Linux on z Systems and LinuxONE Crypto Overview

Dr. Reinhard Buendgen (buendgen@de.ibm.com)

Trademarks

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

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*, AS/400®, e business(logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/390, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* 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.

Why Use z Systems Hardware Cryptography?

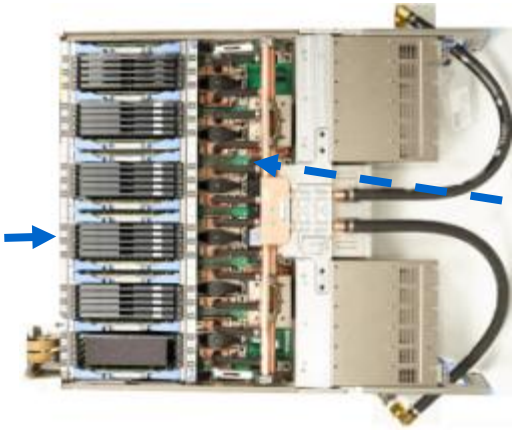
- Trust & reliability
  - Proven implementation in HW
- Cost
  - Security does not come for free: minimize extra cost
    - Save money
      - Off-load expensive CPU workload
    - Save time
      - Faster crypto algorithms
- Ultra high security needed
  - HSMs: secure key with CCA or EP11
- Functionality
  - special build-in security functions for banking and financial applications: secure key
- Regulations
  - FIPS 140-2 certified cryptography adapters

z Systems
HW Cryptography

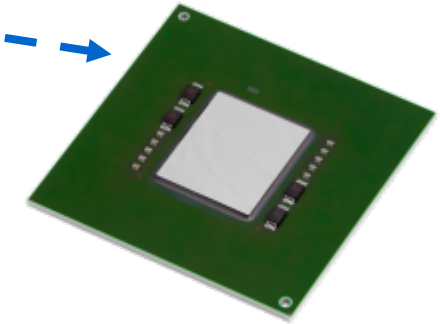
Overview – HW Crypto support in z Systems (z13)

PCIe I/O drawers

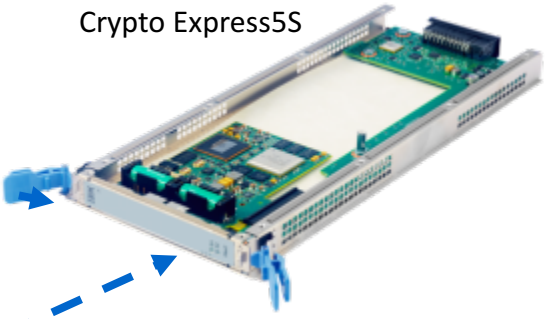
CPC Drawer



PU SCM
Each PU is capable of having the CPACF function



Crypto Express5S

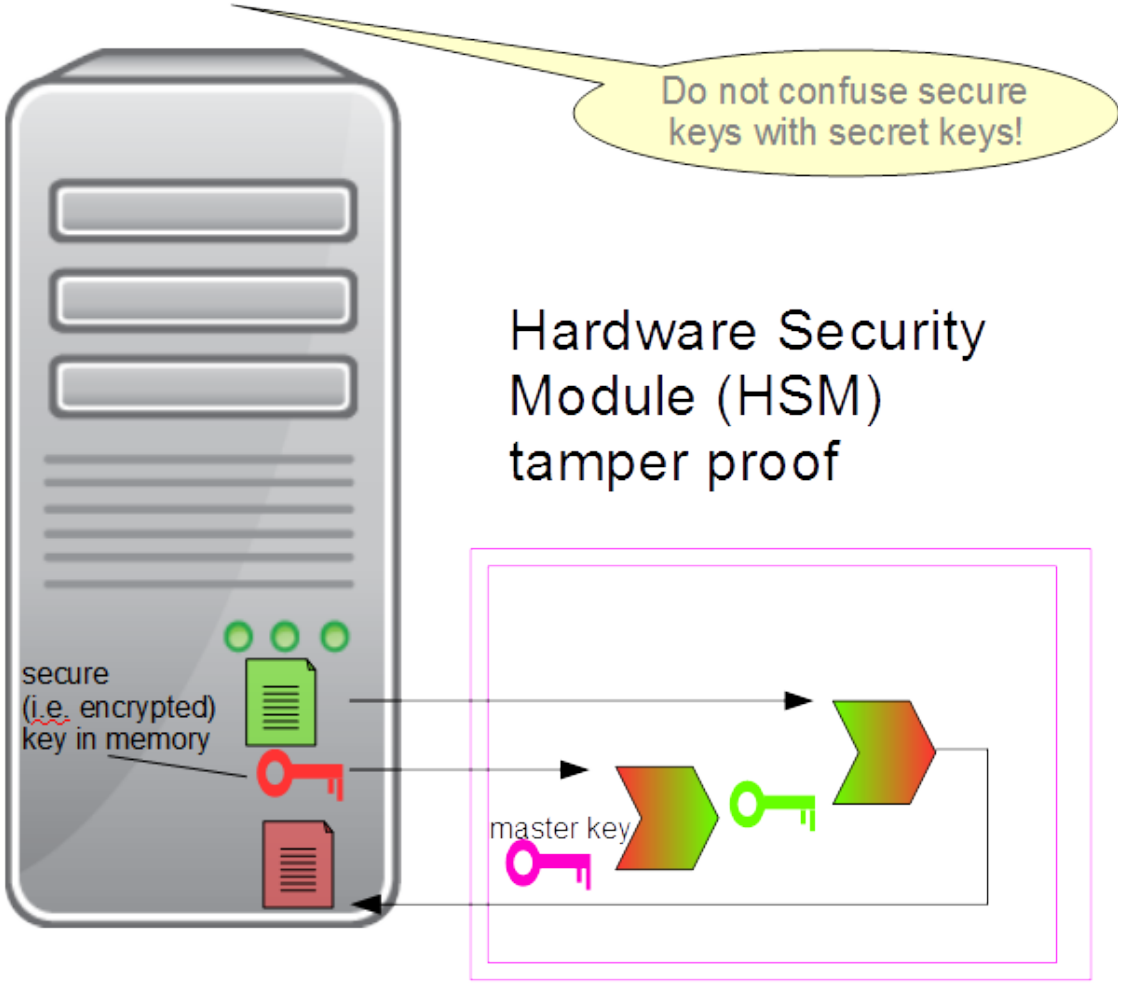


TKE required for management of Crypto Express5S and EP11

Trusted Key Entry (TKE)

Smart Cards

Clear Key vs Secure Key Cryptograpy



System z Crypto HW

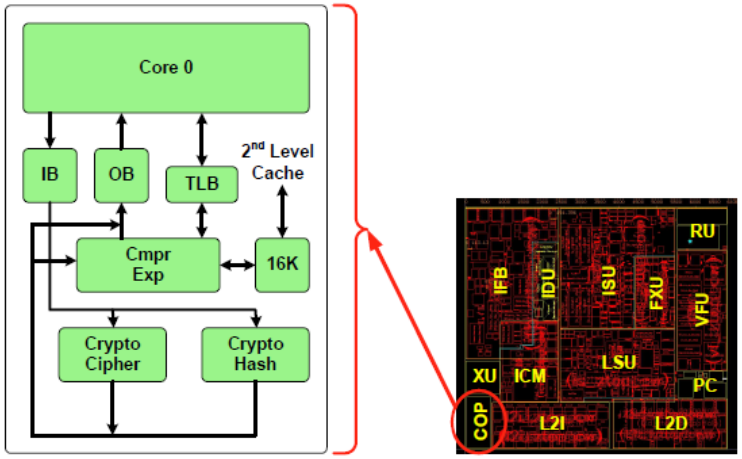
Type	location	Linux names	clear key	secure key	accelerates*
CPACF	CPU	(MSA1 - MSA5)	X	(X)**	SHA 1/2, DES, TDES, AES
Accelerator	Crypto Express adapter	CEX2A – CEX5A	X		RSA (DH, DSA)
CCA Co-Processor	Crypto Express adapter	CEX2C – CEX5C	X	X	RSA (DH, DSA)
EP11 Co-Processor	Crypto Express adapter (starting with CEX4S)	CEX4P - CEX5P		X	--

*) RSA (DH, DSA) is off-loaded off the CPU

**) CPACF offers a technique similar to secure key, called *protected key*. However since the CPACF wrapping keys are not stored in a tamper proof environment, CPACF protected key cryptography does not qualify as a HSM.

CPACF - CP Assist for Cryptographic Functions

- Available on every Processor Unit defined as a CP, IFL, zAAP and zIIP
- non-privileged instructions supporting:
  - hashes: SHA1, SHA-224, SHA-256, SHA-384, SHA-512, GHASH
  - symmetric ciphers: DES, 2DES, 3DES, AES-128, AES-192, AES-256
  - modes of operations: ECB, CBC, CTR, OFB, CFB, XTS, CBC-MAC (CMAC, CCM, GCM)
  - pseudo random number generation: 3DES based PRNG, NIST SP-800-90A SHA-512 based DRNG
- Crypto instructions must be explicitly enabled, using a no-charge enablement feature (#3863),
  - SHA, DRBG algorithms are always avaialble
- Protected key support for additional security of cryptographic keys



Performance improvement for CPACF on z13:

- AES: 2 x throughput of zEC12,
- 3DES: 2 x throughput of zEC12
- SHA: 3.5 x throughput of zEC12

Crypto Express Adapters

- Three different firmware loads
  - Accelerator
  - CCA coprocessor
  - EP11 coprocessor (since CEX4S)
- Adapter virtualization
  - Adapter can be partitioned into different domains (separate master keys per domain)
  - <= CEX4S: 16 domains
  - CEX5S: 85 domains on z13, 40 domains on z13s
- Adapter management via SE or HMC
  - Selection of adapter type (firmware load)
  - Assignment of adapters and domains to LPARs
- Master key management via TKE

CEX5S Support

Toleration Support

- Linux kernel recognizes CEX5S adapter and treats it as CEX4S adapter
- support domains 0 - 84
- new sysfs attribute shows its real identity under `/sys/bus/ap/raw_hwtype`
- new syfs attribute shows max ID of adapter domains: `/sys/bus/ap_max_domain_id`
- supported distributions
  - SLES 11 SP3 + maintenance
  - SLES 12 + maintenance
  - RHEL 7.1
  - RHEL 6.6 + maintenance
  - RHLE 5.11 (only 16 domains)
  - KVM 1.1.1

z/VM prerequisite to support CEX5S adapters

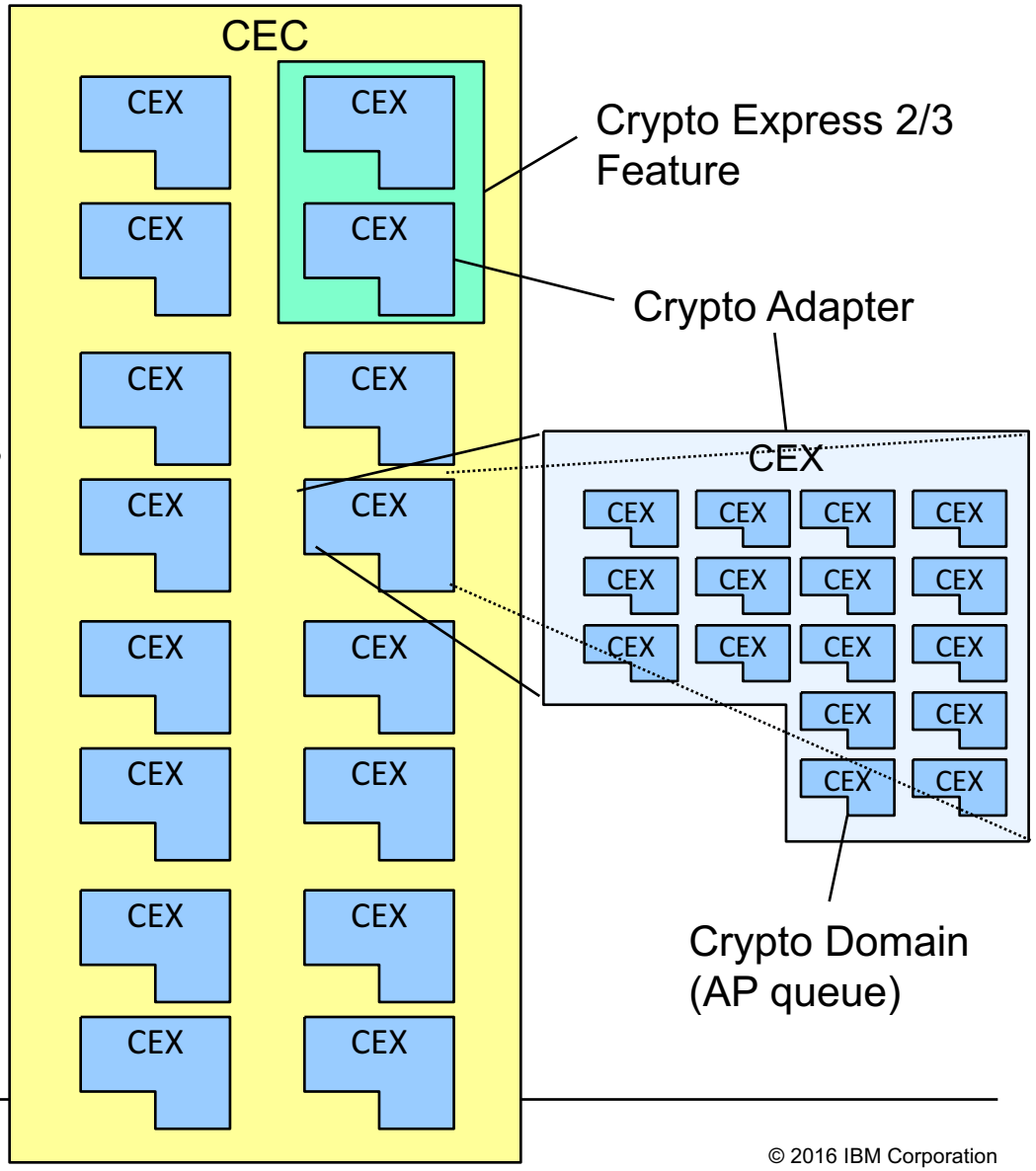
- z/VM 6.2: APAR VM65007
- z/VM 6.3: APAR VM65577

Exploitation Support

- Displays a CEX5S adapter as “CEX5A“, “CEX5C” or “CEX5P”
- supported distributions
  - SLES 12 SP1
  - RHEL 7.2
  - Ubuntu 16.04
  - KVM 1.1.1

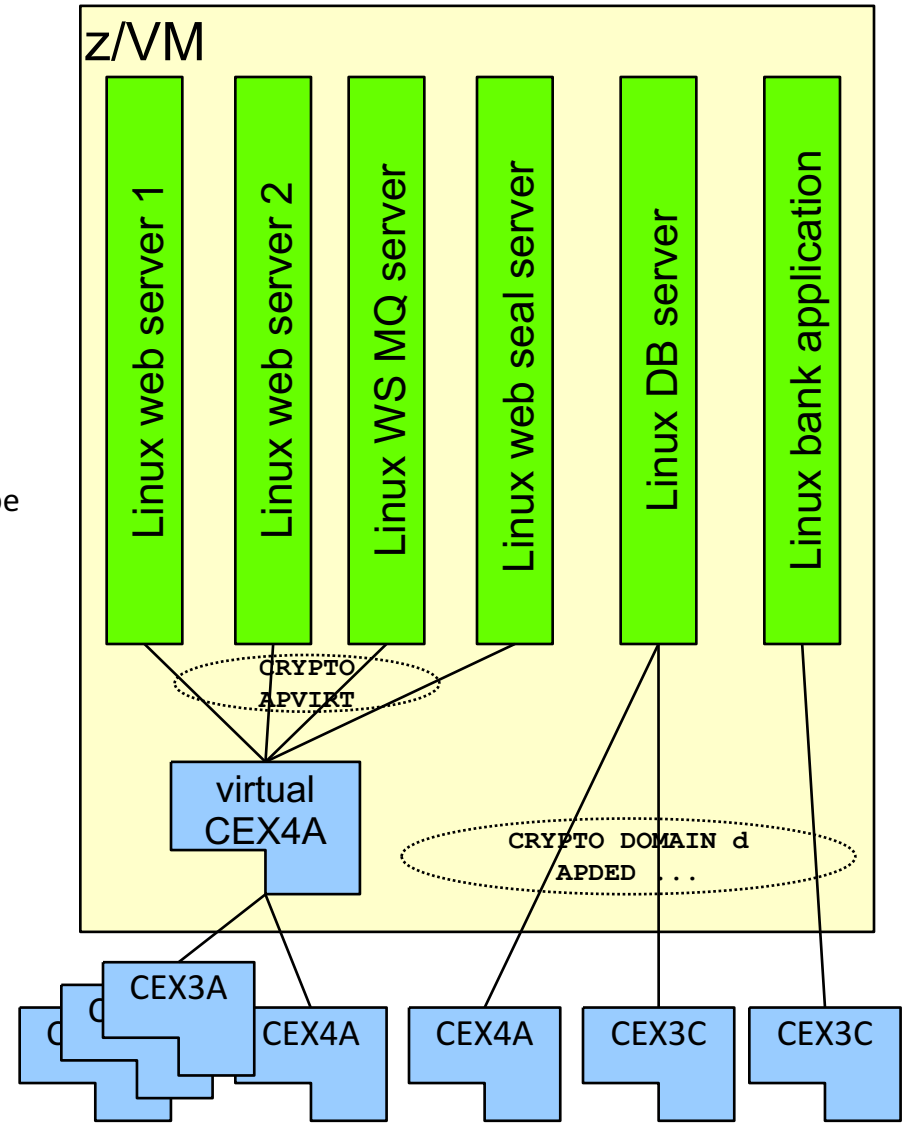
On Features, Adapters, APs, Domains, Queues and such

- There may be up to 16 crypto express adapters (aka APs) per CEC
- CEX2 or CEX3 features have 2 APs
- Each adapter (AP) has
  - an AP Id
  - firmware load (aka mode or type)
    - accel., CCA-coproc. or EP11-coproc.
  - can be divided into multiple domains (HW virtualization)
- AP domain = AP queue
- Configuration constraints
  - LPARs may be granted access to
    - a list (a_1, a_2, ..., a_k) of APs and
    - a list (d_1,d_2, ... d_j) of domains
  - resulting in access to AP queues
    - (a_1d_1, ..., a_1d_j, a_2d_1, ..., a_kd_j)
- The Linux on z device driver
  - only uses one domain on all APs



z/VM Crypto Guest Support

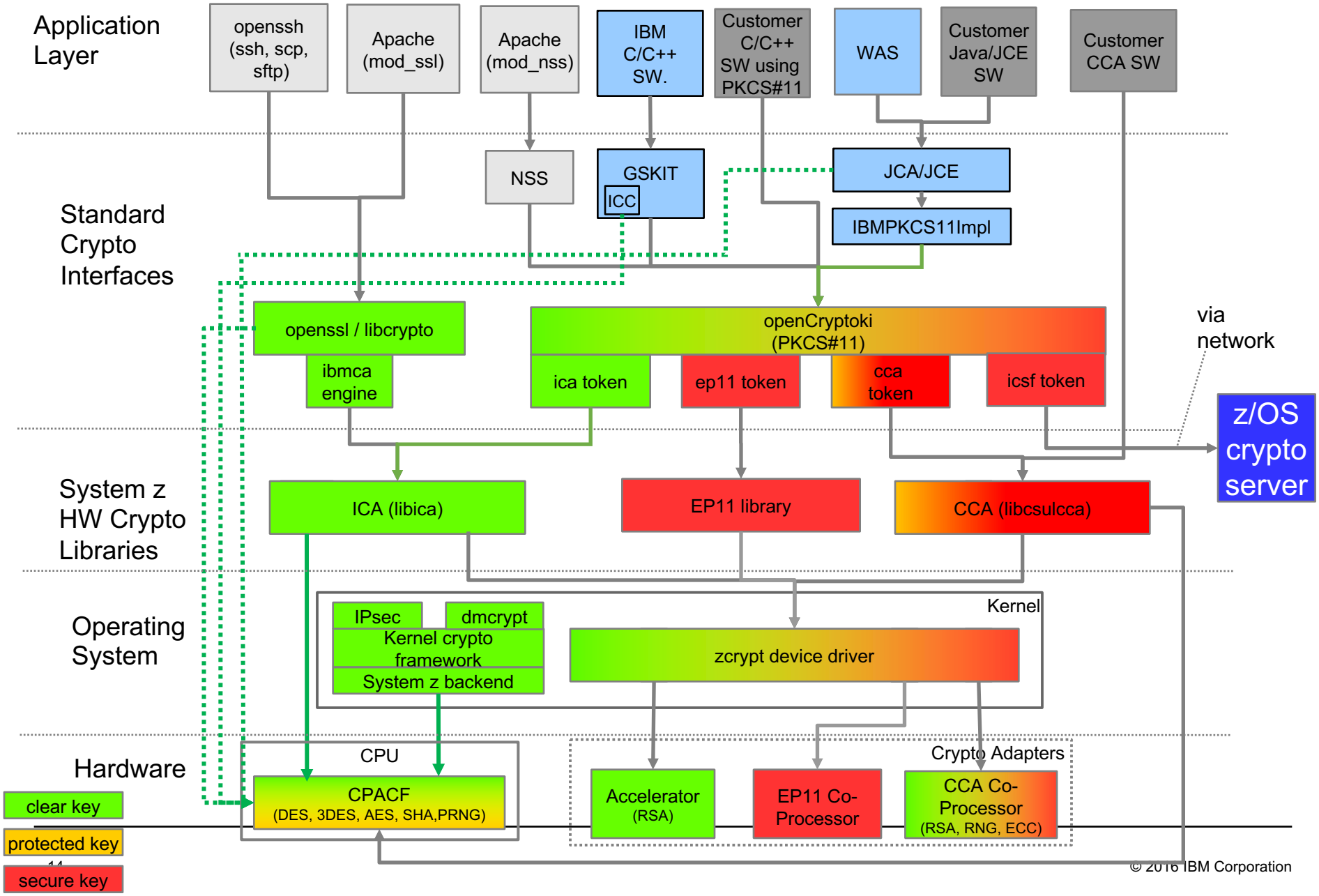
- A z/VM guest may have
  - either dedicated adapters
    - `CRYPTO DOMAIN d APDED a1 a2 ...`
  - or shared adapters
    - `CRYPTO APVIRT`
- Shared adapters
  - Accelerators or CCA-coprocessors only
  - Defined using APVIRT system configuration
    - `CRYPTO APVIRT aps DOMAIN domains`
  - otherwise uses all not dedicated APs of a single “greatest” type
    - CEX5A > CEX4A >CEX3A > CEX2A > CEX5C > CEX4C > CEX3C > CEX2C
  - load balancing, failover, dynamic add/remove
  - Can only be used for clear key operations
- Dedicated adapters
  - meant for secure key adapters
  - no adapter/domain id virtualization
  - only way to mix CCA coprocessors and accelerators in a guest
- Checking Crypto Configuration
  - show status of crypto facilities
    - `Q CRYPTO [ AP|DOMAIN [Users] ]`
  - show status of crypto facilities of guest
    - `Q V CRYPTO`



Linux on z Systems Cryptography Stack

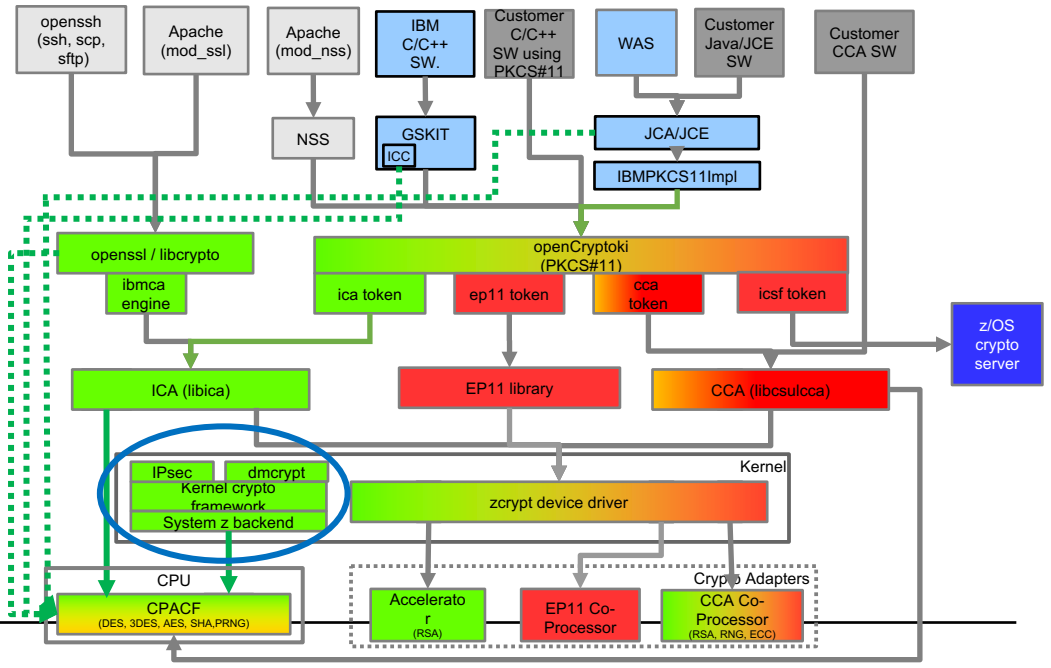
IBM z Systems

Linux on z & LinuxONE Systems Crypto Stack



In-kernel crypto

- The Linux kernel implements a set of cryptographic mechanisms: hashes and symmetric crypto, pseudo random numbers
- Kernel architecture allows to replace software implementation by HW implementations in a modular manner
  - Modules: sha_common, sha1_s390, sha256_s390, sha512_s390, ghash_s390, aes_s390, prng
  - `# modprobe sha1_s390`
  - `# modprobe sha256_s390`
  - …
- If CPACF is installed Linux on System z supports:
  - SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
  - GHASH
  - DES, 3DES: ECB, CBC, CTR
  - AES: ECB, CBC, CTR, XTS
  - PRNG, DRNG
- Exploiters of in-kernel crypto
  - dm-crypt
  - ext4fs crypto
  - IPsec



Using dm-crypt for End-to-End Data Encryption

- **dm-crypt / LUKS**
  - a mechanism for end-to-end data encryption
  - data only appears in the clear when in program
- kernel component that transparently
  - for a whole block device (partition or LV)
    - encrypts all data written to disk
    - decrypts all data read from disk
- How it works:
  - encryption keys stored on disk (partition, LV)
    - containing either swap space or any file system
  - encryption keys on disk are protected by passwords
  - uses in kernel-crypto
    - **can use z System HW if aes_s390 module loaded**
      - AES-CBC
      - XTS-AES (recommended)
- How to use dm-crypt
  - format a partition as encrypted volume -- only required once

```
# cryptsetup luksFormat -c aes-xts-plain64:sha512 -s 512 \
/dev/dasdb1
```

    - password to encrypt (random) key must be supplied
  - open encrypted volume as a device mapper volume

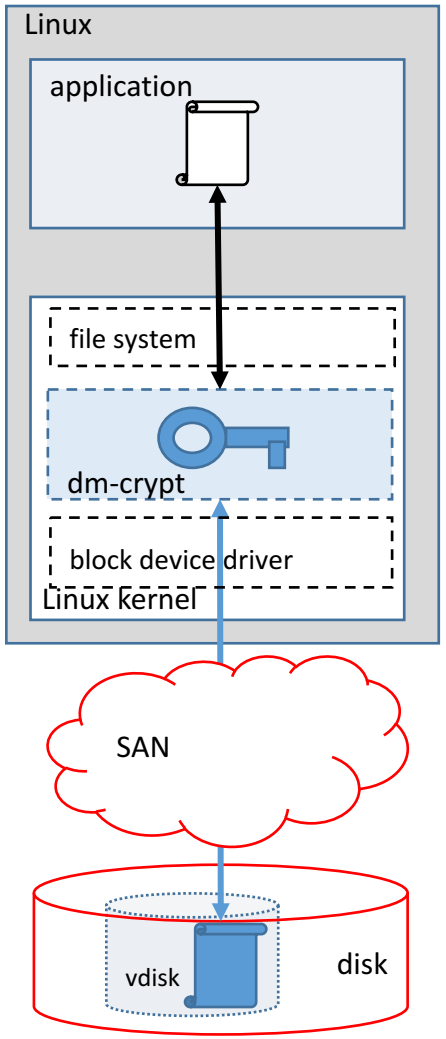
```
# cryptsetup luksOpen /dev/dasdb1 sec_dev
```

    - password must be entered
  - create file system (or swap space) -- only required once

```
# mkfs.ex4f /dev/mapper/sec_dev
```

  - mount (or swapon) device mapper volume

```
# mount /dev/mapper/sec_dev
```



Kernel Support for NIST SP800-90A DRBG

prng kernel module is extended

- now exploits PPNO instruction
- SHA512 based deterministic random number generator
- random bytes can be read from /dev/prandom when prng module is loaded
- prng module can be configured to use old TDES based PRNG instead
  - new module parameters
    - chuncksize: size of internal buffer to generate random numbers
    - mode: 0: best method available, 1: TDES-PRNG, 2: SHA512-DRBG
    - reseedlimit: reseed is triggered after reseedlimit · chuncksize bytes

- available in
  - SLES 12 SP1
  - RHE 7.2
  - Ubuntu 16.04

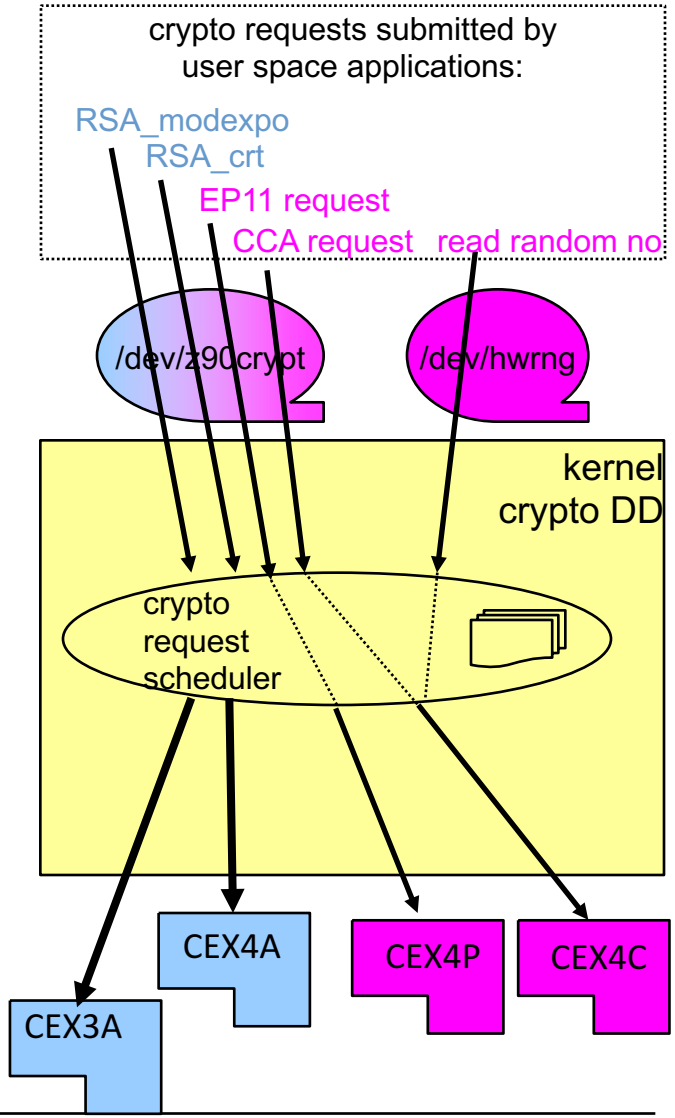
The Linux on z Crypto (Adjunct Processor) Device Driver

the device driver
- ap (zcrypt_cex4, zcrypt_cex2a, zcrypt_pcixcc, zcrypt_pcicc, zcrypt_pcica, zcrypt_masgtype50, zcrypt_magtype6, zcrypt_api)
- fromerly z90crypt

maps all crypto cards to one device (typically `/dev/z90crypt`)
- supports RSA functions (modular exponent & Chinese Remainder Theorem)
- supports CCA functions (for CEX*C, secure key)
- supports EP11 functions (for CEX*P, secure key)
- crypto functions (i.e. IOCTL calls to `/dev/z90crypt`) are
  - routed to a crypto card depending on the card's capability (function, key length supported) and the card performance & load
  - adapter failover in case of failures
  - asynchronous
    - on z/VM and pre z10 LPARs: DD polls for answers
      - enable polling for short latency
        - costs CPU time!
        - `/sys/bus/ap/poll_thread`
        - `/sys/bus/ap/poll_timeout` (ns resolution)
    - on z10 and later LPARs: thin interrupts

provides real random numbers (aka “long random numbers”)
- on `/dev/hwrng` (for CEX*C)

dynamically adding or removing crypto adapters
- when timer `/sys/bus/ap/config_time` expires (LPAR)



Crypto Adapter Administration

lszcrypt shows crypto device attributes

- - V - VV –VVV verbose adapter attribute
- -b show AP bus attributes
- -c show adapte capabilities
- Example

```
[root@R1745030 ~]# lszcrypt -VV
card00: CEX3A     online  hwtype=8  depth=8 request_count=0
card03: CEX3C     online  hwtype=9  depth=8 request_count=3163
[root@R1745030 ~]# lszcrypt -b
ap_domain=4
ap_interrupts are disabled
config_time=30 (seconds)
poll_thread is disabled
poll_timeout=1500000 (nanoseconds)
```

chzcrypt can modify some attributes

- -e / -d enable/disable adapter
- -p / -n enable/disable poll thread
- -t set high resolution polling timer
- -c set timer for reconfiguration scanner
- --help display short help text

See “Device Driver Features and Commands” for details

```
/sys/bus/ap
    cd ap
 ap_control_domain_mask
    ap_domain
    ap_interrupts
    config_time
    devices
        card00 -> ../../../devices/ap/card00
        card03 -> ../../../devices/ap/card03
    drivers
        cex2a
            card00 -> ../../../../devices/ap/card00
            module -> ../../../../module/z90crypt
        pcixcc
            card03 -> ../../../../devices/ap/card03
            module -> ../../../../module/z90crypt
    poll_thread
    poll_timeout
/sys/devices/ap
    card00
        ap_functions
        depth
        driver → ../../../bus/ap/drivers/cex2a
        hwtype
        online
        pendingq_count
        raw_hwtype
        request_count
        requestq_count
        subsystem -> ../../../bus/ap
        type
    card03
        depth
        driver → ../../../bus/ap/drivers/pcixcc
        hwtype
        …
```

Options for lszcrypt

lszcrypt with triple V option (very very verbose):

```
# lszcrypt -VVV
card00: CEX4C       online  hwtype=10 depth=8 request_count=1
pendingq_count=0 requestq_count=0 functions=0x90000000
card01: CEX4C       online  hwtype=10 depth=8 request_count=0
pendingq_count=0 requestq_count=0 functions=0x90000000
card02: CEX4A       online  hwtype=10 depth=8 request_count=0
pendingq_count=0 requestq_count=0 functions=0x68000000
#
```

Capabilities of an adapter:

```
# lszcrypt -c 00
card00 provides capability for:
RSA 4K Clear Key
CCA Secure Key
Long RNG
# lszcrypt -c 02
card02 provides capability for:
RSA 4K Clear Key
#
```

Libica Library

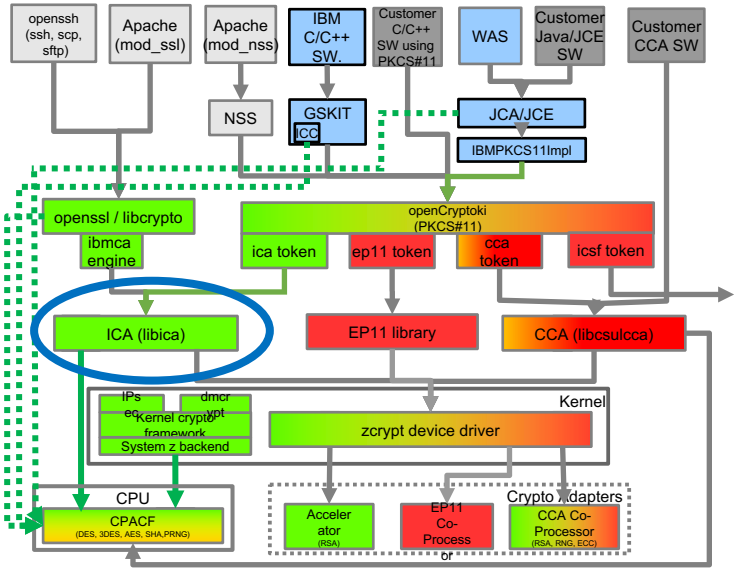
the libica library provides a C API for

- symmetric crypto & hash mechanisms (clear key CPACF support)
  - hash: SHA1, SHA224, SHA256, SHA384, SHA512
  - DES / 3DES: ECB,, CBC, CBC_CS, CFB, OFB, CTR, CBC_MAC, CMAC
  - AES128/192/256: ECB, CBC, CBC_CS, CFB, OFB, CTR, XTS (no 192 key), CBC_MAC, CMAC
  - AEAD: AES128/192/256: CCM, GCM
- clear key RSA upto 4k moduli (CCA-coprocessor & accelerator support)
  - modular exponentiation:
    - ME: encrypt/verify (decrypt/sign)
    - CRT: decrypt/sign
  - key generation
- pseudo random numbers (CCA-coprocessor/CPACF/kernel)
  - SHA512-DRBG

libica book:

libica Programmer's Reference - SC34-2602-xx

https://www.ibm.com/support/knowledgecenter/linuxonibm/com.ibm.linux.z.lxci/lxci_linuxonz.html



Libica – Latest Versions

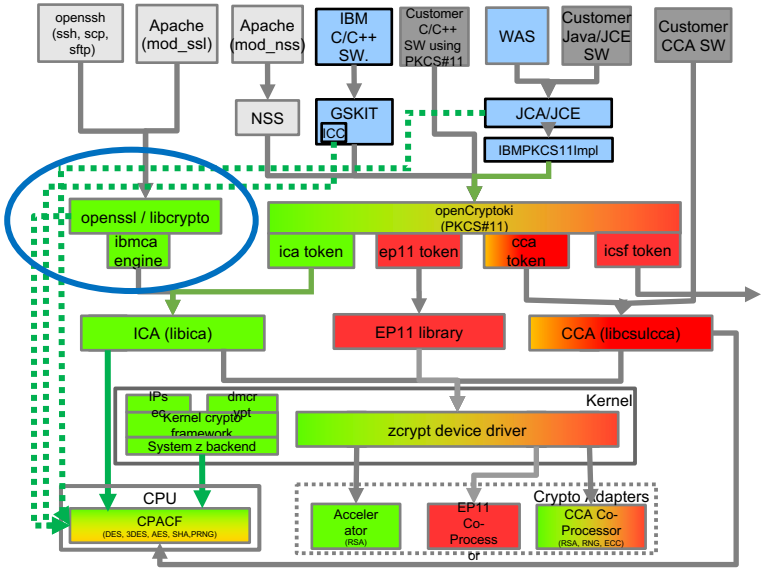
- Libica 2.4
  - new statistics
  - available in SLES 11 SP4 & 12 SP1, RHEL 6.7 & 7.1
- Libica 2.5
  - support GCM in streaming mode
  - status: upstream
- Libica 2.6
  - support NIST SP800-90A SHA512 based DRBG
  - available in Ubuntu 16.04
  - status: upstream
- Upstream Link: https://sourceforge.net/projects/opencryptoki/files/libica/

openSSL / libcrypto

- openSSL implements (SSL and) TLS protocol
- libcrypto is the crypto library of openSSL
  - used by many open source projects
  - e.g. openssh, apache nod_ssl, nodes.js, PHP, postgres, MongoDB EE, Ruby

- version 1.0.x libcrypto has built-in CPACF support
  - CPACF: SHA1, SHA2
  - CPACF: AES: ECB, CBC, CTR, XTS
  - CPACF: GHASH
  - z assembler: long number arithmetic

- the ibmca dynamic engine supports
  - CPACF: SHA1, SHA256
  - CPACF: DES/3DES/AES: ECB, CBC, CFB, OFB
  - CEX*A/CEX*C: RSA, DH, DSA
  - CPACF, CEX*C: pseudo random number generation

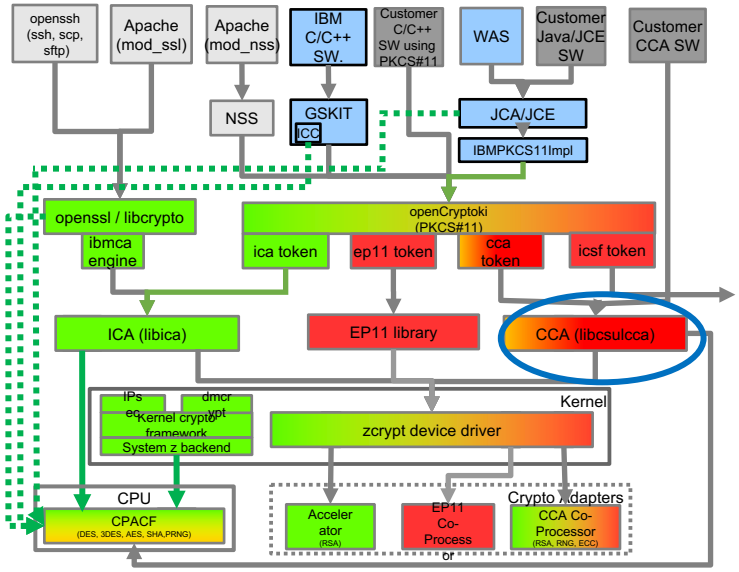


- usage of ibmca engine must be configured in openssl.cnf

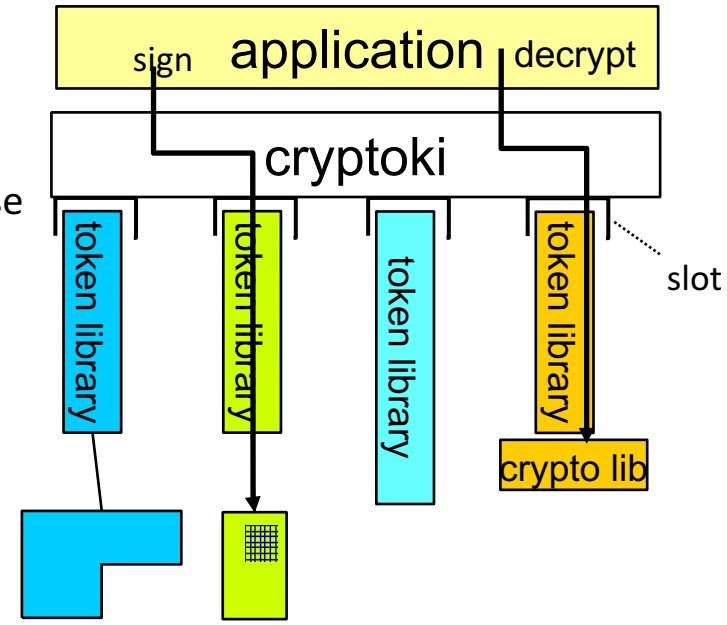
Common Cryptographic Architecture (CCA)

- CCA RPM
  - secure key crypto library libcsulcca (C and Java)
  - supports protected key crypto
  - proxy daemon to connect to TKE (catcher.exe)
  - tools (e.g. panel.exe to set master keys on crypto adapter)
  - available for free from IBM webesite
    - http://www.ibm.com/security/cryptocards/pciecc2/lonzsoftware.shtml
- Latest version: CCA 5.2 RPM
  - z13 & CEX5C:
    - function set of CCA 5.2 firmware
    - up to 85 domains
  - zEC12 & CEX4C:
    - function set of CCA 4.4 firmware
  - TKE 8.0 support
  - Support for SLES & RHEL releases
- documentation
  - http://www.ibm.com/security/cryptocards/pciecc2/library.shtml

*) Note: Please contact Visa for details on licensing the use of this technology.



PKCS #11 - openCryptoki



- PKCS#11 is a popular crypto standard describing an API to use cryptographic methods

- Allows to write software that uses cryptographic algorithms
  - that may be implemented in HW "tokens"
    - HSMs (crypto adapters or *smart cards*)
    - Crypto accelerators
  - largely independent of crypto HW
  - allowing for multiple tokens
  - allowing for different tokens

- Allows to configure existing applications to exploit cryptographic HW
  - applications configurable plug-in mechanisms for PKCS#11 libraries
    - WAS, IHS, Apache/mod_nss, ...
  - crypto modules with plug-in options for PKCS#11 libraries
    - Java JCA/JCE
    - GSKIT (IBM crypto library used for SWG products)

openCryptoki: is an open source implementation of PKCS #11

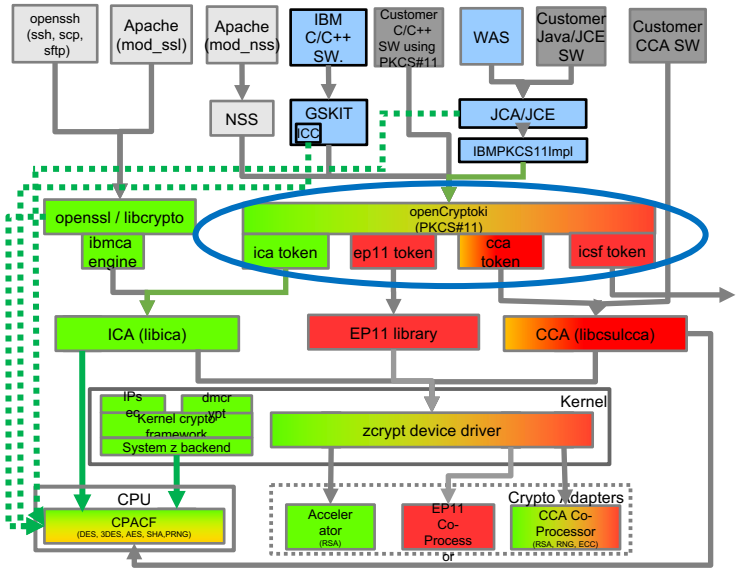
- https://sourceforge.net/projects/opencryptoki/files/opencryptoki/

Current opencryptoki release availability

upstream :	3.6
RHEL 6.7:	3.2
RHEL 7.1:	3.2
SLES 11 SP4:	3.2
SLES 12 SP1:	3.2
Ubuntu 16.4:	3.4.1

Opencryptoki Tokens for Linux on z & LinuxONE

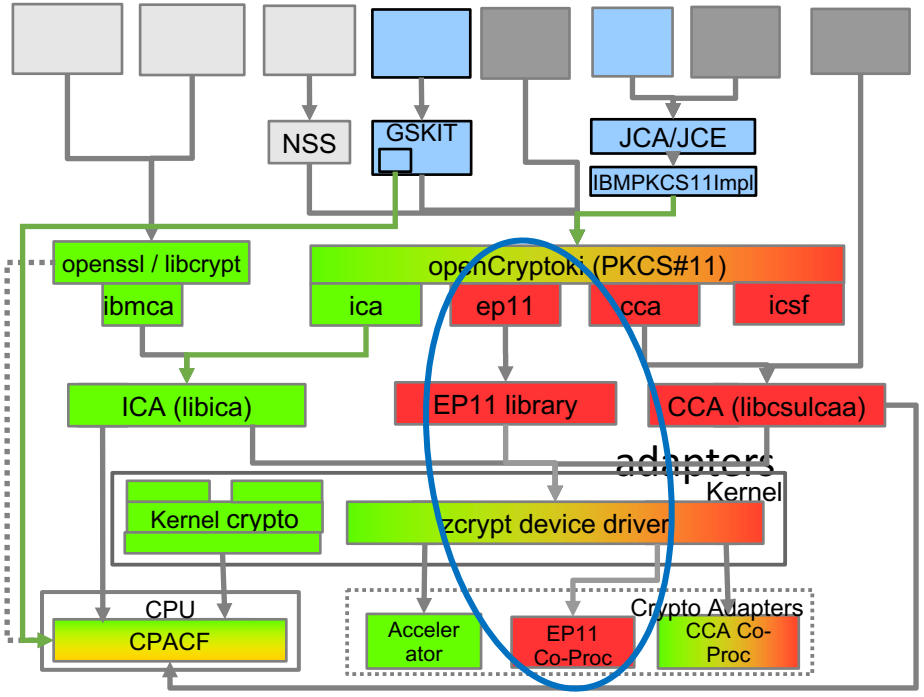
- ica token
  - provides clear key cryptographic functions
  - uses libica
  - exploits CPACF, CryptoExpress accelerators and CCA co-processors
  - System z specific
- cca token
  - provides secure key cryptographic functions
  - uses CCA library (libcsulcca)
  - exploits CryptoExpress CCA co-processors
  - System z specific
- soft token
  - provides clear key cryptographic functions
  - pure software implementation, relies on libcrypto (openSSL)
  - platform independent
- ep11 token (since openCryptoki 3.1)
  - provides secure key cryptographic functions
  - exploits CryptoExpress EP11 co-processors
  - System z specific



- icsf token (since openCryptoki 3.0)
  - remote access to cryptographic functions on a z/OS based ICFS crypto server
  - uses LDAP protocol
  - platform independent

EP11 Support

- CEX4S or CEX5S with EP11 firmware: CEX4P or CEX5P
  - secure key cryptography
- EP11 host library (RPM and Debian package)
  - Available for free via IBM web site: http://www.ibm.com/security/cryptocards/pciecc2/lonzsoftware.shtml
  - support for SLES 11&12 and RHEL 6&7
  - support for Ubuntu16.04
- openCryptoki ep11 token
  - since openCryptoki 3.1
  - rich crypto API:
    - 3DES, AES128/192/256 using ECB, CBC
    - RSA (PKCS 1.5, PSS) with1024-4096 bit keys
    - DH, DSA with 1024-3072 bit keys
    - ECDSA, ECDH with 192-521 bit keys
  - Crypto request can be sprayed to multiple
    - Load balancing
    - Fail over



- Documentation:

  www.ibm.com/support/knowledgecenter/linuxonibm/com.ibm.linux.z.lxce/lxce_usingep11.html

Master Key Management with EP11 Token

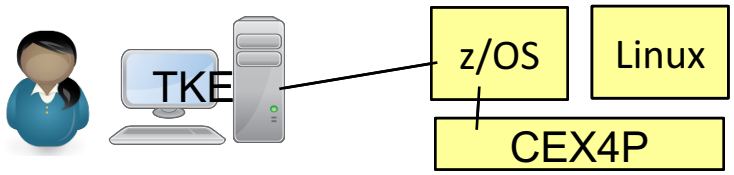
A TKE is needed to set and manage EP11 master keys.

- **Option 1:**

  administration via TKE

  TKE connects to z/OS

  Linux domains belong to z/OS control domains

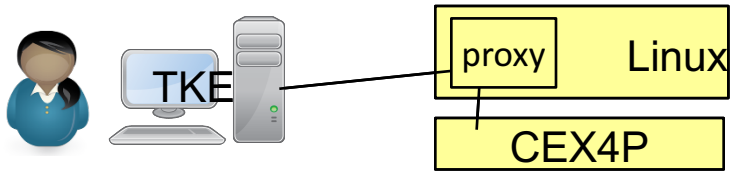


- **Option 2:**

  administration via TKE

  TKE connects to Linux proxy daemon `ep11TKEd`

  Linux must be assigned control domains



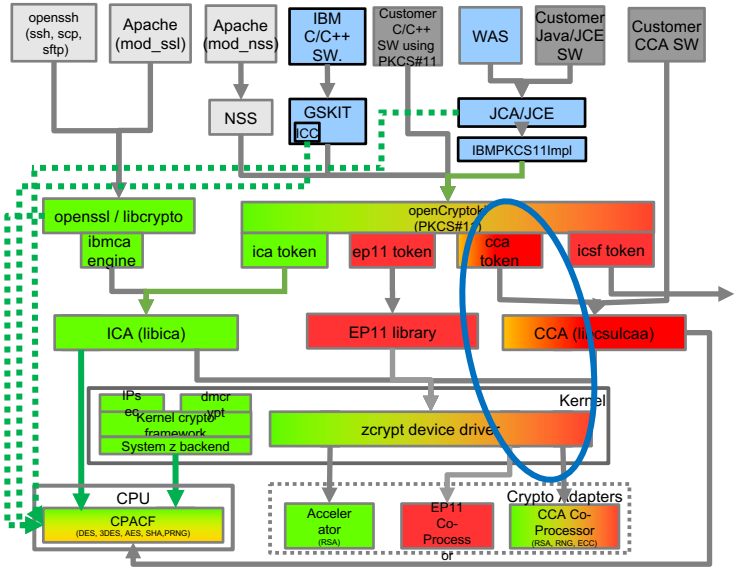
Master Key Change Process

- If master key is changed on the EP11 adapter(s) all key objects in the token object repository of the ep11 token become invalid.
- openCryptoki version 3.1 comes with a key migration tool: `pkcsep11_migrate`
- Master key change requires management process
  1) on TKE: submit and commit a new master key on EP11 adapter
  2) on Linux: stop all processes using openCryptoki with EP11 token
  3) on Linux: back up token object repository of EP11 token
  4) on Linux: migrate keys of object repository of EP11 token with migration tool
  5) on TKE: activate new master key on EP11 adapter
  6) on Linux: restart applications using openCryptoki with EP11 token

The openCryptoki CCA Token

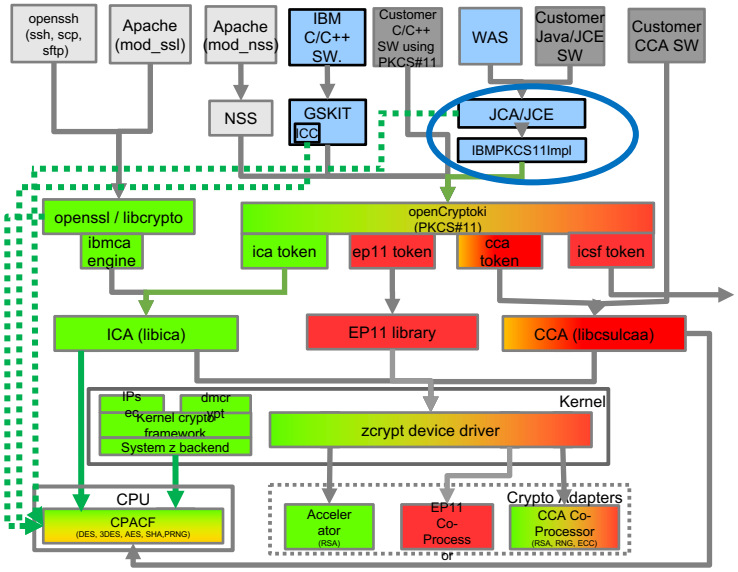
- supports secure key crypto only
  - can use CPACF protected keys
- mechanisms:
  - DES/3DES/AES: ECB, CBC
  - SHA1, SHA256
  - RSA, ECDSA
- note, format of token keys changed between openCryptoki 2.x and 3.x
  - use key migration tool pkcscca to transform keys
- CCA master key change:
  - use key migration tool pkcscca to re-encipher token keys in openCryptoki repository (since version 3.4)
- Documentation: reference information part of CCA Book:

  https://www.ibm.com/support/knowledgecenter/linuxonibm/com.ibm.linux.z.wskc.doc/wskc_c_oc_opencryptoki.html



z Systems & LinuxONE Crypto HW support for Java

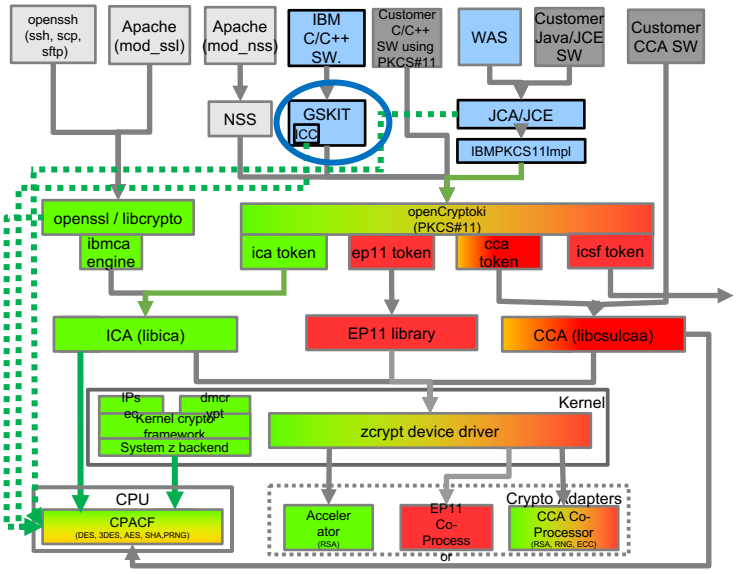
- The IBM Java 8 Java Cryptography Extension (JCE) on z Systems
  - uses CPACF instructions to accelerate
    - DES, 3DES, AES
      - with ECB, CBC. OFB, CFB and CFB x modes of operation
    - SHA1 and SHA2
  - uses z Systems specific code to accelerate
    - ECDHE – NIST P256 and ECDHE-ECDSA



- The IBMPKCS11Impl provider, tested on
  - recent Crypto adapters,
  - recent Linux distributions
  - and the ICA and CCA tokens
  - for details see: http://www-01.ibm.com/support/docview.wss?uid=swg21967855
  - support for EP11 and ICSF tokens in currently progress

GSKit the IBM C/C++ Crypto Library

- not available as stand-alone library but
- GSKit part of many IBM Products
  - e.g DB2, Webshere MQ, TSM
- ICC component
  - uses CPACF functions for
    - AES: ECB, CBC, CFB, OFB, XTS, GCM CCM
    - SHA1
    - SHA2
    - GHASH
- GSKit can be configured to crypto functions from a PKCS #11 API
  - i.e. it can link to openCryptoki



Crypto Statistics

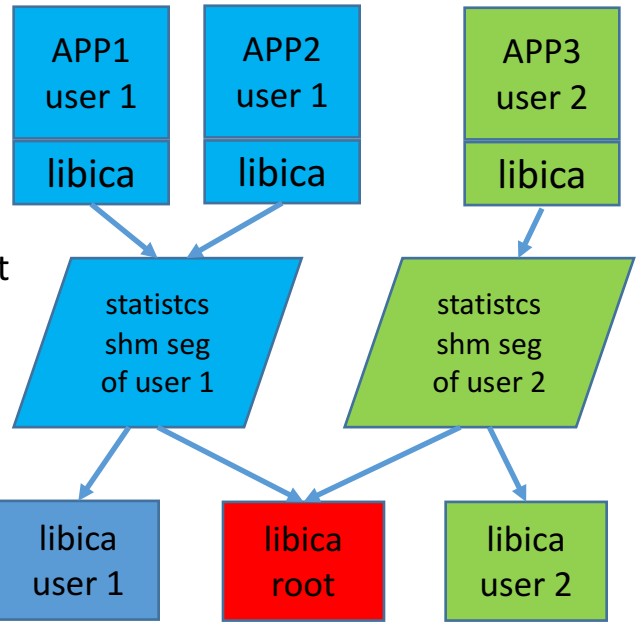
- new icastats for libica
  - as of libica 2.4
    - available with SLES 12.1 and RHEL 7.1
  - user specific data collection (root can see all data)
  - stores statitics data after application is finieshed until explicit reset
  - options for non-privileged users:

```
icastats [--delete|--reset|--help|--version ]
```

  - additional options for root:

```
icastats [--all|--delete-all|--reset-all|--summary
|--user <username>]
```

    - „all“ option: get statistics for all users,
    - summary: cumulative statisitics of all users



- CPACF statistics with cpacfstats
  - available Ubuntu 16.04 and in s390tools 1.29 (developper works)
  - counts AES, DES, SHA and PRNG operations
  - can count CPACF usage of SW not using libica (e.g. kernel, GSKit)
  - works for Linux running in an LPARs
  - LPAR must be authorized to use CPU counters
    - “Crypto activity counter set authorization control" checkbox
  - Example
    - start cpacfstatd daemon:

```
# cpacfstatd
```

    - enable aes counter

```
# cpfafstat -e aes
```

    - read aes counter

```
# cpacfsts -p aes
aes counter: 144
```

- statistics of Crypto Express requests see lszcrypt –VV or lszcrypt –VVV

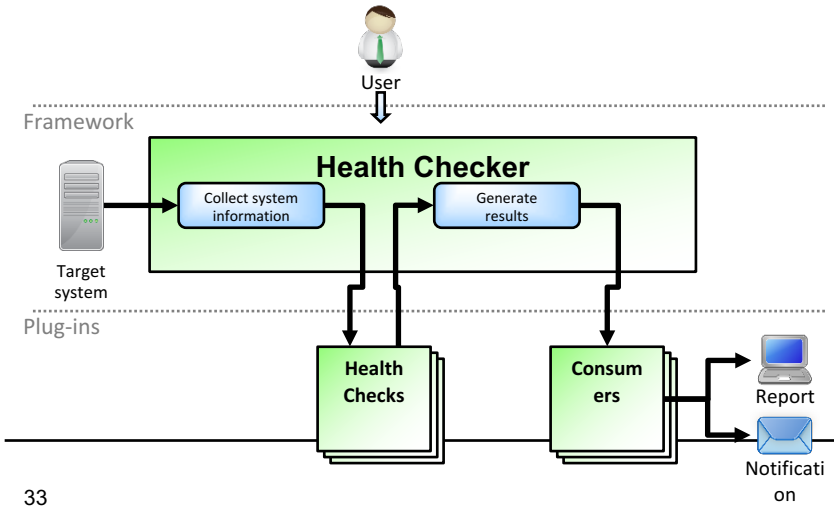
```
[root@s3560007 ~]# lszcrypt -VV
card00: CEX5A … request_count=2
card01: CEX5C … request_count=51
card05: CEX5P … request_count=1092
```

LNXHC -- Crypto Health Checks

Linux Health Checker (LNXHC)

- http://lnxhc.sourceforge.net/
- a framework & tool to check whether the set up of a system is correct or follows best practices
- extensible wrt the set of checks
- adaptable profiles to match set of *applicable* checks to customer environment
- provides
  - indications of problems found
  - explanation of the problems
  - hints to resolve problems

```
linux:~ # lnxhc run
Collecting system information
Running checks (50 checks)
CHECK NAME                                    HOST                    RESULT
================================================================================
boot_runlevel_recommended ..... linux                 SUCCESS
cpu_capacity ..... linux                 SUCCESS
css_ccw_blacklist ..... linux                 SUCCESS
css_ccw_chpid_status ..... linux                 EXCEPTION-LOW

 >EXCEPTION css_ccw_chpid_status.unused_cfg_off(low)
   One or more CHPIDs are in the "standby" configuration state (34-37,
   3f-43, 47, 4e, ...)

css_ccw_device_availability ..... linux                 SUCCESS
css_ccw_device_usage ..... linux                 EXCEPTION-LOW

 >EXCEPTION css_ccw_device_usage.many_unused_devices(low)
   Of 7816 I/O devices, 7806 (99.87%) are unused

css_ccw_driver_association ..... linux                 EXCEPTION-MED

 >EXCEPTION css_ccw_driver_association.no_driver(medium)
   One or more I/O devices are not associated with a device driver:
   0.0.b47f, 0.0.f5fe, 0.0.f7fe
```



New crypto health checks to validate crypto configuration:

- crypto_cca_stack
- crypto_cpacf
- crypto_opencryptoki_ckc
- crypto_opencryptoki_ckc_32bit
- crypto_opencryptoki_skc
- crypto_opencryptoki_skc_32bit
- crypto_openssl_ibmca_config
- crypto_openssl_stack
- crypto_openssl_stack_32bit
- crypto_z_module_loaded

HW Crypto Support for KVM for IBM z 1.1.1

Host support

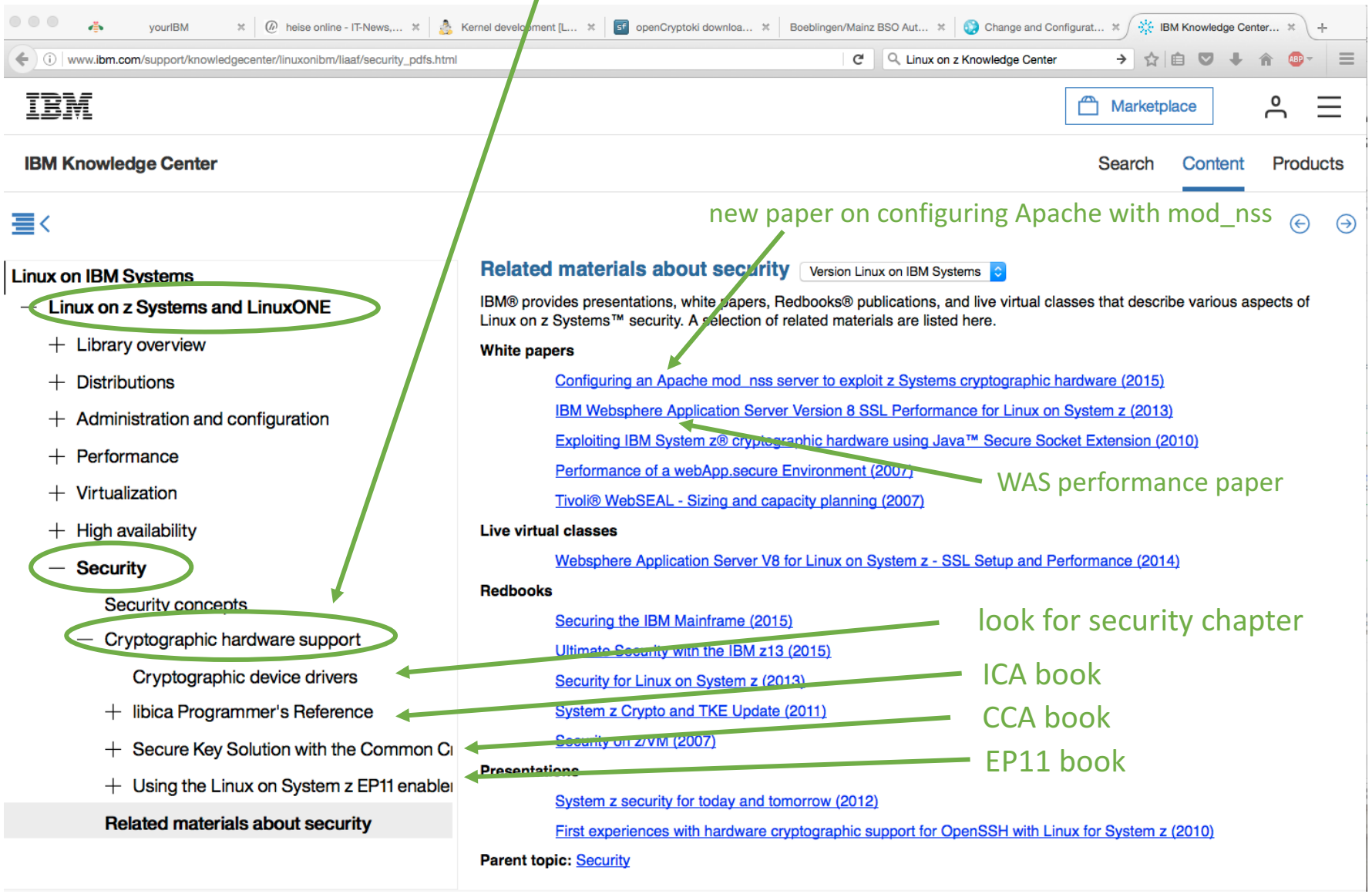
- CAPCF support
  - in Kernel (dm-crypt, IPSec)
    - sha1, sha2 support part of kernel
    - load `aes_s390.ko`, `ghash_s390.ko`, `des_s390.ko`, `prng.ko` mouldes
      - e.g. `modprobe aes_s390`
  - openSSL / libcrypto (with ibmca/libica)
  - NSS with openCryptoki
  - Java 8 with JCE (or IBMPKCS11Impl/openCryptoki)
- Crypto Express Support (clear key)
  - Accelerator (CEX4A, CEX5A) or CCA-coprocessor (CEX4C, CEX5C)
    - load `ap.ko` module
      - `modprobe ap`
  - openSSL / libcrypto with ibmca/libica
  - NSS with openCryptoki
  - Java 8 with IBMPKCS11Impl/openCryptoki
  - CEX4C and CEX5C: support for high quality hardware random numbers

Guest support

- CPACF support
  - in Kernel (dm-crypt, IPSec)
  - openSSL / libcrypto (with ibmca/libica)
  - NSS with openCryptoki
  - Java 6/7 with IBMPKCS11Impl/openCryptoki
  - Java 8 with JCE (or IBMPKCS11Impl/openCryptoki)
- virtual random number generator
  - **recommendation**:
    - only define guests with a virtual random number generator, if the host as access to a HW random number generator (CEX4C or CEX5C).

Linux on z & LinuxONE Security Section in IBM Knowledge Center

https://www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/sec_hw_supp.html



Further Documentation

- Good Overview on Security for Linux on z and z/VM
  - „Security for Linux on System z“, by Lydia Parziale et. al SG24-7728-01 (2013): https://www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/sec_rb_security.html
- How to configure an open source application using PKCS #11:
  - „Configuring an Apache mod_nss server to exploit IBM cryptographic hardware“ by Patric Steuer et al : https://www.ibm.com/support/knowledgecenter/linuxonibm/liaag/l0wnsf00_2015.htm?cp=linuxonibm%2F3-6-2-1
- PKCS #11 with Linux on z Overview article
  - "Using Linux on System z Hardware Cryptography With the PKCS#11 Cryptography Stack" by Reinhard Buendgen in Enterprise Tech Journal on October 6, 2014: http://enterprisesystemsmedia.com/article/using-linux-on-system-z-hardware-cryptography-with-the-pkcs11-cryoptography#sr=g&m=o&cp=or&ct=-tmc&st=%28opu%20qspwjefe%29&ts=1391532094
- Java Crypto Overview for Linux on z
  - "Using Crypto Hardware With Java in Linux on System z" by Reinhard Buendgen, Peter Spera in Enterprise Tech Journal on March 20, 2013: http://enterprisesystemsmedia.com/article/using-crypto-hardware-with-java-in-linux-on-system-z#sr=g&m=o&cp=or&ct=-tmc&st=%28opu%20qspwjefe%29&ts=1391532094

Summary

Crypto support for Linux on System z helps you to

- Leverage unique zSystems cryptographic hardware features
  - CPACF
  - Crypto Express adapters
- Off-load valuable CPU cycles and accelerate workloads that use cryptographic operations
  - in-kernel drivers, for example, IPsec, dm-crypt
  - applications that use OpenSSL, PKCS#11, GSKIT/ICC, Java
  - considerable end-to-end performance gains
- Heighten security by protecting cryptographic keys using a HSM

Questions?

Dr. Reinhard Buendgen

Linux on System z
Architect for Crypto & RAS

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

Phone +49 7031 16-1130
buendgen@de.ibm.com

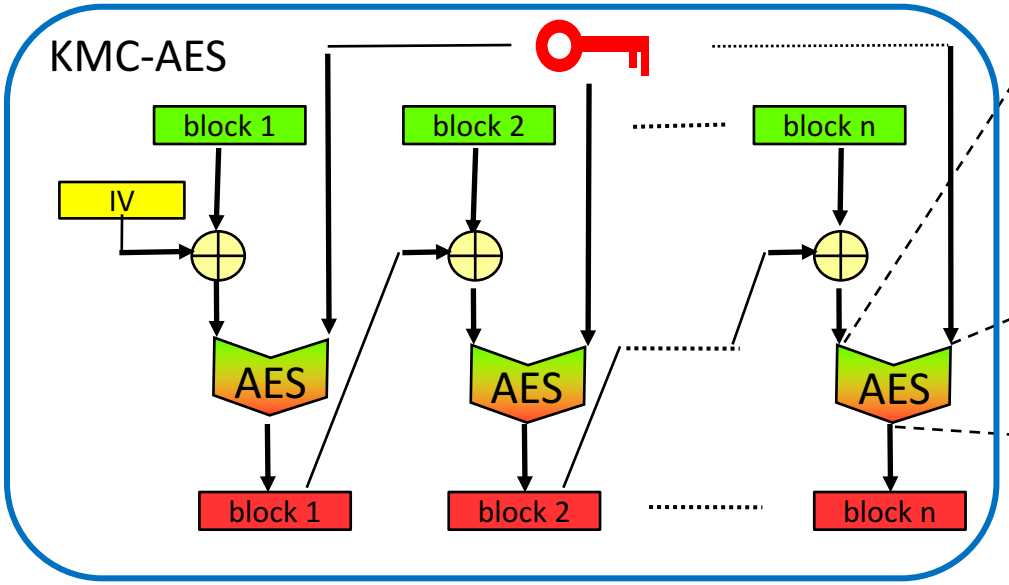
Backup

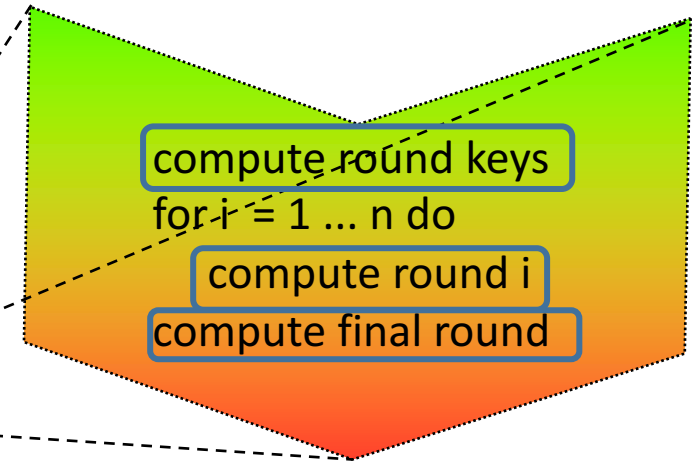
CPACF Instructions (as of z13)

described in z/Architecture Principles of Operation (aka „POP“), Chapter 7 (and 10)
http://www-01.ibm.com/support/docview.wss?uid=isg2b9de5f05a9d57819852571c500428f9a
as Message Security Assist (MSA)

- KM
  - DES/3DES/AES-ECB
  - XTS-AES
- KMC
  - DES/3DES/AES-CBC
  - ANSI X9.17 based PRNG
- KMF
  - DES/3DES/AES-CFB
- KMCTR
  - DES/3DES/AES-CTR
- KMO
  - DES/3DES/AES-OFB
- KIMD, KLMD
  - SHA1
  - SHA2
  - GHASH (GCM)
- KMAC
  - DES/3DES/AES-CBC-MAC
- PCC
  - with KMAC: DES/3DES/AES-CMAC
  - with KM: XTS-AES
- PPNO
  - NIST-SP800-90A based PRNG
- PCKMO (privileged)
  - generate protected key from clear key,

Comparing Intel AES-NI and z Systems CPACF





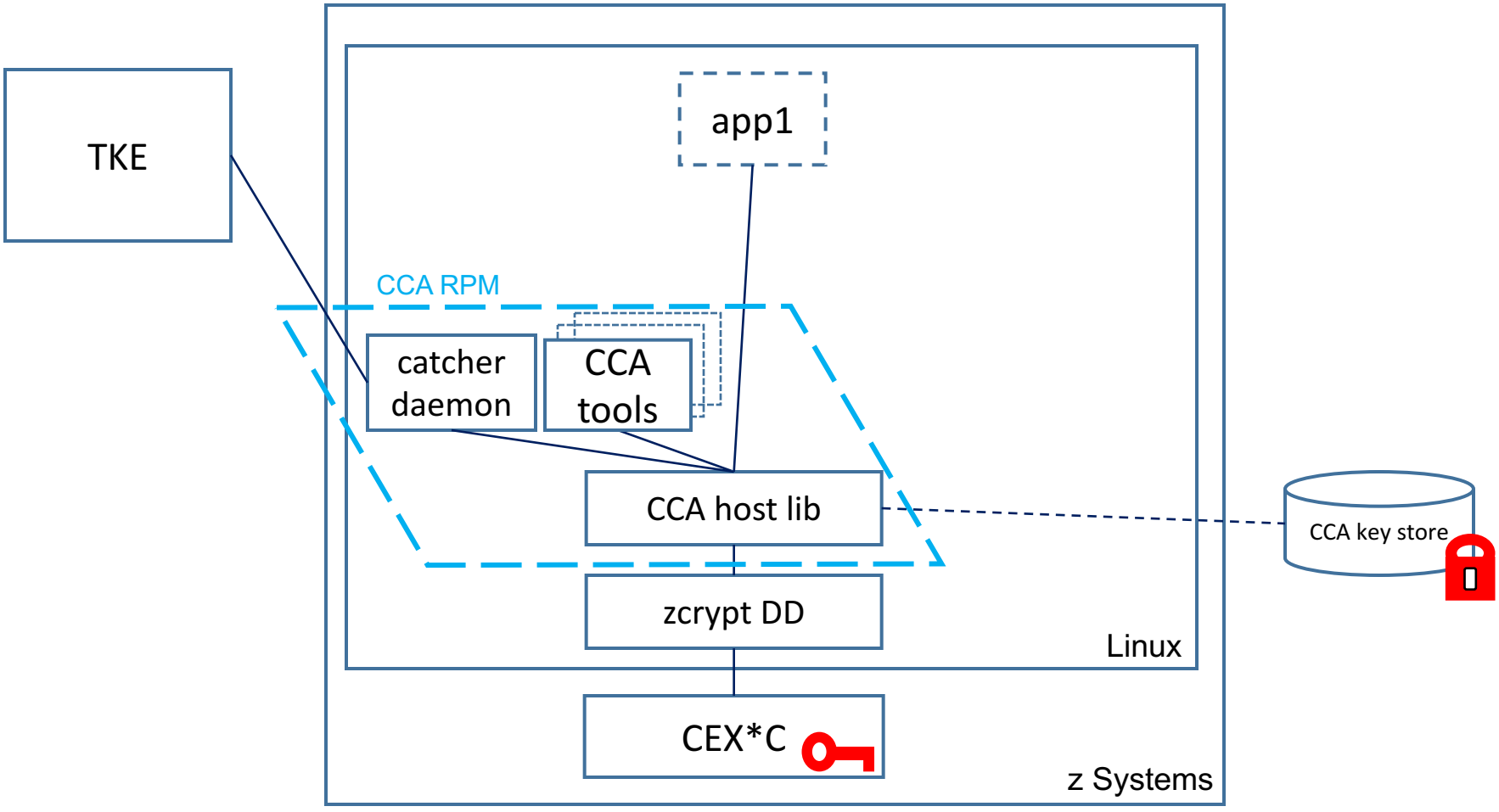
z Systems CPACF

- a set of KM* Instructions
- each operates on a buffer (of arbitrary length) wrt an AES mode of operation
- e.g. KMC computes AES-CBC for a buffer of an arbitrary multiple of 128 bit length

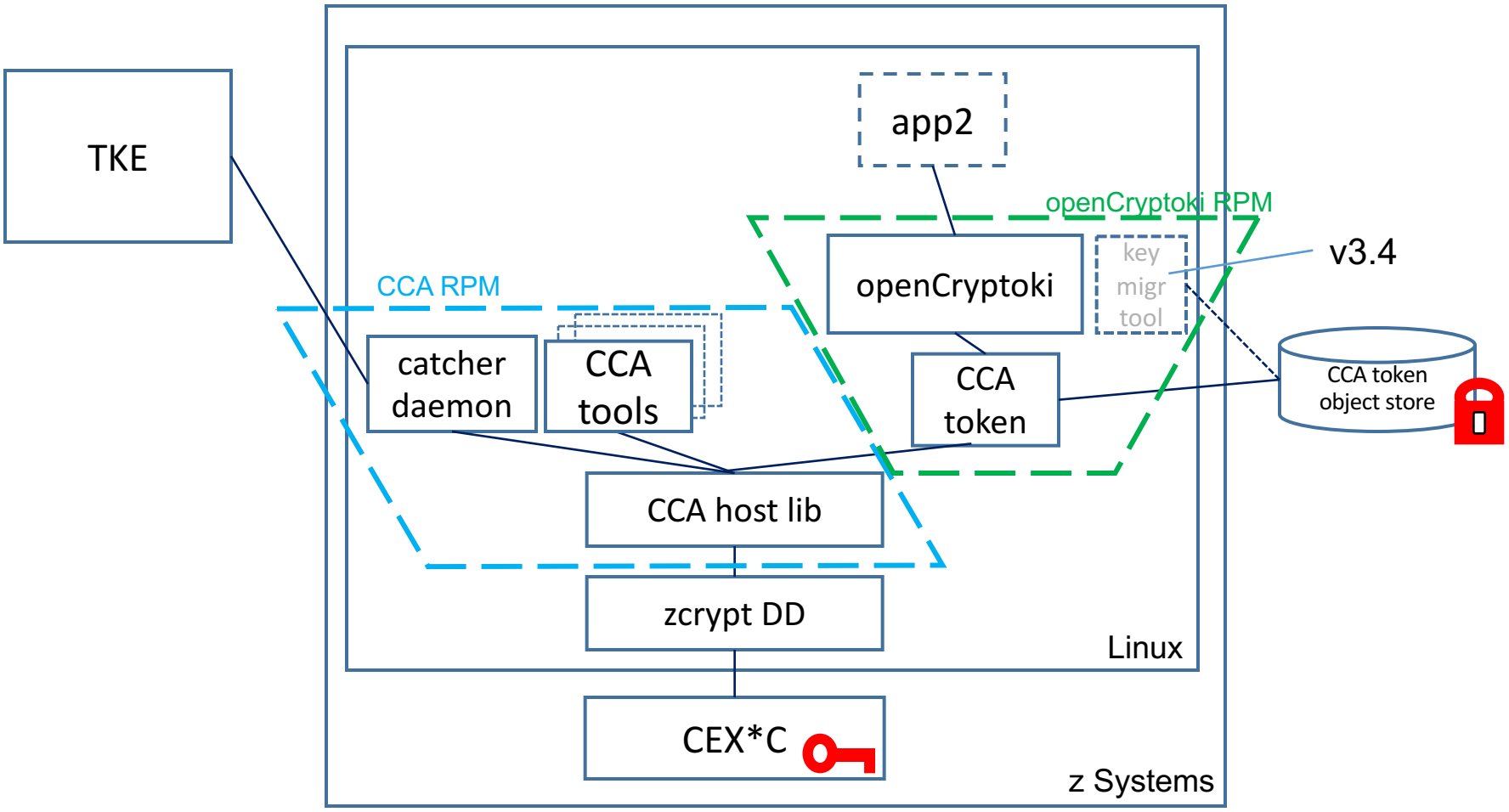
Intel AES-NI

- a set of instructions to
  - compute rounds
  - compute round key
  - assist in round key computation
  - assist in inverse mix columns
- on a single 128 bit block

The CCA Ecosystem -- basic



The CCA Ecosystem with PKCS #11 Interface



Latest openCryptoki Versions

- Version 3.1
  - ep11 token
- Version 3.2
  - generic support for RSA OEAP, and RSA PSS
  - ica token: RSA_OEAP, SHA{1,256,384,512}_RSA_PSS
  - soft token: RSA_OEAP, SHA1_RSA_PSS
- Version 3.3
  - dynamic logging / tracing support
- Version 3.4
  - ica token; GCM support
  - cca token: tool to support master key migration
- Version 3.5
  - bugfixes
- Current openCryptoki release availability
  - upstream (sourceforge): 3.6
  - RHEL 6.7: 3.2
  - RHEL 7.1: 3.2
  - SLES 11 SP4: 3.2
  - SLES 12 SP1: 3.2
  - Ubuntu 16.4: 3.4.1

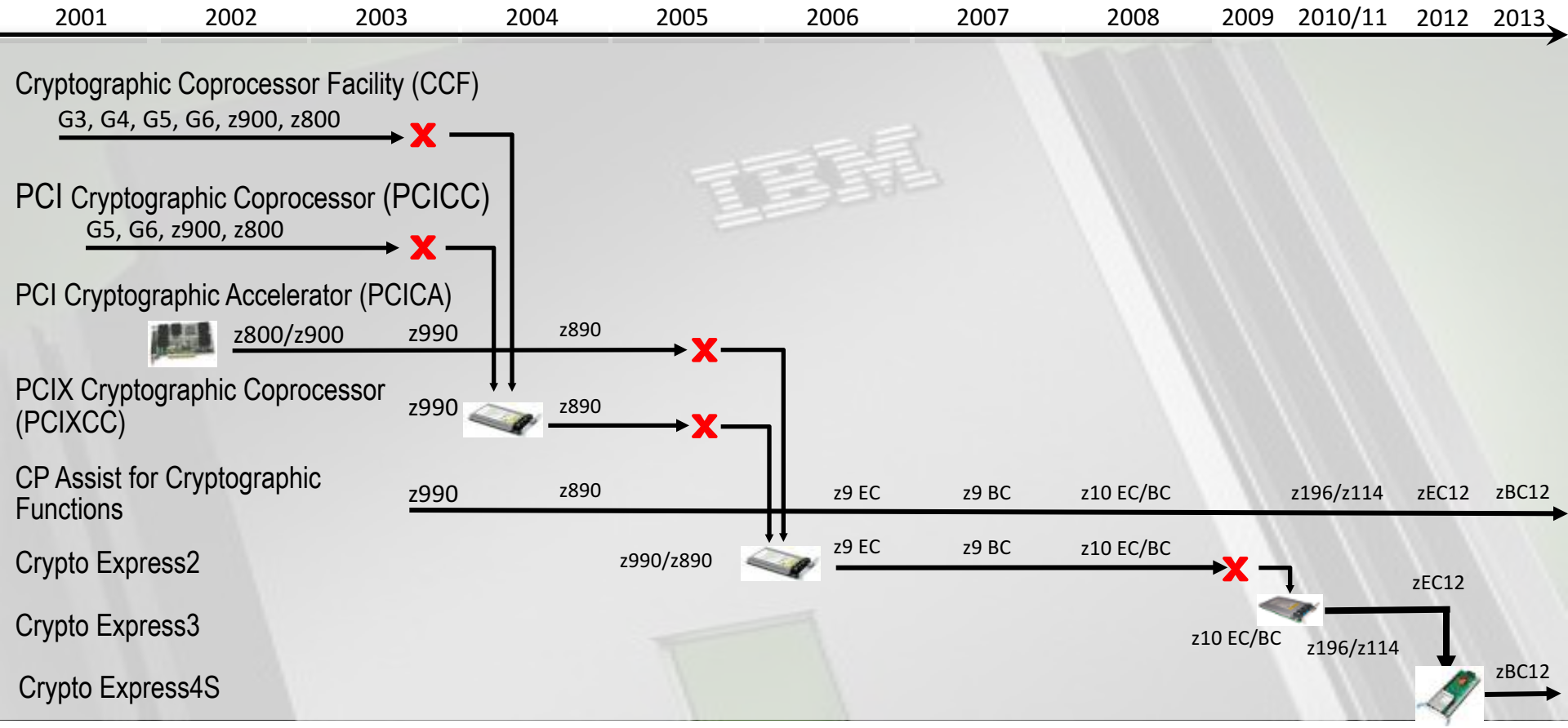
CPACF History

z Systems Generation	MSA	Instructions introduced / *extended*	Functions introduced
z990		KM, KM, KMAC	SHA1, DES/TDES: ECB, CBC, CBC-MAC
z9	ext 1	*KM*, *KMC*, KIMD, KLMD	SHA256, AES-128: ECB,CBC, PRNG
z10	ext 2	*KM, KMC, KIMD, KLMD*	SHA512, AES192, AES256: ECB, CBC
z10	ext3	*KM, KMC, KMAC, KIMD, KLMD*, PCKMO	protected key support for DES/TDES, AES128/192/256
z192/z114	ext 4	*KM, KMC*, KMCTR, KMF, KMO, *KMAC*, *KIMD*, PCC	GHASH, DES/TDES, AES128/192/256: CFB,CTR, OFB, CMAC, AES128/192/256:CBC-MAC, AES128/256:XTS
EC12	ext 5	PPNO	SHA512-PRNG

Crypto Express History

Crypto Card	supporte z Systems Generation	New functiosn
CEX2	z990?/z890?, z9, z10	
CEX3	z10,z192/z114, zEC12	CEX3A: RSA 4k moduli
CEX4	zEC12	CEX4C: ECC
CEX5	z13/z13s	CEX5P

z Systems Crypto History



- Cryptographic Coprocessor Facility – Supports "Secure key" cryptographic processing
- PCICC Feature – Supports "Secure key" cryptographic processing
- PCICA Feature – Supports "Clear key" SSL acceleration
- PCIXCC Feature – Supports "Secure key" cryptographic processing
- CP Assist for Cryptographic Function allows limited "Clear key" crypto functions from any CP/IFL
  - NOT equivalent to CCF on older machines in function or Crypto Express2 capability
- Crypto Express2 – Combines function and performance of PCICA and PCICC
- Crypto Express3 – PCIe Interface, additional processing capacity with improved RAS
- Crypto Express4S - IBM Standard PKCS #EP11

Hardware preceding CCF includes:

- IBM 3845 Channel Attached DES (1977)
- IBM 3848 channel-attached TDES (1979)