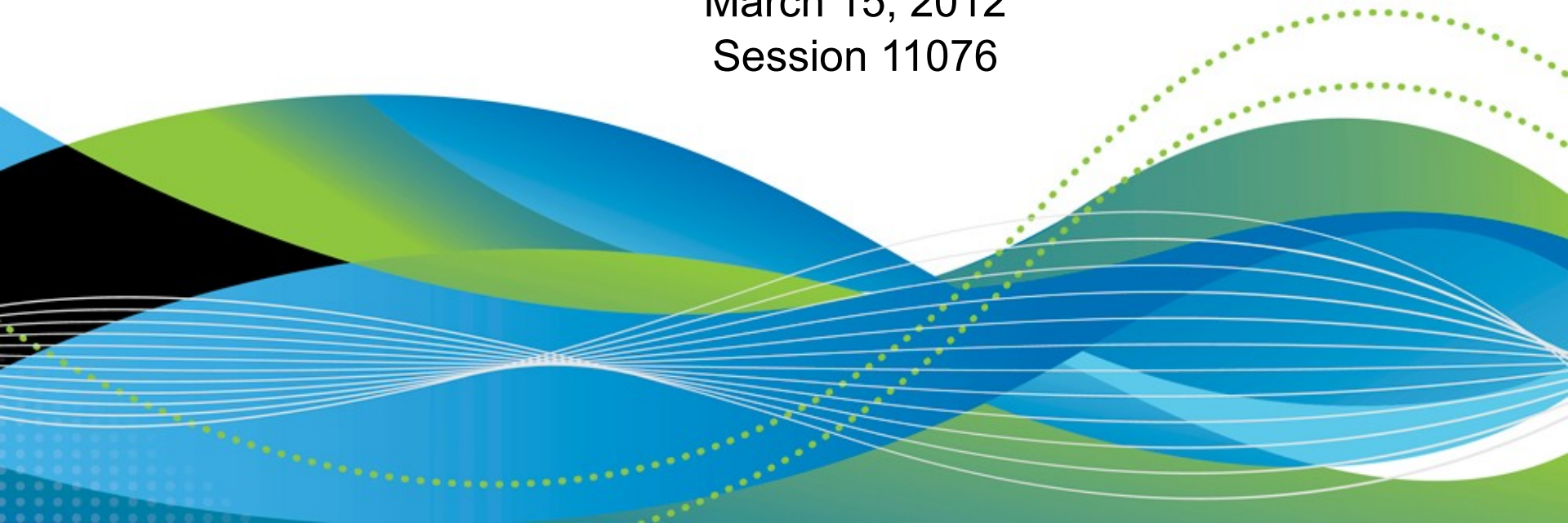
#SHAREorg

SHARE
Technology · Connections · Results

Using Hardware Crypto Support in Linux on System z

Dr. Reinhard Buendgen (buendgen@de.ibm.com)
IBM Deutschland Research & Development

March 15, 2012
Session 11076



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.

Agenda

- Motivation
- Some cryptography basics
- Crypto HW support on System z
- The Linux on System z SW crypto stack

What is Cryptography Good For ?

- Protect data from unauthorized or even hostile access
  - to
    - protect secrets, privacy
    - avoid fraud: stealing or tampering data
    - establish trust among among communication partners
  - during
    - confidential communication (https, ssl/tls, IPsec, vpn)
    - storage of confidential data (encrypted disks, tapes, files, backups!)
    - secure financial transactions (ATMs, POS, e-business, e-banking, online shopping)
    - e-business, e-contracts

- Cryptography provides a tool box to achieve the following goals in IT security
  - confidentiality of data (encryption)
  - data integrity (digests, MACs)
  - authentication of partners (signatures, certificates)
  - authorization of data usage (passwords, management of encryption keys)

Why Crypto HW Acceleration?

- 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
  - Banks: secure key/CCA
- Functionality
  - special build-in security functions for banking and financial applications:secure key/cca
- Regulations
  - FIPS 140-2 certified cryptographic adapters

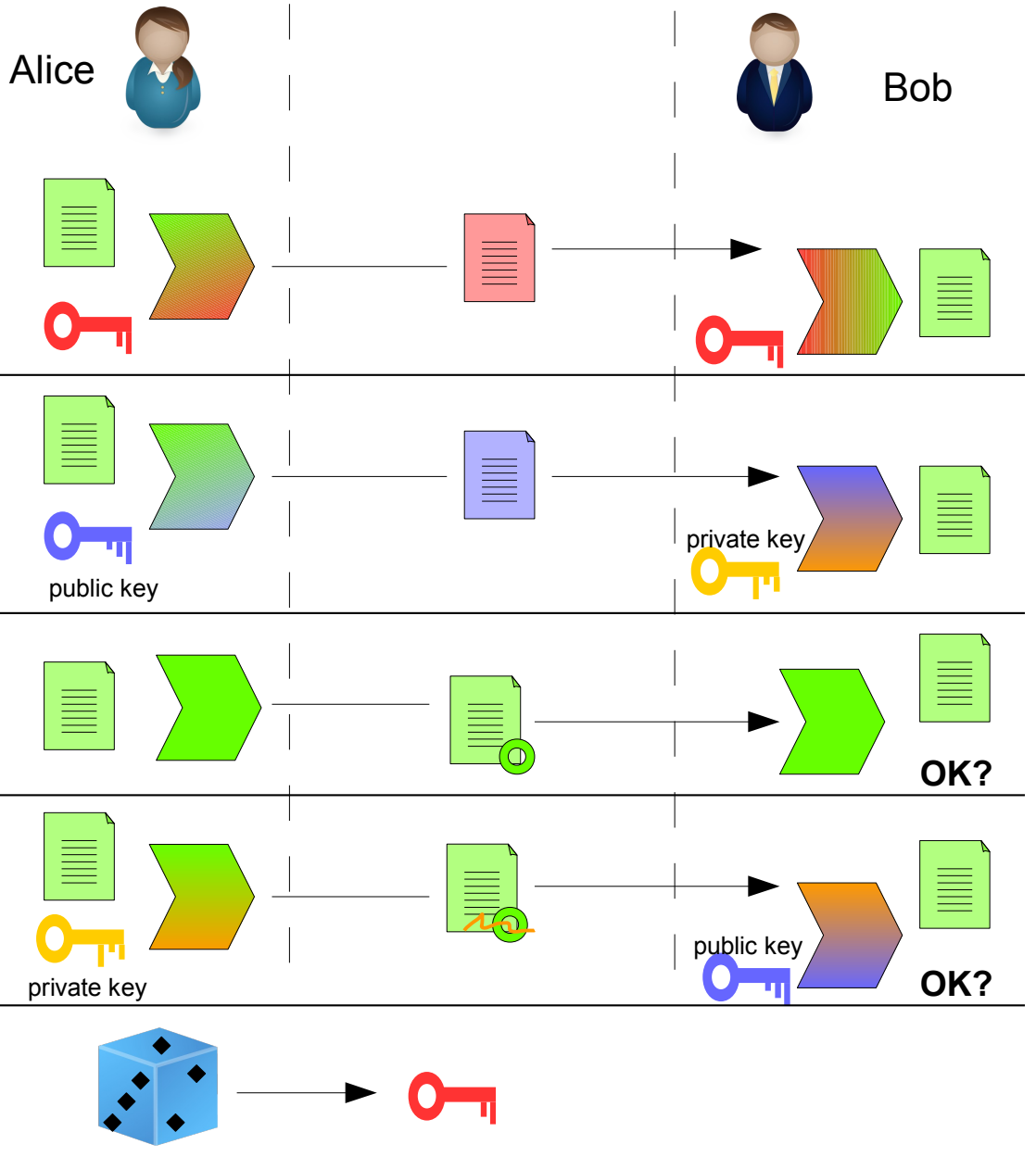
Cryptography Basics: Terms & Definitions

cryptographic operations: ciphers, hashes, random numbers

clear key vs. secure key cryptography

keys & key lengths

Types of cryptographic operations



- Symmetric Encryption
  - DES, 3DES, AES
  - fast
  - encrypt messages, files, disks
- Asymmetric Encryption
  - RSA, ECC, DH
  - Used during key negotiation
  - slow
  - encrypt keys or hashes
- Sealing (Digest/Hashing)
  - MD5, SHA-1, SHA-2, CMAC
  - fast
- Signing
  - Certificates
  - DSA (RSA, ECC)
  - Slow
- (Pseudo) random numbers
  - e.g. for key generation
  - requires entropy

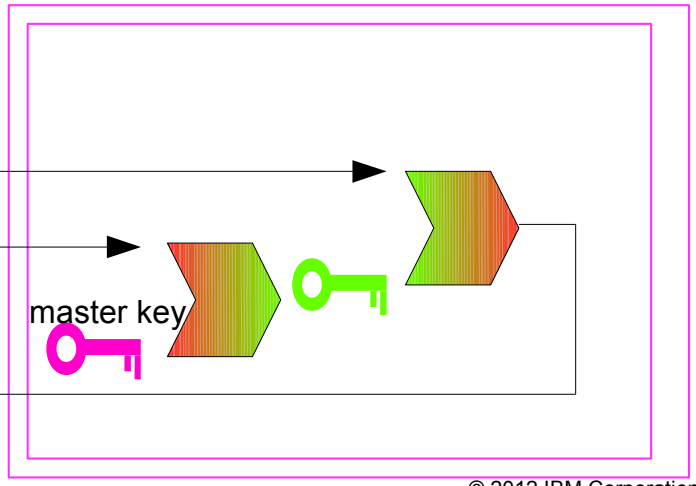
Clear Key vs. Secure Key Crytography

Clear key

Clear key in memory

Secure key

Hardware Security Module (HSM) tamper proof



On Key and Hash Lengths

- Why do we care about lengths a of keys and hashes?
  - the longer a key the lower the probability to guess the right key
  - the longer a hash the lower the probability to guess a matching text for a given hash
  - key and hash sizes that are considered secure change over time
- Cryptography is not security, it is only low probability!? - But “Low” is “V E R Y L O W”!
- Examples of sizes

German Lotto (6 out of 49 with super digit)	1 / 139,838,160 to win approx. €5,000,000
DES: 2^{56} different keys	72,057,594,037,927,936
age of the universe	$5*10^{28}$ ns
AES-128: 2^{128} different keys	38 digits
number of atoms in the earth	about $1.33*10^{50}$
number of atoms in our solar system	about $1.2*10^{57}$
AES-192: 2^{192} different keys	58 digits
AES-256/SHA-256: 2^{256} different keys/hashes	77 digits
number of atoms in the universe	about 10^{82}
SHA-512: 2^{512} different hashes	154 digits

- Key lengths for symmetric keys are not comparable to those of asymmetric keys
  - E.g. only a “few” numbers out of 2^{1024} 1024-bit numbers are valid keys for RSA

Recommended Key and Hash Lengths

Recommending Org	Criteria	Hash	Symmetric	Assymmetric	ECC
ECRYPT II	Level 7: -2030	224	112	2432	224
(2010)	Level 8: -2040	256	128	3248	256
	Level 9: “forever”	512	256	15424	512
BSI	2011 - 2015	224		1976	224
(2011)	2016-2017	256		1976	250
	> 2017	256		1976	250
NSA Suite B	secret	256	128		256
(2010)	top secret	384	256		384
FINSA	2010-2020	200	100	2048	200
(2010)	>2020	256	128	4096	256
NIST	2011-2030	224	112	2048	224
(2007)	>2030	256	128	3072	256
	>>2030	384	192	7680	384
	>>>2030	512	256	15360	512

source: http://www.keylength.com/

Crypto HW support on System z

CP Assist Cryptographic Functions (CPACF)

Cryptographic accelerators and co-processors

z/VM guest crypto support

System z Crypto HW Overview

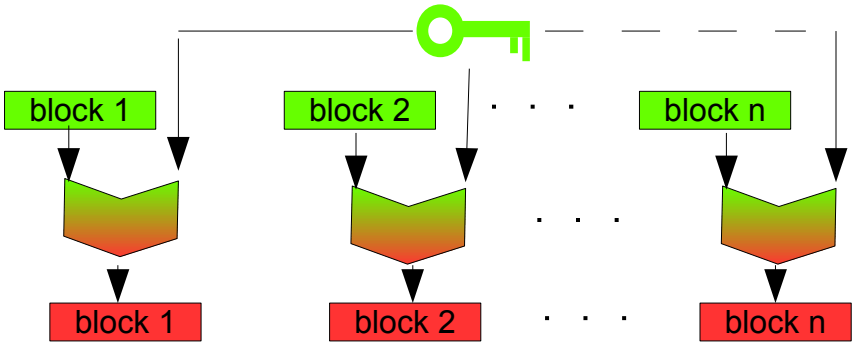
Crypto HW	sym. crypto	asym. crypto	hashes	random number generation	secure crypto
CPACF	DES 3DES AES		SHA-1 SHA-2 CMAC	Pseudo RNG	
Crypto Express Accelecator		RSA (fast)			
Crypto Express Coprocessor		RSA ECC (via CCA)		True RNG	via CCA

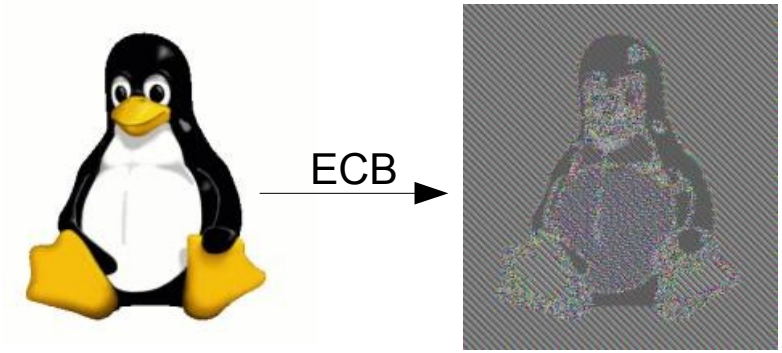
System z HW Crypto Support Details

Crypto feature	z HW	Supported Crypto Mechanisms	Linux support via	
CPACF	z9	DES, TDES, AES-128, SHA-1, SHA-256	in-kernel, libica	
		Modes of Operation: ECB, CBC		
		PRNG	/dev/prandom	
	z10	AES-192, AES-256, SHA-224, SHA-384, SHA-512	in-kernel, libica	
		protected key	CCA (with CEX*C)	new
	z196 / z114	Modes of Operation: CFB, OFB, CTR, XTS, CMAC (GCM,CCM)	in-kernel, libica 2.1	new
CEX2A	z9, z10	RSA (ME, CRT) key <= 2048	libica	
CEX2C	z9, z10	RSA (ME, CRT) key <= 2048	libica, cca	
		RNG	/dev/hwrng	
CEX3A	z10	RSA (ME, CRT) key <= 2048	libica	
	z196 / z114	RSA (ME, CRT) key <= 4096	libica 2.0	new
CEX3C	z10	RSA (ME, CRT) key <= 2048	libica, CCA	
		RNG	/dev/hwrng	
	z196 / z114	RSA (ME, CRT) key <= 4096	libica, cca	
		ECC	CCA 4.1	new

CPACF MSA4 extensions: new modes of operation

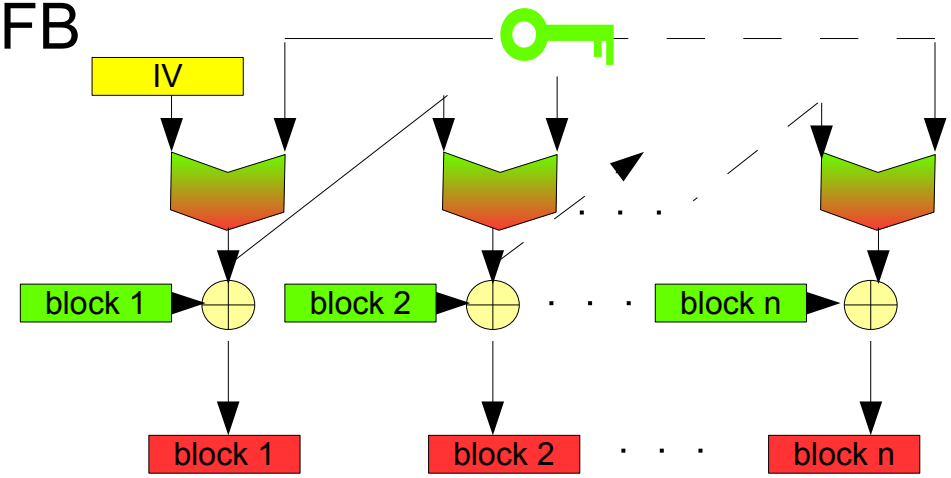
ECB





http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

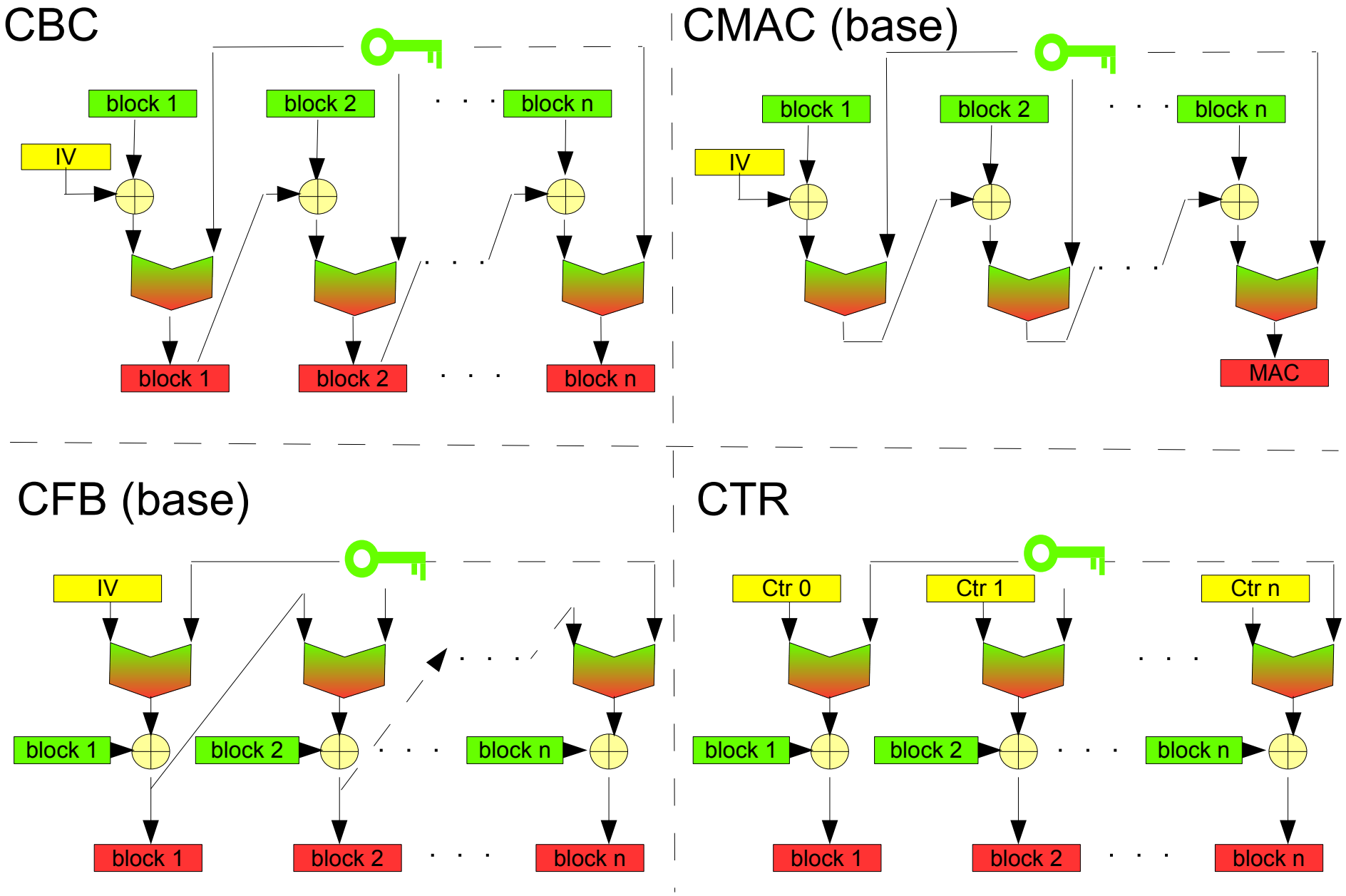
OFB



mode	kernel	libica
ECB	X	X
CBC	X	X
CFB		2.1
CTR	3.0	2.1
OFB		2.1
XTS (AES)	3.0	2.1
CMAC		2.1 (AES)
GHASH	3.0	

- for DES, 3DES, AES en/decryption
- standardized by NIST SP 800-38A,B;D;E

CPACF MSA4 extensions: new modes of operation (cont'd)



CPACF MSA4 extensions: new modes of operation (cont'd 2)

Reasons for having different modes of operation

- avoid the effect that the same plain data always encrypts to the same cipher text
- deal with transmission problems
  - detect missing / forged data
  - self synchronization
- restrictions on message length
  - multiples of cipher block length
  - stream cipher
- kind of data
  - messages in communication
  - stored data

Handling modes of operation

- initialization vector (IV) or initial counter value (Ctr 0) must be transferred in addition to message
  - in plain text
  - must not repeat
- standard crypto libraries like openssl or opencryptoki support different modes of operation
  - extend openssl engine and ica token to support modes of operation from libica
- the kernel supports some modes of operation

Crypto Adapters

- 2 adapters per CEX2 or CEX3 feature
- Adapter modes
  - copressessor: CEX2C, CEX3C
    - clear key

      RSA (<= 4k keys)

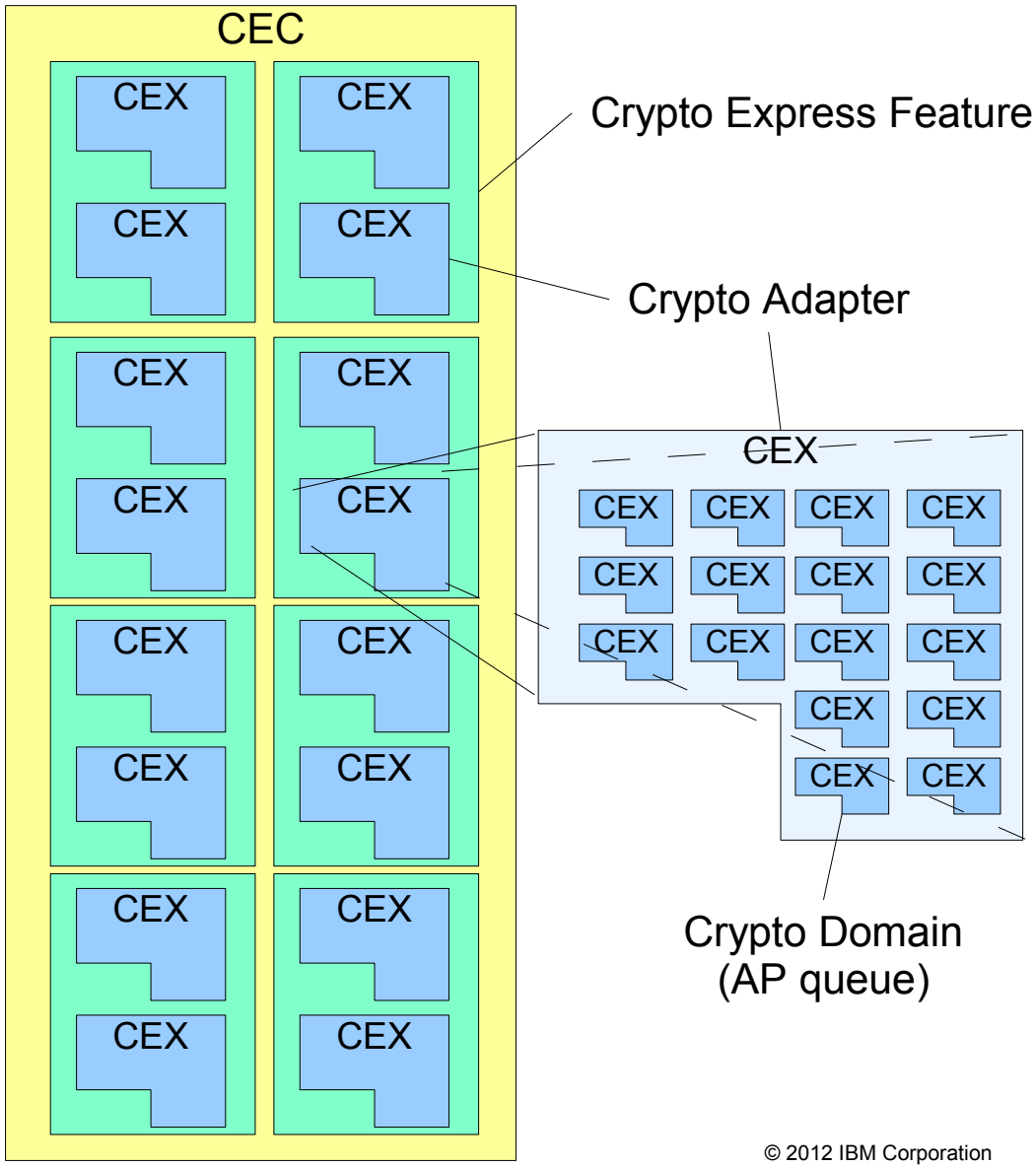
      long random numbers (RNG)
    - secure key

      CCA (incl. ECC)
  - accelerator: CEX2A, CEX3A
    - clear key

      RSA (<= 4k keys)
- up to 16 Domains (queues)
- LPAR access to adapters and domains must be configured at SE

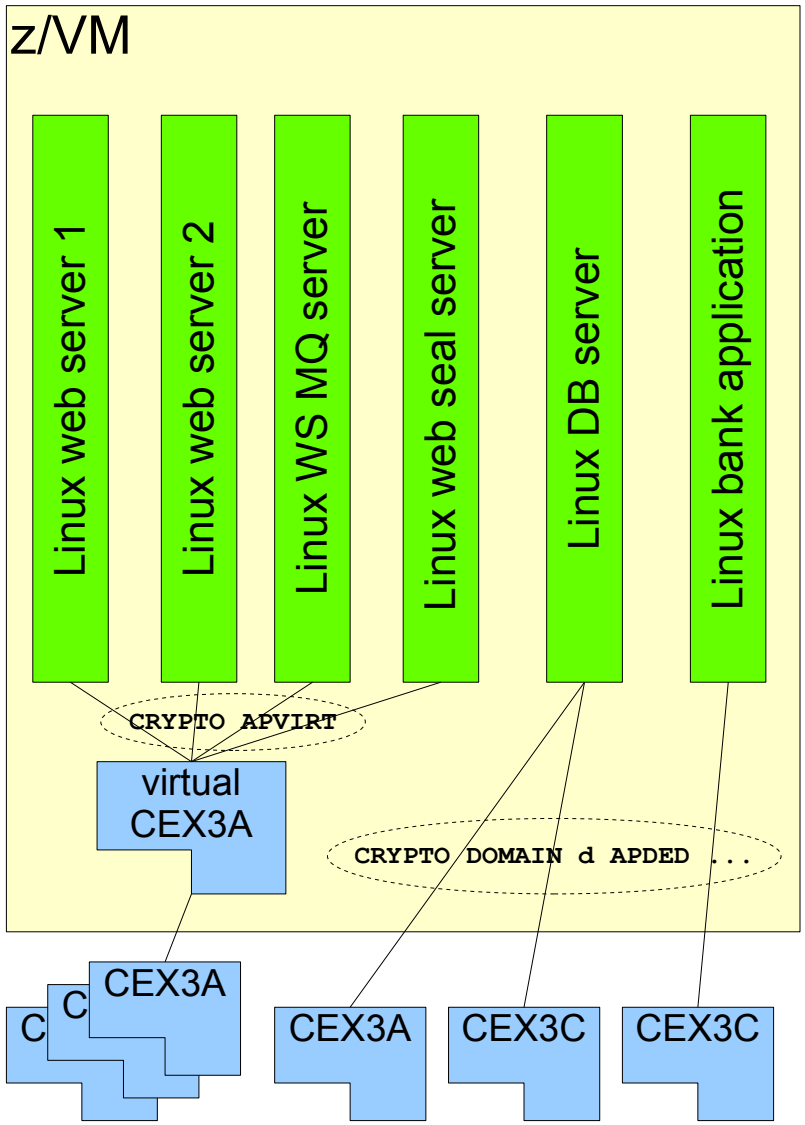
On Features, Adapters, APs, Domains, Queues and such

- There may be up to 8 crypto express features per CEC
- Each CEX2, CEX3 feature has two adapters (aka APs)
- Each adapter has an AP Id
- Each adapter has a mode
  - coprocessor or
  - accelerator
- Each adapter can be divided in upto 16 domains (HW virtualization)
- each domain in an AP is represented in SW by an AP queue
- Configuration constraints
  - each LPAR 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/AP queue per AP



z/VM Crypto Guest Support

- A guest may have
  - either dedicated adapters
    - `CRYPTO DOMAIN d APDED a1 a2 ...`
  - or shared adapters
    - `CRYPTO APVIRT`
- Shared adapters
  - are of a single type
    - uses only highest priority type
    - priority:
      - CEX3A > CEX2A > CEX3C > CEX2C
  - should only be used for clear key operations
  - Support of CEX3 requires VM64656 before z/VM 6.2
  - Support for 4k RSA keys requires APAR VM64829 before z/VM 6.2
- Checking Crypto Configuration
  - show status of crypto facilities
    - `Q CRYPTO [ DOMAINs [Users] ]`
  - show status of crypto facilities of guest
    - `Q V CRYPTO`



The Linux on System z SW crypto stack

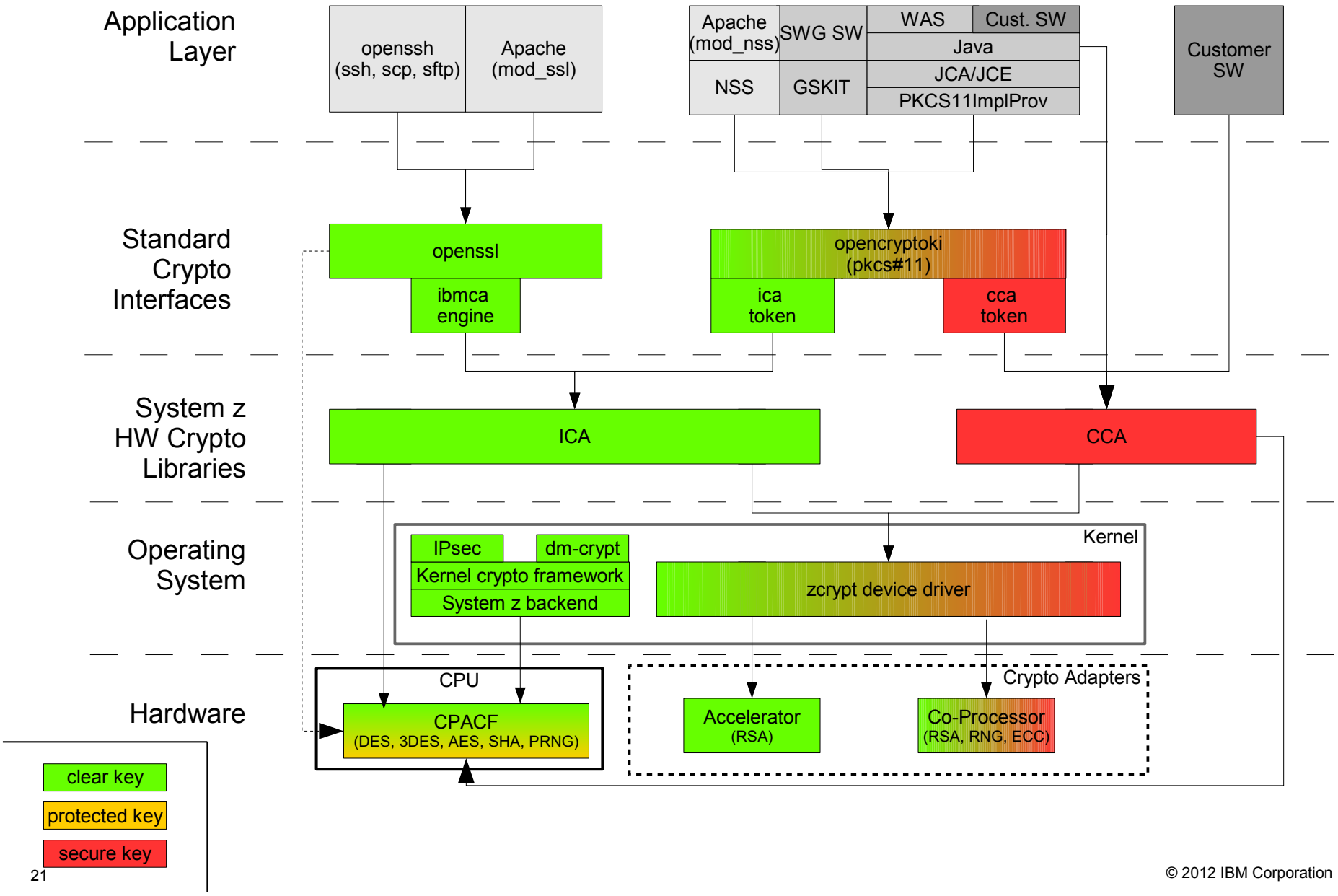
kernel support

libica

CCA library

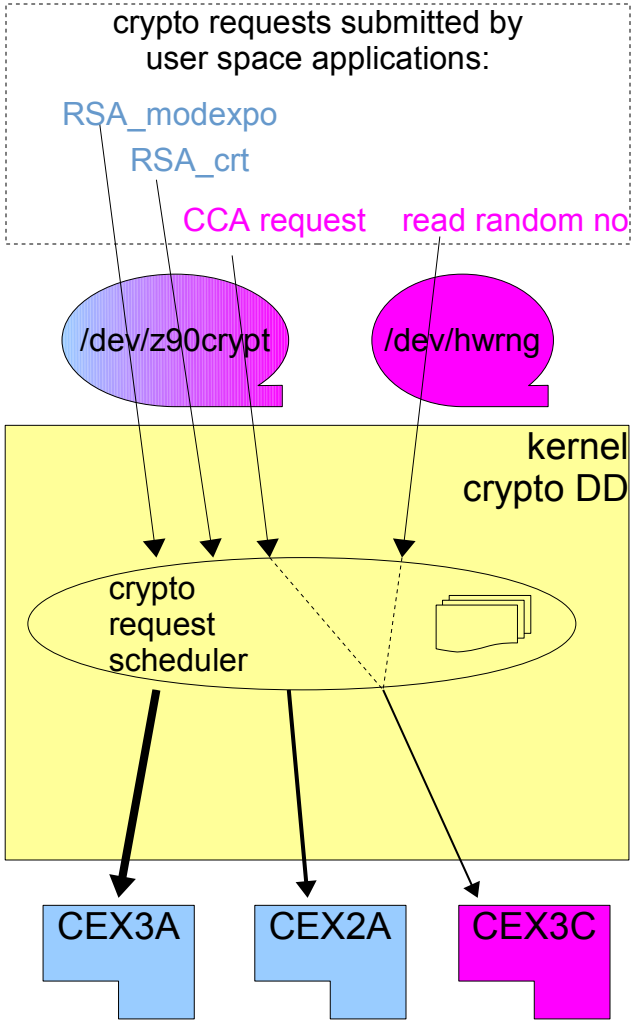
openssl /opencryptoki

Linux on System z Crypto Stack



The Linux on System z Crypto (Adjunct Processor) Device Driver

- the device driver
  - ap (zcrypt_cex2a, zcrypt_pcixcc, zcrypt_pcicc, zcrypt_pcica, zcrypt_api)
  - z90crypt
- maps all crypto card to one device (typically `/dev/z90crypt`)
  - supports RSA functions (modular exponent & Chinese Remainder Theorem)
  - supports CCA functions (for CEX*C, 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)
      - the card performance & load (clear key only)
    - 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 post z10 LPARs: thin interrupts
- provides real random numbers (aka “long random numbers”)
  - on `/dev/hwrng` (for CEX*C)
- dynamically adding or removing crypto adapters
  - on LPAR when timer `/sys/bus/ap/config_time` expires



Crypto Adapter Administration

lszcrypt shows crypto device attributes

- - V - VV verbose adapter attribute
- -b show AP bus attributes
- 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)
```

chcrypt can modify some attributes

- -e / -d enabale/disablble adapter
- -p / -n enable/disable poll thread
- -t set high resolution polling timer
- -c set timer for reconfiguration scanner

See “Device Driver Features and Commands” for details

```
/sys/bus/ap
        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
           depth
           driver → ../../../bus/ap/drivers/cex2a
           hwtype
           online
           request_count
           subsystem -> ../../../bus/ap
           type

        card03
            depth
            driver → ../../../bus/ap/drivers/pcixcc
            hwtype
            online
            request_count
            subsystem → ../../../bus/ap
            type
```

The New Version of the Libica Library

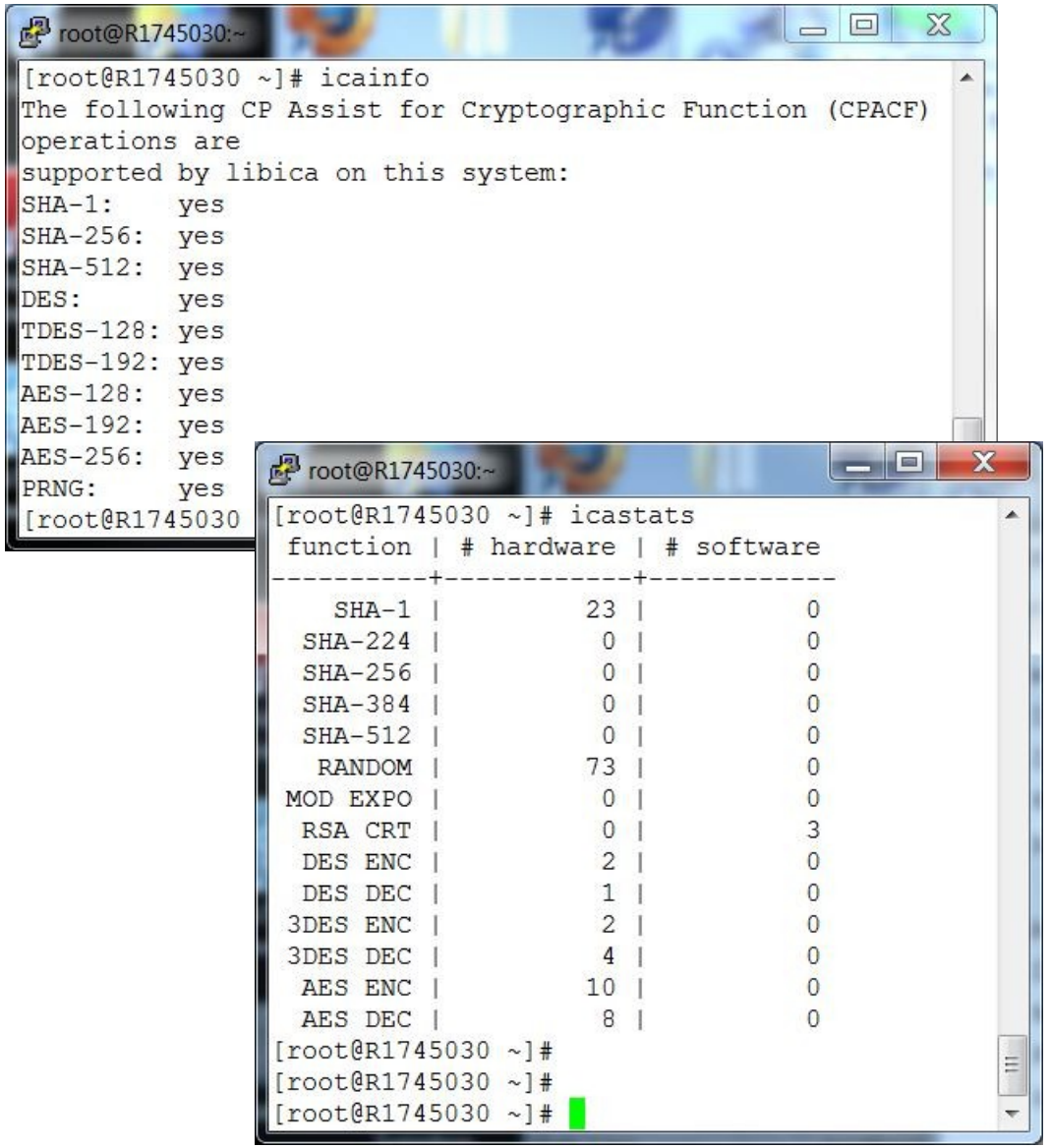
- libica library provides a C API for
  - symmetric crypto & hash mechanisms (CPACF support)
  - RSA encryption, ME,CRT decryption (coprocessor & accelerator support)
  - RSA key generation (SW)
  - pseudo random numbers (coprocessor/CPACF/kernel)
- libica version 2.1
  - upward compatible with version 2.0
  - New symmetric mechanisms according NIST standards
    - DES-CFB, DES-OFB, DES-CTR
    - 3DES-CFB, 3DES-OFB, 3DES-CTR
    - AES-128-CFB, AES-128-OFB, AES-128-CTR, AES-128-XTS, AES-128-CMAC
    - AES-192-CFB, AES-192-OFB, AES-192-CTR, AES-192-CMAC
    - AES-256-CFB, AES-256-OFB, AES-256-CTR, AES-256-XTS, AES-256-CMAC
    - example AES OFB function:

```
unsigned int ica_aes_ofb(
        const unsigned char     *in_data,
        unsigned char           *out_data,
        unsigned long           data_length,
        const unsigned char     *key,
        unsigned int            key_length,
        unsigned char           *iv,            /* input/output value */
        unsigned int            direction);     /* encryption /decryption */
```

  - new version function
- libica is the backend to openssl/ibmca engine and opencryptoki/ica token
- libica book: libica Programmer's Reference - SC34-2602-xx

icatools

- icainfo
  - displays which CPACF ciphers and hash algorithms are available on your system
- icastats
  - displays how often which ciphers, hash and random number generators were processed by libica using HW and SW
  - works only while there is at least one process that uses libica
  - trick to obtain a “long running” process using libica
    - configure openssl to use ibmca engine
    - start “openssl speed”
    - suspend “openssl speed” process (ctrl-z)



Secure Key Cryptography: The CCA 4.1 Library

- Common Cryptographic Architecture (CCA)
  - IBM proprietary architecture for secure key cryptography on crypto coprocessors
  - targeted to support banking and financial applications
  - Linux on System z cca library available as free download (see link below)
  - usage
    - direct calls to CCA C API
    - JAVA: via CCA JNI
    - calls to opencryptoki API (PKCS#11, restricted functionality only)
- New functions in CCA 4.1
  - Enhanced PIN security with the addition of ANSI X9.8 restriction capabilities
  - Wrap CCA keys in Cipher-Block Chaining (CBC) mode
  - Elliptic Curve Cryptography (ECC) support
    - key generation
    - digital signature generation and verification using the EC Digital Signature Algorithm
  - Hashed Message Authentication Code (HMAC) support for key generation and processing, but not for key storage.
- download: http://www-03.ibm.com/security/cryptocards/pciecc/ordersoftware.shtml
- programmers guide: http://www-03.ibm.com/security/cryptocards/pciecc/library.shtml

Standard Crypto Libraries

- **OpenSSL**
  - open source library to implement SSL/TLS
  - a de-facto standard
  - has some limited build-in exploitation for CPACF functions since version 1.0 (RHEL 6.0)
  - can be extended by “engines” to exploit HW support
    - the *ibmca engine* exploits System z clear key crypto HW (CPACF & CEX) via libica

      ibmca version 1.2 (available with RHEL 6.2 and SLES 11 SP2) supports ECB, CBC, OFB (new) and CFB (new) modes of operation

- **OpenCryptoki**
  - open source implementation of PKCS#11 standard
  - has “slots” into which “tokens” can be put
  - a token represents a HW crypto feature (e.g., a smart card or a crypto adapter)
    - the *ica token* provides access to the System z clear key crypto HW (CPACF & CEX) via libica

      the ica token of opencryptoki 2.4 (available with RHEL 6.2 and SLES 11 SP2) supports ECB, CBC and CTR (new) modes of operation
    - the *CCA token* provides access to the System z secure key crypto HW (CPACF & CEX*C) via cca library

      the CCA token of opencryptoki 2.4 (available with RHEL 6.2 and SLES 11 SP2) supports ECDSA
  - slot management requires PINs for security officer (SO) and user
  - can be called and exploited by Java (JCE) via IBMPKCS11Impl provider

IBM Crypto Libraries

- C/C++
  - GSKIT (part of IBM Products)
    - ICC - symmetric crypto uses CPACF functions (ECB & CBC modes) directly
    - SSL/TLS support can use RSA via opencryptoki/icatoken

- Java
  - Java Cryptography Architecture (JCA) / Java Cryptography Extension (JCE)
    - JCA providers implement cryptographic mechanisms
    - IBMPKCS11Impl provider implements cryptographic HW support via opencryptoki
    - configuration of available providers via java.security file
    - configuration of IBMPKCS11Impl provider by special configuration file referenced as argument to the provider
    - requires to “log into” PKCS#11 token
    - links:

      JCA: http://www.ibm.com/developerworks/java/jdk/security/60/secguides/JceDocs/CryptoSpec.html

      PKCS11Impl: http://www.ibm.com/developerworks/java/jdk/security/60/secguides/pkcs11implDocs/IBMJavaPKCS11ImplementationProvider.html

System z HW Crypto Enablement

- CPACF -- install CPACF feature
  - check CPC Details/Instance Information on SE or HMC
  - the feature is free, but must be installed by a CE
- /dev/prandom -- modprobe prng
- /dev/hwrng -- modprobe z90crypt
- in-kernel crypto -- enable CPACF
- RNG based in-kernel entropy -- modprobe z90crypt (SLES 11 SP2 only)
- libica -- install libica rpm, enable CPACF,
  - optional RSA support: modprobe z90crypt,
  - optional PRNG support: modprobe prng
- CCA -- install cca rpm, modprobe z90crypt
- openssl -- install ibmca rpm, configure openssl.cfg, enable libica
  - see “Using Hardware Cryptographic Support With OpenSSH in Linux on System z” on how to configure openssl.cfg
- opencryptoki -- install opencryptoki rpm,
  - enable libica (for clear key) or enable CCA (for secure key)
  - start opencryptoki: pkcs11_startup, pkcsslotd
  - initialize token(s): set SO and user pins with pkcsconf

Random Numbers

- Where needed
  - in cryptography: session keys, IVs, nonces, salts
  - in kernel: e.g: address space randomization
  - simulation & modelling
  - random sampling
  - ...
- Types of random numbers
  - pseudo random numbers (seed + mathematical function)
  - cryptographically secure pseudo random numbers
  - true random numbers
- “Entropy”
  - randomness observable to kernel
  - kernel collects entropy input in entropy pool
  - used to (re)seed pseudo random number generators (PRNGs)
- Sources of (pseudo) random numbers in Linux on System z
  - /dev/urandom (platform independent PRNG)
  - /dev/random (platform independent “RNG”)
  - /dev/prandom (PRNG using CPACF instruction)
  - /dev/hwrng (RNG using CEX*C function)
  - libica

References

- Kernel & kernel management tools:
  - Device Drivers Features and Commands (part “Security”)
    http://www.ibm.com/developerworks/linux/linux390/documentation_dev.html
- Good overview on using System z Crypto HW (LPAR & z/VM configuration)
  - Redbook: Security for Linux on System z (SG24-7728-00), Chapter 5
    http://www.redbooks.ibm.com/abstracts/sg247728.html?Open
- Programming against the libica library
  - libica Programmer's Reference - SC34-2602-xx
    http://www.ibm.com/developerworks/linux/linux390/documentation_dev.html
- Enabling of openSSH (openssl configration)
  - Using Hardware Cryptographic Support With OpenSSH in Linux on System z
    http://www.mainframezone.com/it-management/using-hardware-cryptographic-support-with-openssh-in-linux-on-system-z
- Enabling HW Crypto for IBM HTTP (opencryptoki configuration)
  - Configuring WebSphere V7.0 and IBM HTTP Server V7.0 to use Cryptographic Hardware for SSL Acceleration on Linux on IBM System z
    http://www-03.ibm.com/systems/resources/was7_ish7_hwcrypto.pdf
- Using HW crypto from Java (JSSE)
  - Exploiting IBM System z Cryptographic Hardware using
    http://public.dhe.ibm.com/software/dw/linux390/perf/ZSW03153-USEN-01.pdf

Conclusion

- **Crypto support for Linux on System z helps you to**
  - Leverage unique 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, NSS/PKCS#11 or GSKIT/PKCS#11
  - Heighten security by protecting and storing cryptographic keys in the hardware

- There is new improved crypto support
  - for CPACF (new modes of operation for symmetric encryption)
  - crypto accelerators: 4k RSA key
  - Coprocessors/secure key: CCA with ECC

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