k     7254.79k     7312.04k     7326.72k     7307.48k
des-ede3    CPACF     49064.34k   123198.92k   202297.69k   235053.59k   248463.36k

sha1                   8845.96k    26396.85k    60219.31k    87987.80k   102200.66k
sha1        CPACF      6264.86k    22925.25k    81874.86k   235188.91k   515279.52k

sha256                 4671.23k    11572.10k    21553.07k    27581.44k    29907.60k
sha256      CPACF      6082.73k    21638.91k    74573.14k   191437.31k   355145.05k
```


openssh

- openssh is a way to provide a secure login to a remote server.
- oenssh uses RSA, DES, Tripple DES, AES, …
- openssh uses OpenSSL
- Today (openssh version 4.2 in SUSE SLES 10 SP1 does not use dynamic engine loading support of OpenSSL and does not provide a way to explicitely specify the engine ibmca.
  (-> all encryption is done in software w/o CEX2x or CPACF)
- Starting with opnessh version 4.4 there is a flag --with-ssl-engine for the configure step to benefit from OpenSSL dynamic engine support.
  - If distributors will build openssh with this new flag then available hardware support with System z is automatically used.

Apache



Apache

- Prepare Apache to use SSL
- Enable Apache to use ibmca OpenSSL engine interface
  - In the ssl-global.conf file:

```
<IfModule mod_ssl.c>
SSLCryptoDevice ibmca
```

- Consider to specify a specific SSL Cipher Suite
  - Adapt SSLCipherSuite in the vhost-ssl.conf file:

    Example:

```
SSLCipherSuite EDH-RSA-DES-CBC3-SHA:DES-CBC3-SHA
```

PKCS#11 - openCryptoki

openCryptoki is Open Source implementation of PKCS#11 interface to provide crypto devices that can manage and store user keys on PKCS#11 devices.
It contains:

- Slot manager daemon (`/usr/sbin/pkcsslotd`)
  - Controls token slots provided to application
  - Managed devices store tokens in the slot manager database
- Slot manager daemon control script (`/etc/init.d/pkcsslotd`)
- API for slot token dynamic link libraries (STDLLs)
  - /usr/lib/opencryptoki/libopencryptoki.so
  - /usr/lib64/opencryptoki/libopencryptoki.so
- Configuration utilities
  - /usr/sbin/pkcs11_startup
  - /usr/sbin/pkcs_slot
  - /usr/sbin/pkcsconf
  - /usr/sbin/pkcsconf64
- STDLLs plugins to the cryptographic adapters
  - /usr/lib/opencryptoki/stdll/PKCS11_ICA.so
  - /usr/lib64/opencryptoki/stdll/PKCS11_ICA.so

PKCS#11 – openCryptoki . . .

- openCryptoki must be configured using pkcs11_startup script to:
  - Create Linux group pkcs11
  - Scan for installed devices (`/dev/z90crypt`)
  - Create slot configuration file (`/var/lob/opencryptoki/pk_config_data`)
- Ensure that z90crypt is loaded before starting pkcs11_startup:
  - As of today, when the IBM ICA token is generated without having z90crypt loaded, all encryption via PKCS#11 will not use Crypto Express2 hardware but will be performed in software only!
  - This token is also necessary to use the CPACF!
- After execution of the pkcs11_startup script start the slot manager daemon

  ```
  rcpkcsslotd start
  ```
- Ensure that the pkcsslotd daemon is loaded after system initialization

  ```
  chkconfig pkcsslotd on
  ```
- Any application that accesses the PKCS subsystem must run as root or under a Linux user that is member of pkcs11 group

PKCS#11 – openCryptoki . . .

- Together, the PKCS#11 layer API and underlying crypto devices is refered to as a PKCS#11 device. Tokens are associated with devices and are used to manage and store user keys. Use pkcsconf command to:
  - Initialize tokens
  - Set Security Officer (SO) PINs.
  - Initialize, set and change user PINs.

- Display token info:

```
pkcsconf -t
```

- Display PKCS#11 info:

```
pkcsconf -i
```

- Display slot info:

```
pkcsconf -s
```

- Initialize the token:

```
pkcsconf -c 0 -I
```

  - Enter the SO PIN (default is 87654321)
  - Enter a token label (– will be refered when you generate keys)

- Change SO PIN (avoid PIN 12345678):

```
pkcsconf -c 0 -P
```

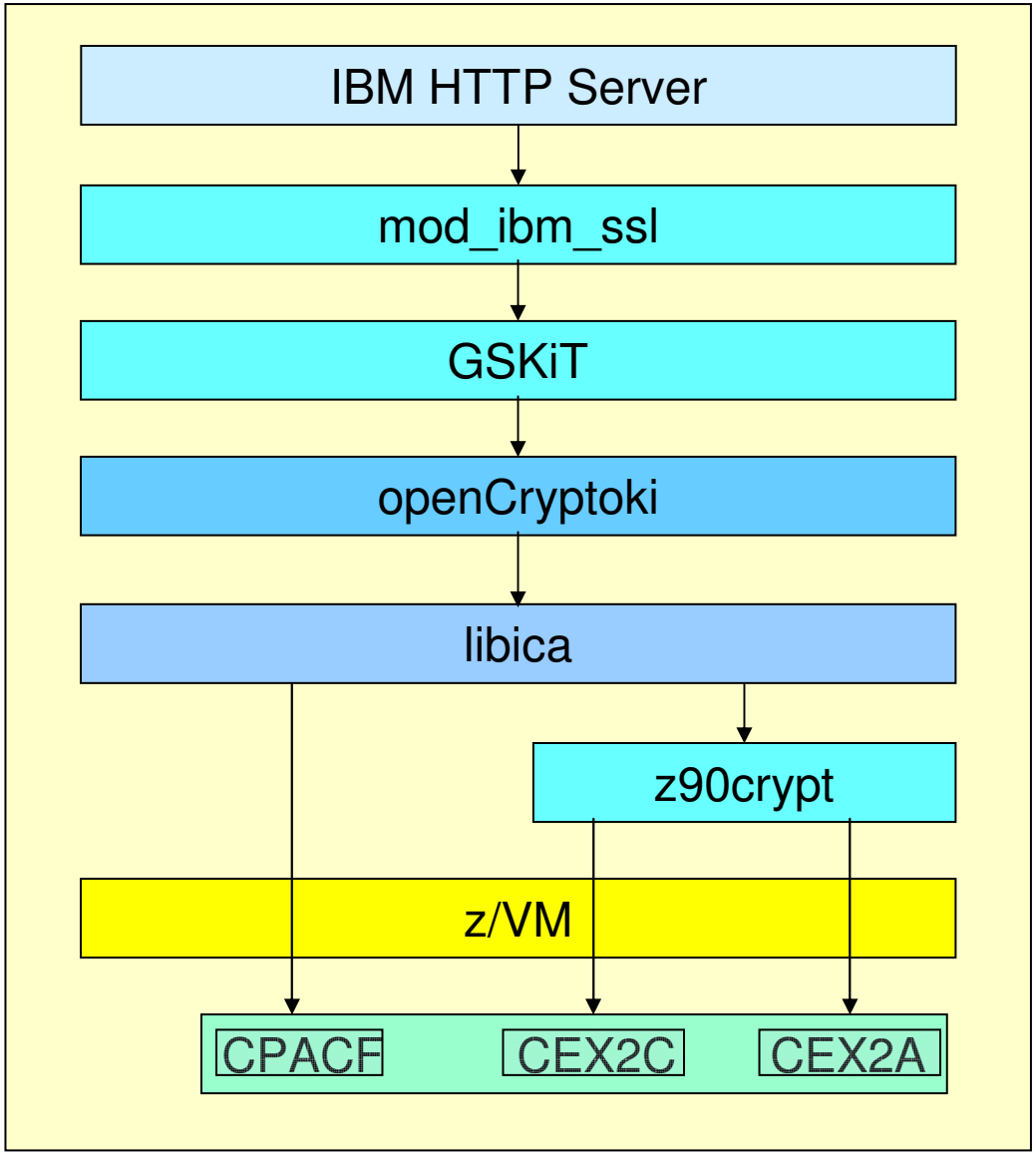
- Set a user PIN

```
pkcsconf -c 0 -u
```

- Change of user PIN

```
pkcsconf -c 0 -p
```


IHS



IHS

- Ensure z90crypt is loaded
- Ensure PKCS#11 subsystem is configured and started
- Use GSKiT to create a server certificate
  - When using ikeyman specify to chose the Cryptographic Token Label of the PKCS#11 device
- Configure IHS for SSL
- In VirtualHost directive in SSLServerCert specify the appropriate label

  Example:

  ```
  SSLServerCert MGCRYPTO:MYCERT
  ```

- Consider to use specific SSL Cipher Suite
  - Adapt virtual host definition

    Example:

    ```
    SSLCipherSpec SSL_RSA_WITH_3DES_EDE_CBC_SHA
    SSLCipherSpec TLS_RSA_WITH_AES_128_CBC_SHA
    ```

Hardware Cryptography via Java

- IBM PKCS11 Implementation provider (IBMPKCS11Impl) uses Java Cryptographic Extension (JCE) and Java Cryptographic Architecture frameworks to seamlessly add capability to use hardware cryptography using Public Key Cryptographic Support 11 (PKCS#11) standard.

- IBMPKCS11Impl provides
  Message Digest, symmetric and asymmetric algorithmes

- z90crypt loaded
- PKCS#11 (openCryptoki) configured, token generated
- Use ikeyman to generate key/certificate
- Initialize the provider IBMPKCS11Impl (using one of three methods: Java Preference method, JAAS Login Module, direct method)
  - Depending on method, adapt the provider list in java.security file
- Run Java application

Agenda

- Introduction
- System z9 hardware setup and configuration
- z/VM considerations
- Hardware Cryptography with Linux for System z
  - Software and hardware crypto access
  - Clear key cryptography
    - In-kernel crypto
    - z90crypt
    - OpenSSL
    - PKCS#11 – openCryptoki
    - Java
  - **Secure key cryptography**
- Summary
- Appendix

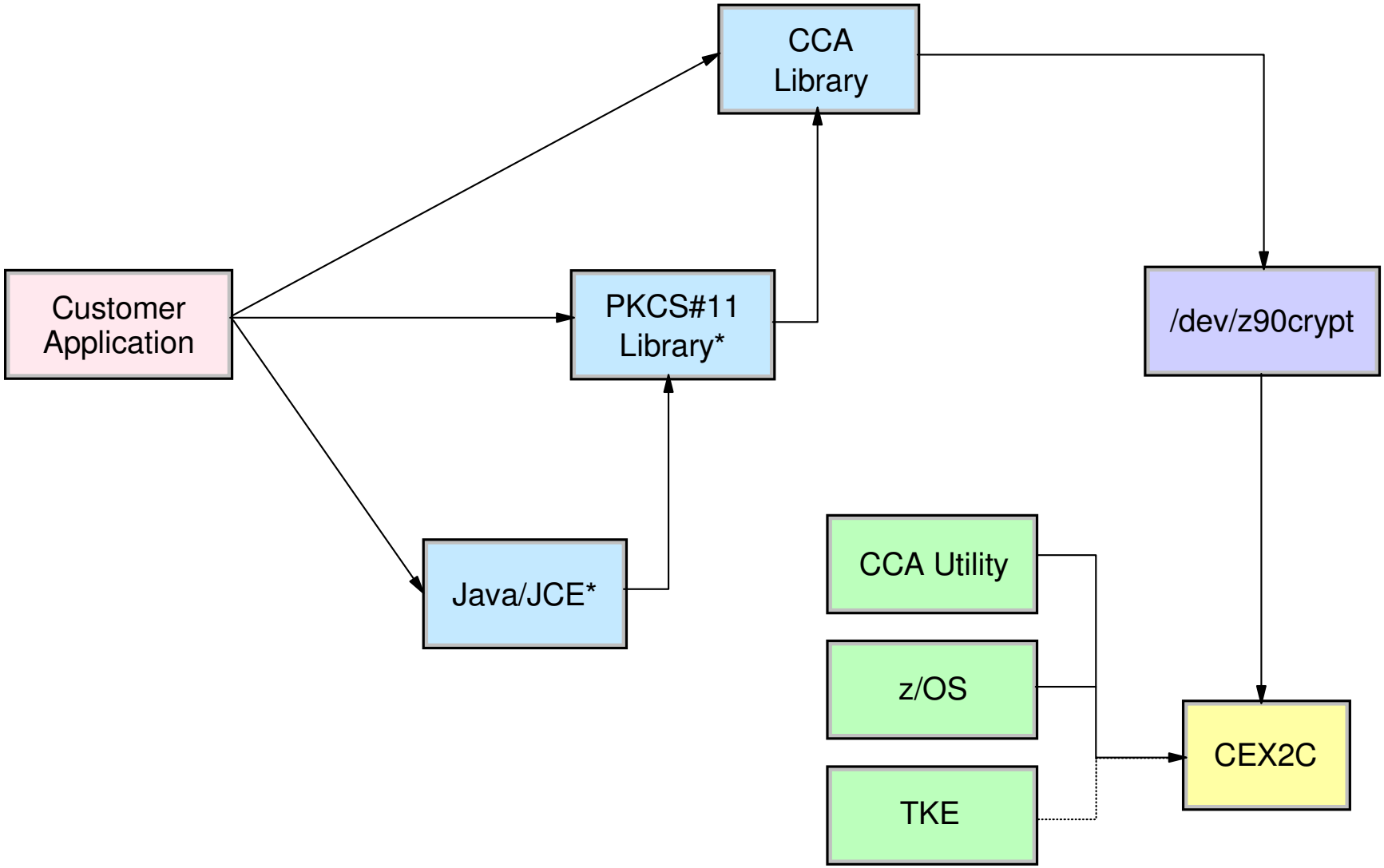
Secure key cryptography

In 2007: Support for secure key cryptography for Linux for System z
Linux can benefit from capabilities of Crypto Express2

Solution consists of:
- Crypto Express2 configured as CEX2C
- Device driver z90crypt
- Common Cryptographic Architecture (CCA) libraries

Management of crypto keys and crypto hardware:
- Using z/OS ICSF
- Using a Trusted Key Entry (TKE) console with connection to a z/OS
- Using a new Linux CCA utility
- Using a Trusted Key Entry (TKE) console with connection to a new Linux CCA utility

Secure key crypto solution



Secure key crypto - Installation

- You need all software as mentioned for clear key cryptography
- Ensure, that CCA libraries are installed

```
gnirss@tmcc-123-168:~> rpm -qa | grep xcrypto
xcryptolinzGA-3.28-rc08
```

- Package xcryptolinzGA-3.28-rc08.s390x.rpm is available from
  http://www.ibm.com/security/cryptocards/pcixcc/ordersoftware.shtml
- Package contains a README.linz file with all relevant information (installation notes, description or syntax, as well as usage notes)
- Content of package:
  - CCA libraries
  - Installation verification program (ivp.e)
  - TKE Catcher (TKEC)
    responds to commands fro a remote TKE workstation
  - Panel CLI (panel.exe)
    is a command line utility to manage keys

Secure key: Installation verification

- ivp.e allows quick and easy verification

Missing permission:

```
gnirss@tmcc-123-168:~> /opt/IBM/4764/bin/ivp.e

/opt/IBM/4764/bin/ivp.e: error while loading shared libraries:
libcsulmkapi.so.1: cannot open shared object file: Permission denied
```

No crypto keys stored in adapter:

```
tmcc-123-168:/home/gnirss # /opt/IBM/4764/bin/ivp.e

ivp - Installation Verification Program

Adapter query STATCCAE  completed successfully.

  CCA version:          z3.25.00

  CCA build date:       20060511

  No symmetric masterkey is loaded!

  No asymmetric masterkey is loaded!

CFQ return code = 0 (0x0000), reason code = 0 (0x0000)
```


Secure key: Installation verification . . .

Crypto keys already stored in adapter:

```
tmcc-123-168:/home/gnirss # /opt/IBM/4764/bin/ivp.e
ivp - Installation Verification Program
Adapter query STATCCAE  completed successfully.
  CCA version:          z3.25.00
  CCA build date:       20060511
  Current symmetric masterkey register contains a key.
  Current asymmetric masterkey register contains a key.
CFQ return code = 0 (0x0000), reason code = 0 (0x0000)
```


Secure key: Installation verification . . .

Device driver z90crypt not loaded:

```
tmcc-123-168:/home/gnirss # /opt/IBM/4764/bin/ivp.e
ivp - Installation Verification Program
An error occurred trying to query the adapter.
CFQ return code = 12 (0x000C), reason code = 338 (0x0152)
```

No crypto queue available:

```
tmcc-123-168:/home/gnirss # /opt/IBM/4764/bin/ivp.e
ivp - Installation Verification Program
RC=80400009 Status=0 errno=9 ThreadID=28f020
An error occurred trying to query the adapter.
CFQ return code = 8 (0x0008), reason code = 1100 (0x044C)
```


Secure key: The CLI panel

- Administration functions of the active crypto card (similar to z/OS with ICSF panels).
- This utility is mainly intended in Linux only environments, where no access to a TKE workstation is available.

```
tmcc-123-168:/home/gnirss # /opt/IBM/4764/bin/panel.exe -?
Panel usage (-k,-a,-g,-x,-l,-s,-c,-q,-t,-o,-?:
        [CC] To determine if a TKE is allowed to administer a
card:
                -k
        [CC] To specify a card other than the Ø-th instance:
             (only useful combined/preceding other args)
                -a <card number>
        To list the current cards available (and basic status):
                -x
```


Secure key: The CLI panel . . .

```
>>>Master Key (MK) Manipulation<<<
        NOTE: -l,-s,-c,-q are mutually exclusive and must be
              (along with sub-options) the last option specified

To LOAD  a Master Key (MK) PART:
                   -l (for interactive)
                or -l -t [A|S] -p [F|M|L] KEYPART
To SET   a Master Key:
                   -s (for interactive)
                or -s -t [A|S]
To CLEAR a Master Key (clears prior key parts in 'New' Register):
                   -c (for interactive)
                or -c -t [A|S]
To QUERY a Master Key Verification Pattern:
                   -q (for interactive)
                or -q -t [A|S] -r [N|C|O]
To initialize a key storage file:
                  -t <type> -f <file> -i
To reencipher key storage:
                  -t <type> -f <file> -r
. . .
```


Secure key: TKE catcher

- The TKE catcher is a program running on Linux for System z that allows remote access from the workstation to administrate crypto cards and the according keys.
- To make use of the TKE catcher, the TKE must be enabled to access the system via s390 SE panel and using port 50003.
- Control Domain Index and TKE commands must be permitted for the used crypto adapters.
- Consider the following 3 cases for using TKE for Linux for System z:
  - Environment with Linux and z/OS LPARs sharing a Crypto Express2 adapters
    - Difficult environment if you intend to use TKE catcher to administrate the crypto queues accessible by Linux and the z/OS TKE for the crypto queues accessible by z/OS. TKE catcher can not figure out whether there is a z/OS LPAR and whether crypto is being configured with z/OS TKE .
    - To avoid conflicts, we recommend to use the z/OS TKE in such an environment.
  - Environment with Linux and z/OS LPARs with each exclusive use of Crypto Express2 adapters
    - Usage of TKE catcher is possible.
    - Note: Situation gets dificult if environment is reconfigured to share adapters.
  - Linux for System z exclusive environment
    - Using TKE with TKE catcher is most secure way to administrate crypto infrastructure.

Agenda

- Introduction
- System z9 hardware setup and configuration
- z/VM considerations
- Hardware Cryptography with Linux for System z
  - Software and hardware crypto access
  - Clear key cryptography
    - In-kernel crypto
    - z90crypt
    - OpenSSL
    - PKCS#11 – openCryptoki
    - Java
  - Secure key cryptography
- **Summary**
- Appendix

Secure key and clear key: Summary

- Using hardware support for encryption with Linux for System z
  - Increases throughput / reduces CPU load
  - Increases security
- Use more than one crypto adapters for redundancy and loadbalancing
- Do not forget disaster recovery aspects also for cryptographic support
- Linux for System z provides a complete infrastructure to use secure key cryptography according CCA and to run 64 bit secure key applications in addition to the established clear key capabilities.
- New generations of applications implementing secure technologies like PIN verification, single sign-on and service oriented architecture can benefit from secure key capabilities and the highly flexible Linux environment.

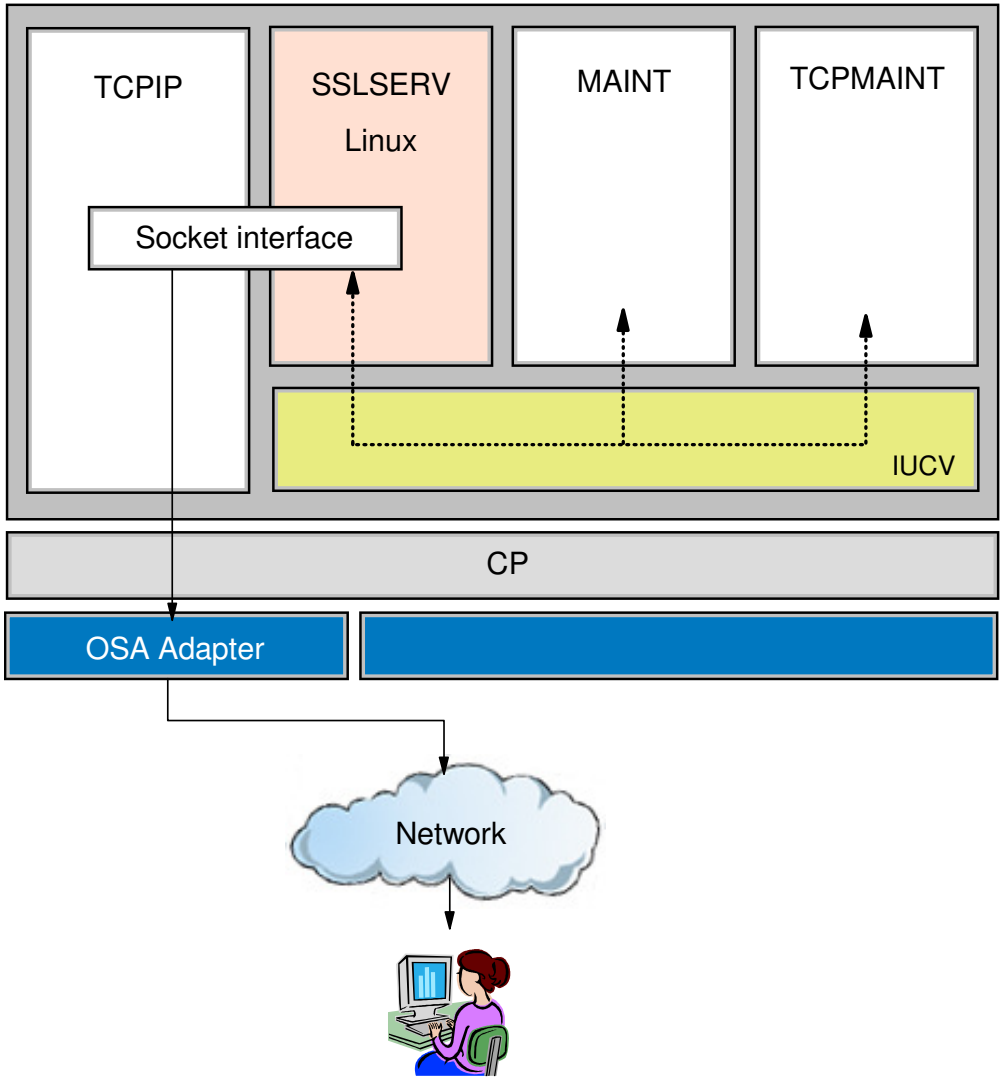
Thank You

Questions?

Agenda

- Introduction
- System z9 hardware setup and configuration
- z/VM considerations
- Hardware Cryptography with Linux for System z
  - Software and hardware crypto access
  - Clear key cryptography
    - In-kernel crypto
    - z90crypt
    - OpenSSL
    - PKCS#11 – openCryptoki
    - Java
  - Secure key cryptography
- Summary
- **Appendix**

Securing SSL connection to z/VM



Where to get more information

- Security on z/VM – Redbooks SG24-7471 – available soon
- z/OS ICSF TKE Workstation User's Guide, SA22-7524
- Linux for System z, Secure Key Solutions with the Common Cryptographic Architecture, Application Programmer's Guide, SC33-8294
- IBM System z9-109 Configuration Setup, SG24-7203
- SSL Server Implementation for z/VM 5.2 - RedPaper 4348
- z/VM TCP/IP Planning and Customization, SC24-6125
- http://www.vm.ibm.com/related/tcpip/vmsslinf.html