Pervasive Encryption for z/VM and Linux on z

Brian W. Hugenbruch, CISSP
IBM Z Security for Virtualization & Cloud
z/VM Development *Lab: Endicott, NY*
@Bwhugen

Trademarks

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

BladeCenter*	FICON*	OMEGAMON*	RACF*	System z9*	zSecure
DB2*	GDPS*	Performance Toolkit for VM	Storwize*	System z10*	z/VM*
DS6000*	HiperSockets	Power*	System Storage*	Tivoli*	z Systems*
DS8000*	HyperSwap	PowerVM	System x*	zEnterprise*	
ECKD	IBM z13*	PR/SM	System z*	z/OS*	

* Registered trademarks of IBM Corporation

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.

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.

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

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.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and

Linux is a registered trademark of Linus Torvalds 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.

OpenStack is a trademark of OpenStack LLC. The OpenStack trademark policy is available on the OpenStack website.

TEALEAF is a registered trademark of Tealeaf, an IBM Company.

Windows Server and the Windows logo are trademarks of the Microsoft group of countries.

Worklight is a trademark or registered trademark of Worklight, an IBM Company.

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

Other product and service names might be trademarks of IBM or other companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

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

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

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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

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

This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Disclaimer

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "AS IS" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

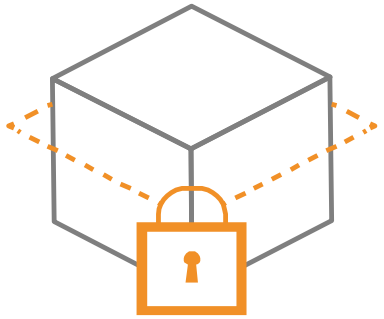
In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

Any performance data contained in this document was determined in a controlled environment and, therefore, the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environments.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming or services in your country.

The IBM Z Pervasive Encryption Strategy

- Extensive use of encryption is one of the most impactful ways to help reduce the risks and financial losses of a data breach and help meet complex compliance mandates.

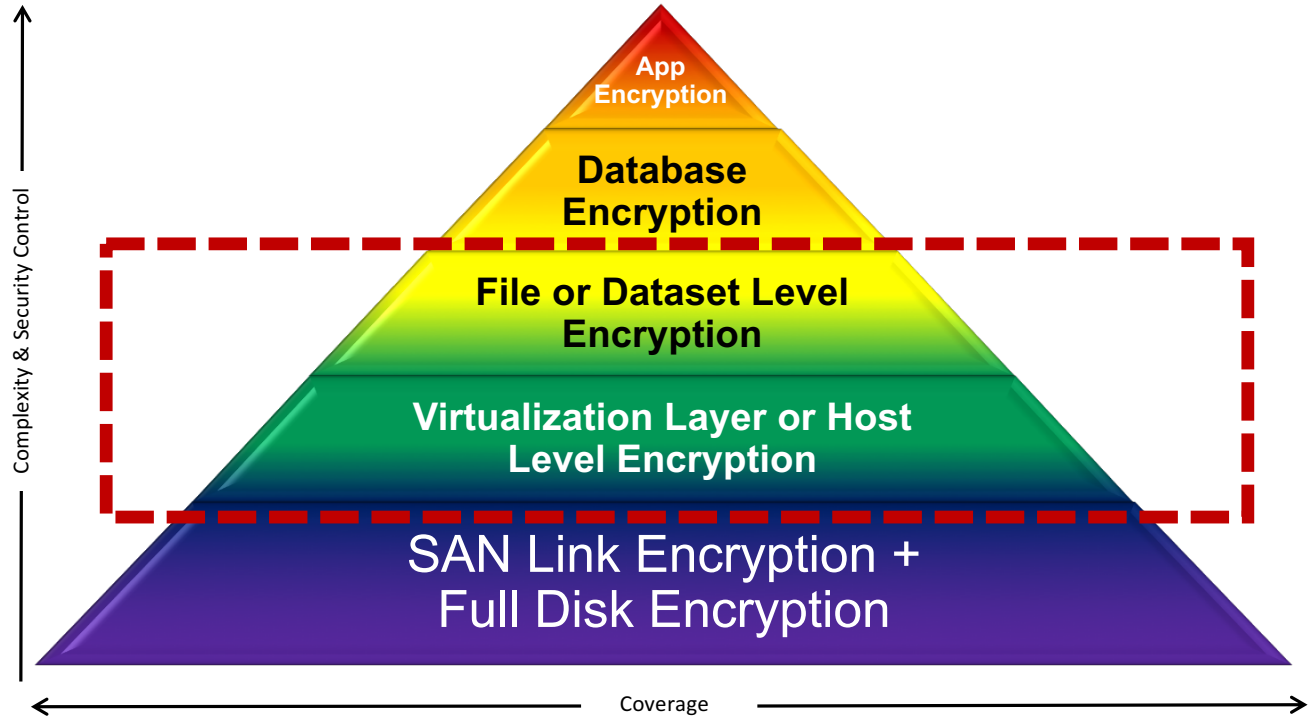


- However, implementing encryption can be a complex process …
  1. What data should be encrypted?
  2. Where should encryption occur?
  3. Who is responsible for encryption?

Transparent and consumable approach to enable extensive encryption of data in-flight and at-rest to substantially simplify & reduce the costs associated with protecting data & achieving compliance mandates

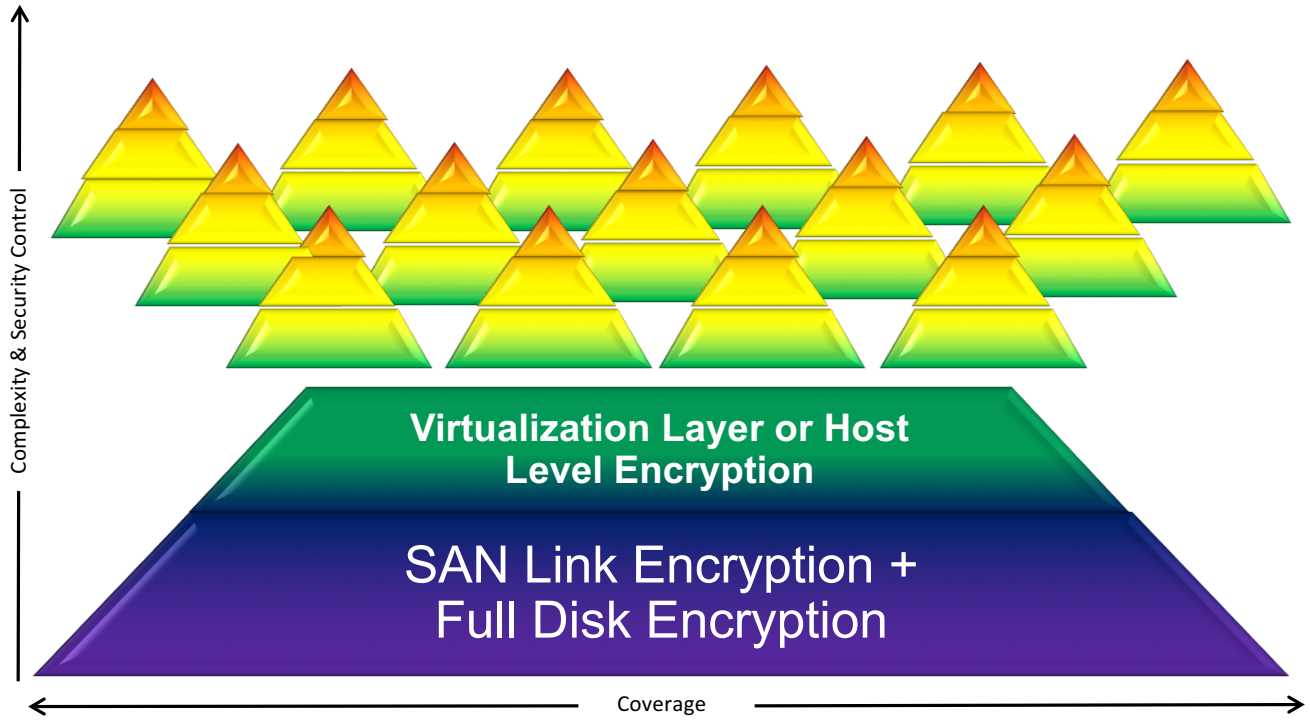
IBM Z Pervasive Encryption

From a Virtualization Point of View



IBM Z Pervasive Encryption

From a Virtualization Point of View



IBM Z Pervasive Encryption for z/VM and Linux on z

z14 – Designed for Pervasive Encryption

- **CPACF** – Dramatic advance in bulk symmetric encryption performance
- **Crypto Express6S** – Doubling of asymmetric encryption performance for TLS handshakes

z/VM – Virtualizing Encryption for Linux

- **Virtualization** of IBM Z Crypto Hardware (**updated August 2017**)
- Crypto Express **acceleration** for encrypted data in flight (**available March 2017**)
- **Encrypted Paging** for z/VM (**coming 4Q2017**)

Linux on z – Full Power of Linux Ecosystem plus z14 Capabilities

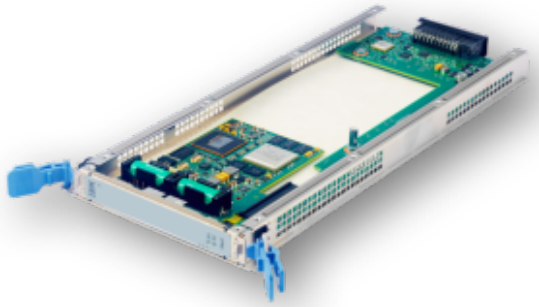
- **LUKS dm-crypt** – Transparent file & volume encryption using industry unique CPACF protected-keys
- **Network Security** – Enterprise scale encryption and handshakes using zNext CPACF and SIMD
- **Secure Service Containers** – Automatic protection of data and code for virtual appliance

IBM Z Cryptographic Features

- **IBM z Systems provide two flavors for offloading and accelerating cryptographic operations which help you to**
  - Move cryptographic workload away from central processors
  - Heighten your security level by protecting and securing keys
  - Accelerate encryption and decryption

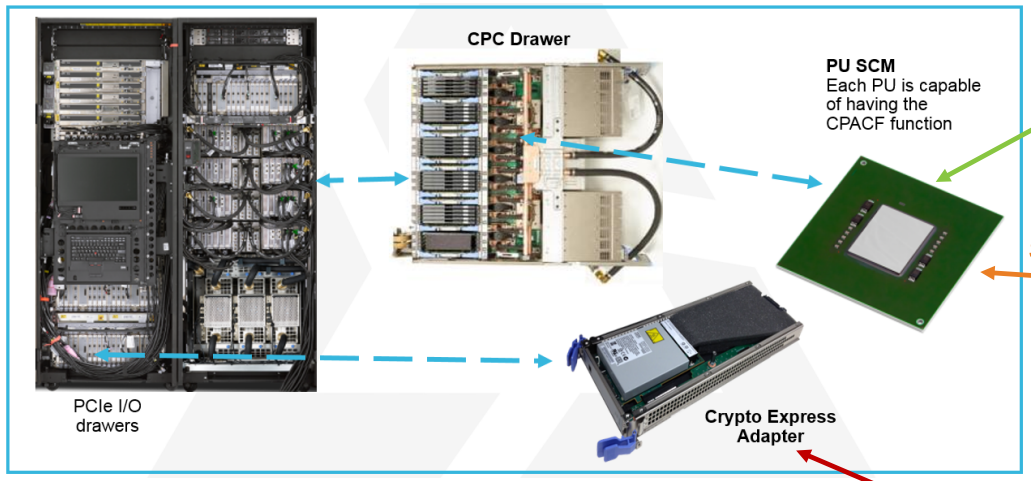
- *CP Assist for Cryptographic Function* (**CPACF**)
  - Support for **symmetric** and hashing algorithms included in every CP and IFL
  - Pseudo-random number generator

- ***Crypto Express*** *features*
  - **Asymmetric** and hashing algorithm offload
  - Host master-key storage
  - Hardware RNG
  - PKCS #11 cryptographic support



What are clear, secure and protected keys?

Secure keys have key values that are encrypted by a Master Key on a tamper-responding Crypto Express adapter.



High speed

Clear Key

Key values are not encrypted. Crypto operations may be performed in CPACF or on a Crypto Express adapter

Speed & Security Hybrid

Protected Key

Key values are encrypted under a CPACF wrapping key. Crypto operations are performed only using CPACF

High security

Secure Key

Key values are encrypted under a Master Key. Crypto operations are performed only on a Crypto Express adapter

IBM Z Pervasive Encryption for z/VM and Linux on z (By Layer)

		For more, see sessions:
▪ **Partition**	*with Secure Service Containers*	▪ GS07 (Erich Amrehn)
▪ **Hypervisor**	*Hardware Virtualization* *TLS/SSL Server (Data in flight)* *z/VM Encrypted Paging*	▪ GS13 (Brian Hugenbruch)
▪ **Linux (Data At Rest)**	*dm-crypt and Protected Key AES (amongst other things)*	▪ GS10 (Manfred Gnirss)
▪ **Linux (Data In Flight)**	*openSSL and IPSec*	
▪ **Use-Cases**	*or, what can you do with all this?*	▪ VM02 (Christian Tatz) ▪ GS11 (Wilhelm Mild)

With thanks to Reinhard Buendgen and Michael Jordan, IBM

Secure Service Containers

Pervasive Encryption – Partition Layer

Data Protection // Secure Service Container

Extending the value of z hardware crypto

Client Value Proposition:

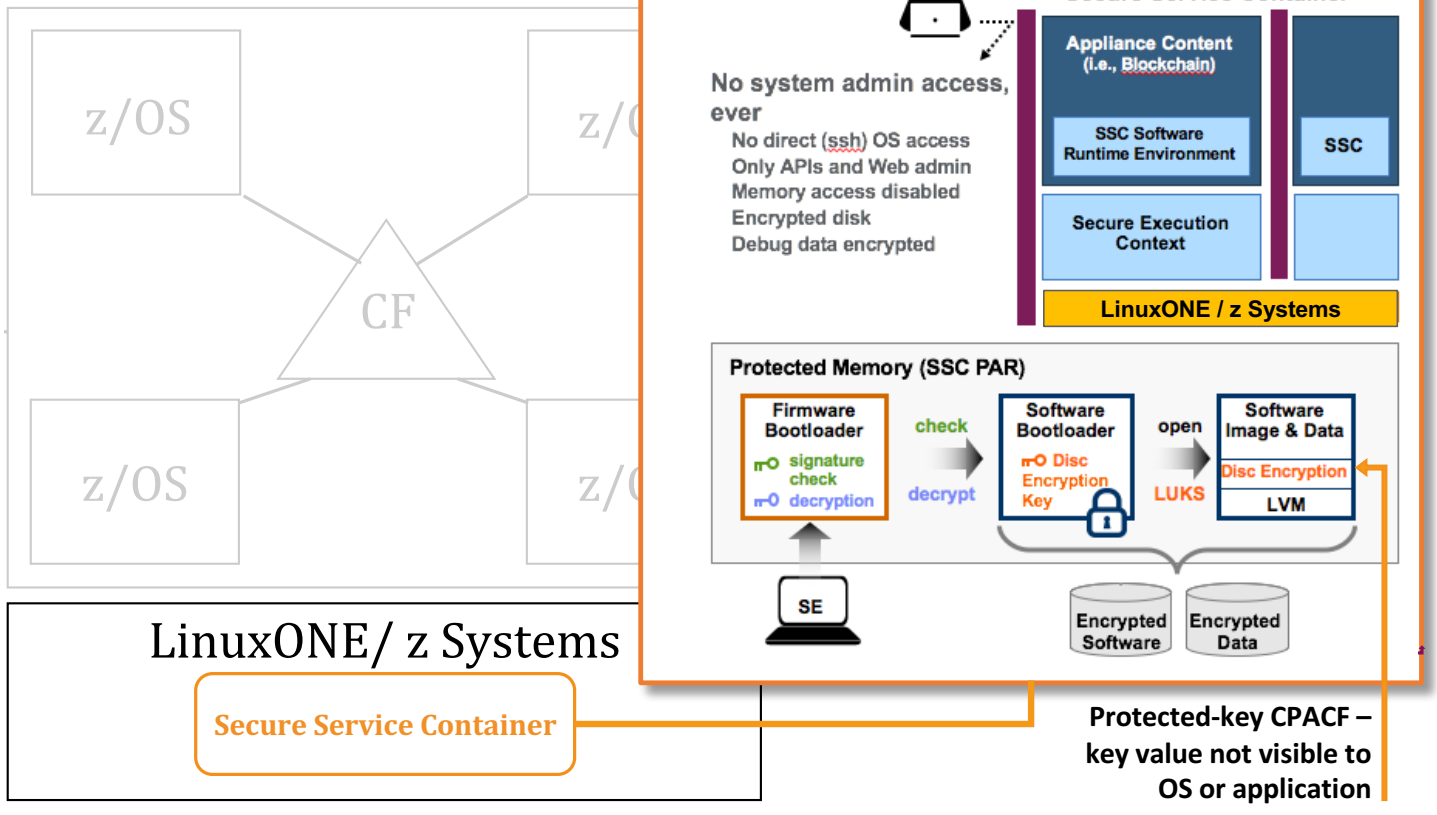
Simplified, fast deployment and management of packaged solutions

Tamper protection during Appliance installation and runtime

Confidentiality of data and code running within the Appliance both at flight and at rest

Restricts administrator access to workload and data

Secure Service Container architecture builds on the value z systems hardware crypto using a runtime environment designed to help clients reduce risk.



Secure Service Containers (SSC)

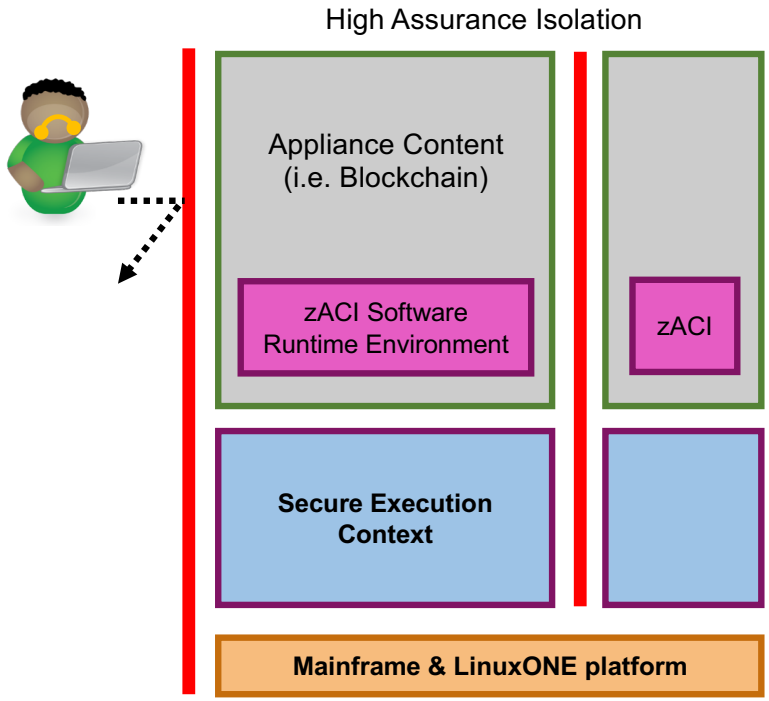
The Base Infrastructure to Host and Build Software Appliances

- Provides simplified mechanism for fast deployment and management of packaged solutions
- Provides tamper protection during Appliance installation and runtime
- Ensure confidentiality of data and code running within the Appliance – both at flight and at rest
- Management provided via Remote APIs (RESTful) and web interfaces
- Enables Appliances to be delivered via distribution channels

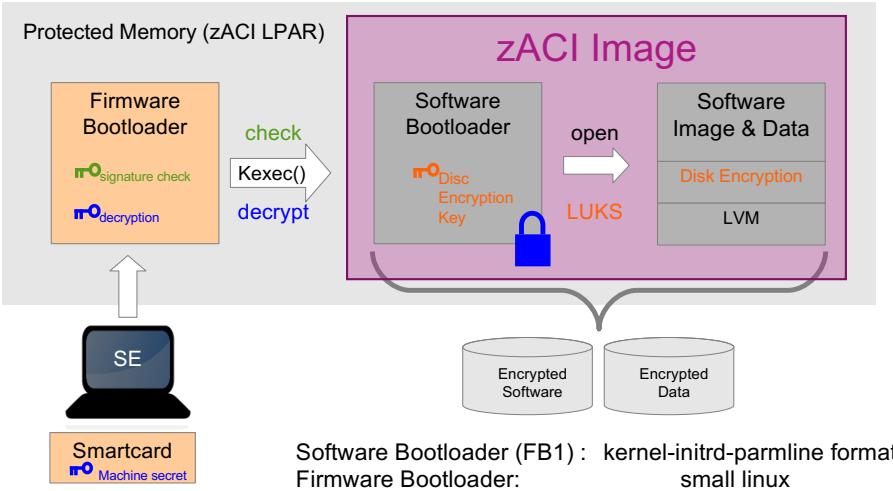
Secure Service Containers: Access Controls

- No system admin access
  - Once the appliance image is built, OS access (ssh) is not possible
    - Only Remote APIs available
  - Memory access disabled
  - Encrypted disk
  - Debug data (dumps) encrypted

- Strong isolation between SSC instances
  - Based on LinuxONE EAL5+ protection profile
  - Requires dedicated HW



SSC: Firmware Image Security

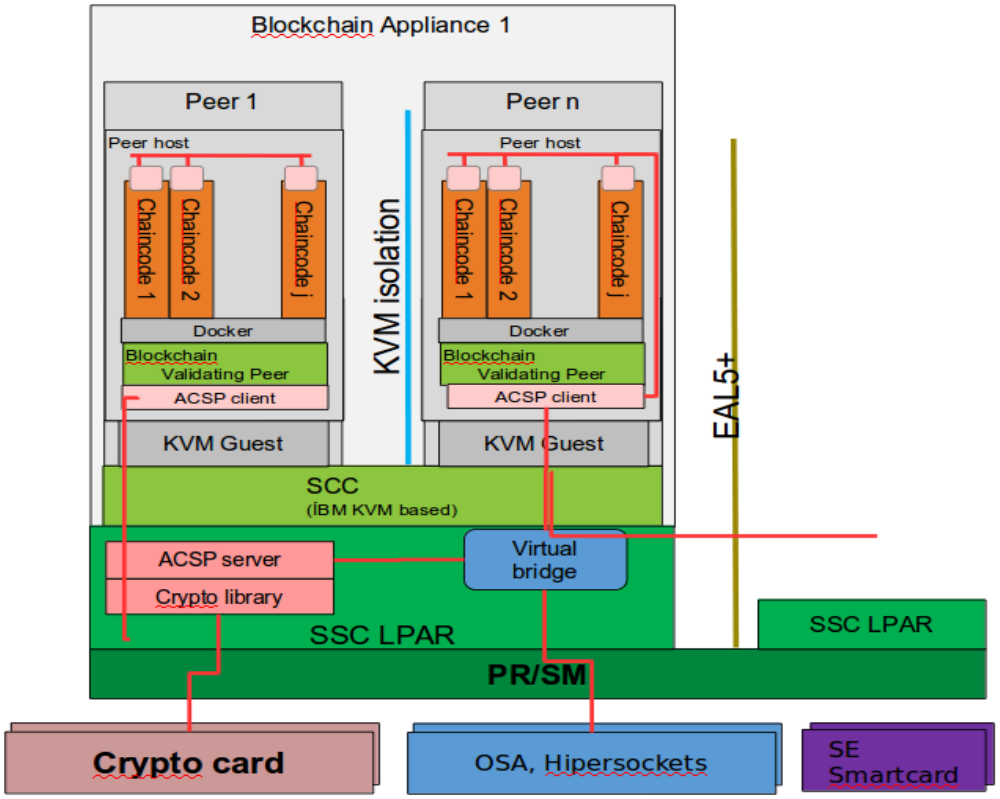


Software Bootloader (FB1) : kernel-initrd-parmline format
Firmware Bootloader: small linux

Boot sequence

1. Firmware bootloader is loaded in memory
2. Firmware loads the software bootloader from disk
   - Check integrity of software bootloader
   - Decrypt software bootloader
3. Software bootloader activate encrypted disks
   - Key stored in software bootloader (encrypted)
   - Encryption/decryption done on the flight when accessing appliance code&data
4. Appliance designed to be managed by remote APIs only
   - REST APIs to configure Linux and apps
   - No ssh (allowed in dev mode)

SSC Application Security



z/VM Encrypted Paging

Pervasive Encryption – Hypervisor Layer

Pervasive Encryption and z/VM

Bringing Pervasive Encryption to z/VM involves the following:

1. Ease of use needs to be mandatory
   - Client interviews and feedback a must

2. Enablement of **hardware facilities for guest usage**
   - If we're not first and foremost a virtualization platform, we're off-mission
   - Exploitation of crypto hardware for guests needs to happen Day 1

3. Encryption of security-pertinent hypervisor components
   - Question of **security policy** vs. **performance** vs. **risk**

z/VM Support of z14 Cryptographic Hardware

PTF for APAR VM65942

- New CPACF facilities and Crypto Express6S orderable features
  - CPACF now includes TRNG and AES GCM
  - Some fantastic performance benefits over previous hardware

- Elliptic Curve Cryptography for Shared Crypto Domains ("APVIRT")
  - All domains assigned to the CP-managed queues must be CCA coprocessors
  - No change to dedicated crypto domains – those function as before
  - Accelerates use of elliptic curve crypto for Linux or z/OS guests

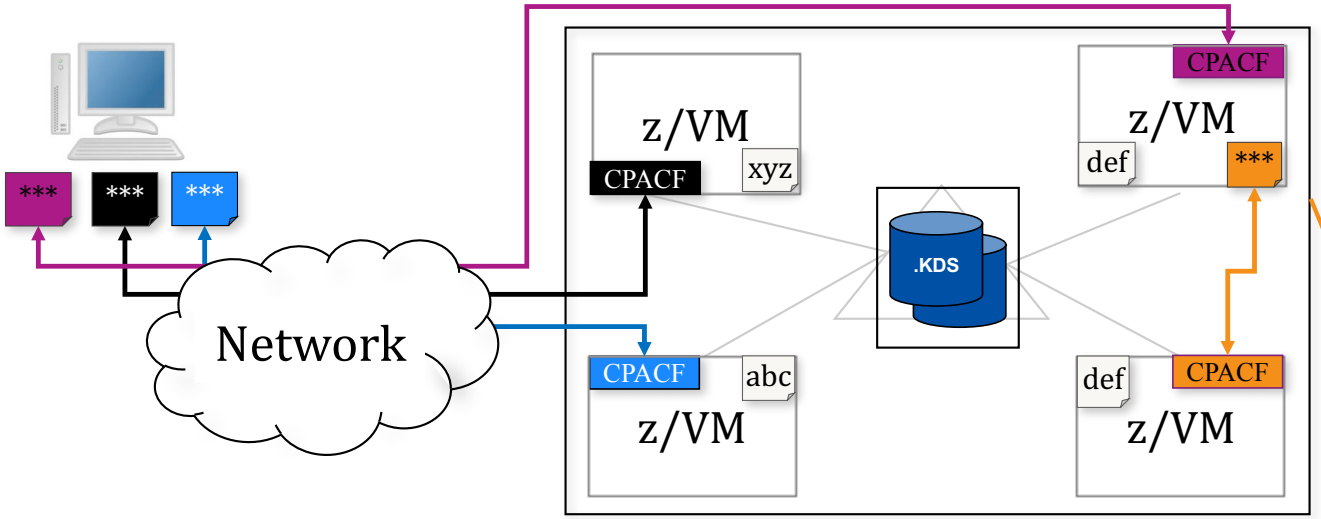
    z/VM TLS/SSL Server doesn't use ECC yet

– For more information, see the z14 Announce Letter at: https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS117-044&appname=USN

Data Protection // z/VM Network Security

Protection of data in-flight

z/VM 6.4
PTF for APAR VM65993



z/VM Single System Image cluster

Client Value Proposition:

Not all organizations use host-based network encryption today …
reduced cost of encryption enables broad use of network encryption

z/VM Secure Communications

- **Threat**: disclosure of sensitive data in flight to the hypervisor layer
- **Solution**: encrypt traffic in flight.

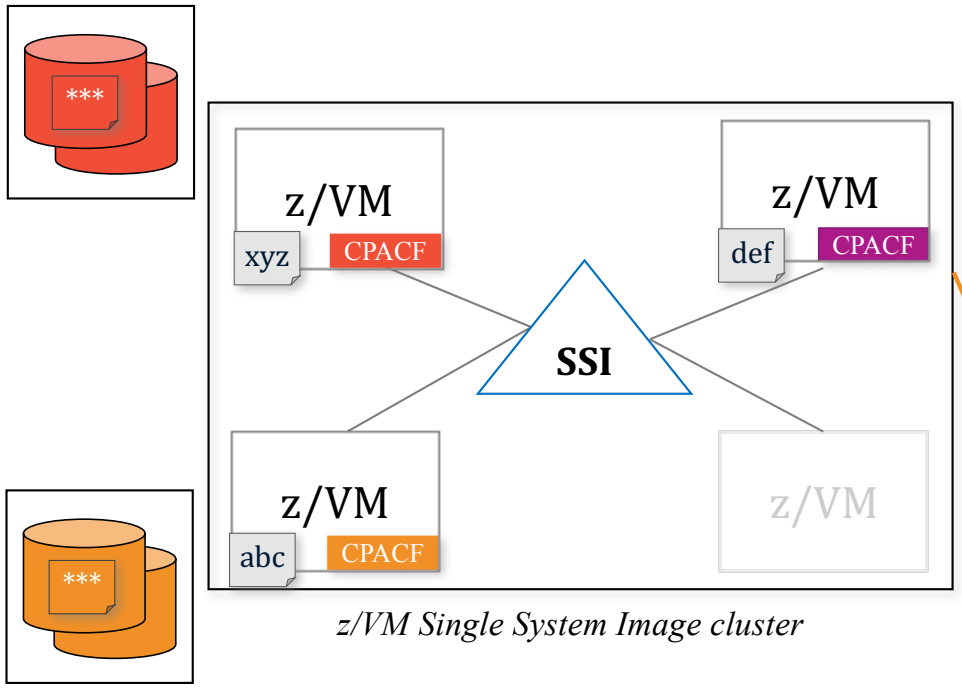
Notes:

- Automatic use of CPACF for symmetric algorithms
- One-line change to enable automatic use of Crypto Express features for acceleration of asymmetric algorithms
- Built on System SSL and ICSFLIB for z/VM

Data Protection // z/VM Encrypted Paging

Protection of data at-rest

z/VM 6.4
PTF for APAR VM65993
(Coming December 2017)



z/VM Single System Image cluster

Legend:
*** - encrypted data
abc - unencrypted data

Encrypted Paging
- **Threat**: access to sensitive data when stored on CP owned disk
- **Solution**: encrypt guest data on page-out.

Notes:
- Paging is not SSI-relevant
- Paging data does not need to survive an IPL
- Ephemeral CPACF protected-key stored in CP (not on disk somewhere)
- AES encryption
- Very low overhead via CPACF

Client Value Proposition:
Protect guest paging data from administrators and/or users with access to volumes

Getting Started with Encrypted Paging

How Do I Get Value?

z/VM Encrypted Paging

1. Starting point: z/VM partition on a z14 with CPACF enabled
2. Select configuration in System Configuration file (can modify it dynamically later, if you change your mind)
3. Generate an ephemeral *n*-bit AES encryption key during IPL process
4. If ENCRYPT PAGING is ON, then pages are encrypted as they move to/from paging volumes.
5. Use monitor records to determine performance impact for workloads

Relevant Hills: SUB-HILLS 1 & 3
Relevant Sponsor User Roles: Data Owner, Security Admin, Auditor
Security Admin Products: z/VM

Using Encrypted Paging for z/VM (1/2)

- ***new*** ENCRYPT Statement in System Configuration file
  - `ENCRYPT PAGING ON ALGORITHM AES256`

- ***new*** QUERY/SET ENCRYPT
  - `SET ENCRYPT PAGING {OFF | ON | REQUIRED}`
  - ALGORITHM selection when first enabled (AES 128, 192, 256)

- **Note**: REQUIRED may cause complications with DR sites
  - System will not IPL on earlier hardware, or if missing CPACF
  - Recommendation: keep a backup System Configuration file for SALIPL emergencies
  - Recommendation: use sysname keywords in System Configuration to specify ENCRYPT by system or node
  - Recommendation: IPL your system with `ENCRYPT PAGING ON <algorithm>`
    - `SET ENCRYPT PAGING REQUIRED` via AUTOLOG1 or via a COMMAND Statement
    - Audit trail demonstrates encryption was never “off.”

Using Encrypted Paging for z/VM (2/2)

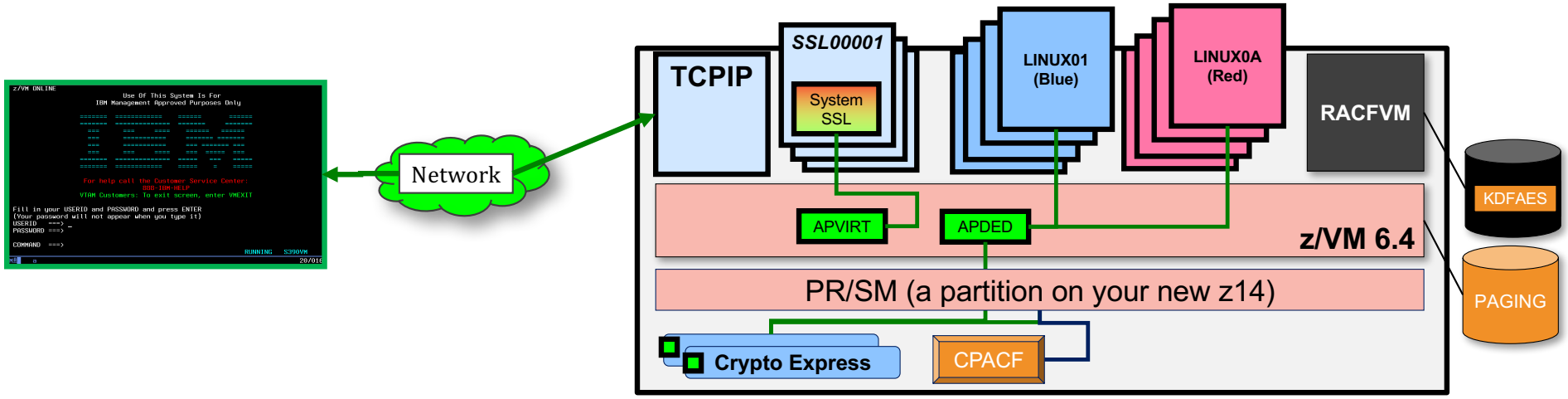
- Auditing with MONITOR Records
  - D1R4 – System Configuration and current status thereof
  - D3R2 – Change record for status (SET ENCRYPT), with userid
  - ***new*** D1R34 – Pages encrypted/decrypted, CPU utilization for encryption

- If moving from ON to OFF, pages will still be decrypted when read into guest memory

- Only way to ensure 100% compliance is to IPL your z/VM system with
  - `ENCRYPT PAGING ON ALGORITHM AES256`

- Auditing with SMF Records
  - Auditing in RACF automatically covers new CP commands, per above
  - Just enable tracking in your VMXEVENT profile

Summary: z/VM and Pervasive Encryption



- Protection for guest operating systems
  - Encryption needs to exist in virtual environments, too!
- Protection of data in flight
  - Modernized software crypto library
  - Crypto Express acceleration for hypervisor traffic
- Protection for data at rest
  - Encrypted Paging as the first step (12/2017)
  - More to follow …
- Simplification and ease of use
  - Security and cryptography should not be an impediment to business

Data at Rest for Linux on z

A discussion of dm-crypt and Protected Key AES

Linux on z Crypto Libraries

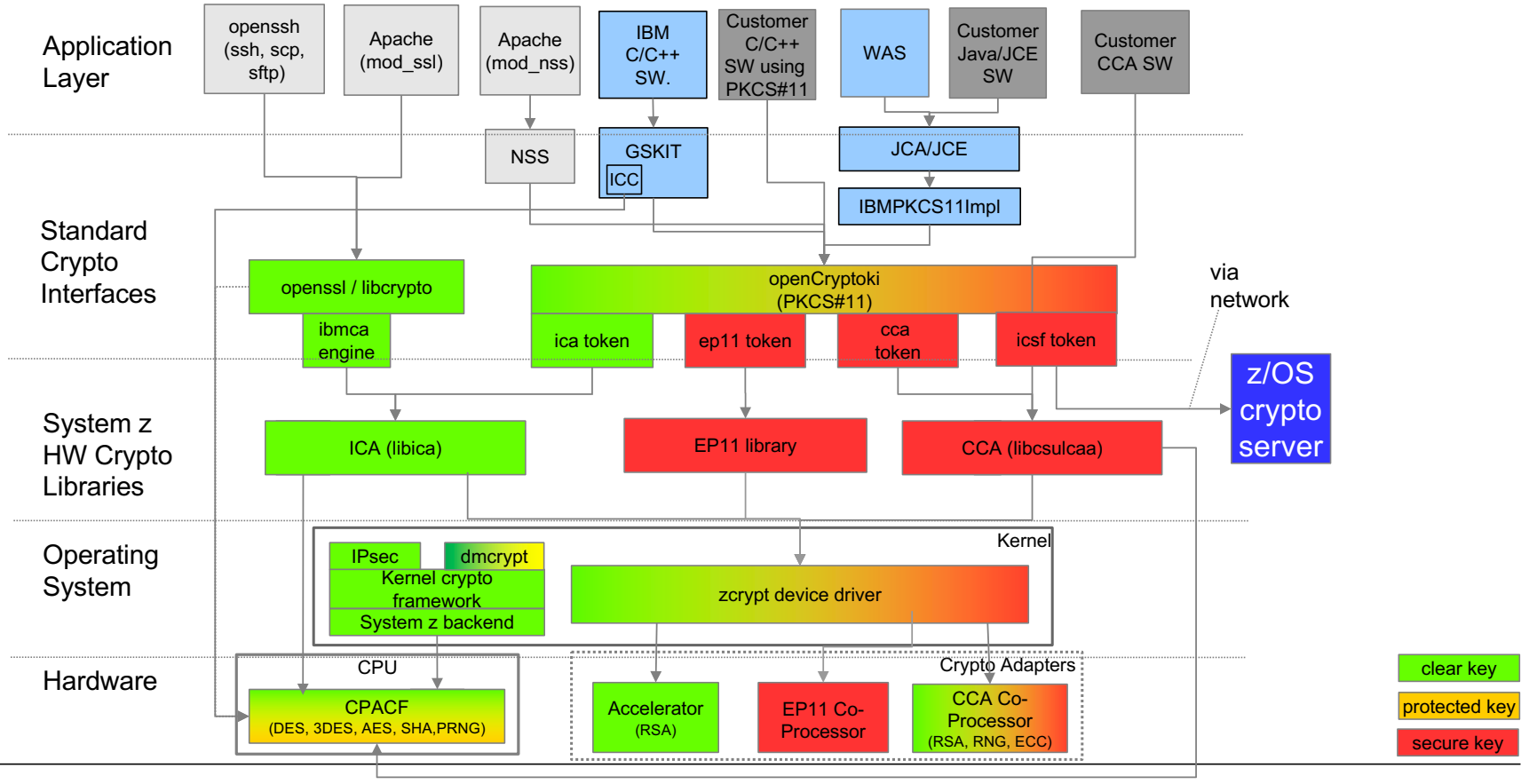
Crypto Libraries supporting CPACF

- libica
  - latest release supports z14 CPACF
- openSSL (libcrypto API)
  - option: configure ibmca engine
  - z14 GCM support via ibmca engine
- openCryptoki (PKCS#11 API)
  - with ica token (calls libica)
- GSKit
  - used by IBM software
  - latest release supports Z14 CPACF
- IBM Java 8 IBMJCE (JCE API)
  - latest release supports Z14 CPACF

Crypto Libraries supporting Crypto Express

- libica
  - clear key RSA
- libcsulcca
  - CCA coprocessor
- openssl with ibmca (libcrypto API)
  - clear key RSA
- openCryptoki (PKCS #11 API)
  - ica token: clear key RSA
  - cca token: CCA coprocessor
  - ep11 token EP11 coprocessor
- GSKit
  - via openCrxptoki using PKCS #11 API
- IBM Java IBMPKCS11Impl (JCE API)
  - via openCryptoki using PKCS #11 API
- IBM Java IBMJCECCA (JCE API)
  - CCA coprocessor

Linux on z Crypto Infrastructure



IBM

Pervasive Encryption and Linux on z

Bringing Pervasive Encryption to Linux involves the following:

1. Pushing crypto usage upstream
   - Kernel enablement is a pain point – HW crypto should not be a burden

2. Extending crypto usage for data in flight and data at rest
   - Find ways to differentiate, even if "Linux is Linux"
   - Transparent crypto usage by kernel, protected-key dm-crypt …

Technical Aspects of Pervasive Encryption for Linux on z

Improved crypto performance
- benefit form accelerated CPACF functions
- exploit improved & new CPACF for zNext
- exploit z13 & z14 SIMD support for asymmetric encryption

Easy crypto consumability
- Linux is Linux, but using z specific HW shall not be an extra burden
- Transparent crypto exploitation:
  - in-kernel crypto (dm-crypt, IPSec)
  - *new*: direct contributions to libcrypto/openSSL library code -> apache, ssh, …
- protected key dm-crypt
  - allows automatic disk access (boot)

Improved security
- A: abundant entropy
  - to generate good and strong keys
  - feed CPACF true random numbers into kernel entropy pool
- B: *unique security enhancement* for dm-crypt:
  - Protected key support
  - requires Crypto Express adapter
- C: Secure Service Container
  - tamper protected and confidential appliance container

Achieving crypto compliance
- compatibility to standard Linux processes
- supporting protected key crypto – resolve security issues involved in storing keys

Pervasive Encryption and Linux on z

Which Kernels and which distros?

1. Kernel level 4.11 (with more to follow over time, as it's all accepted upstream)
   - CPACF TRNG (*hwrng*) in 4.12

2. DeveloperWorks first, then it's picked up by the Linux Distributions for "native" support.

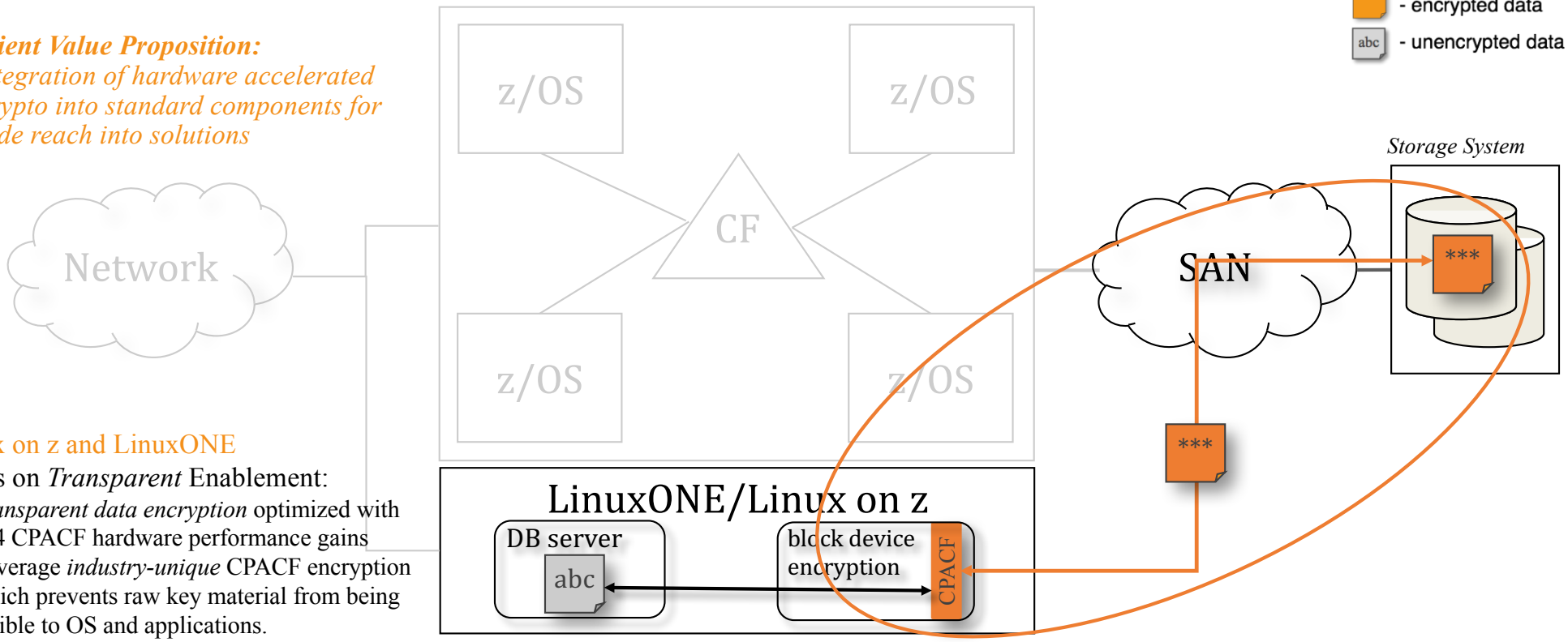
3. Distro to Kernel level matrix (August 2017)
   - **SLES 12 SP2** **v4.4.21**
   - **SLES 11 SP3** **v3.0.76**
   - **RHEL 7.3** **v3.1**
   - **Ubuntu 17.04** **v4.10**

Data Protection // Linux on z File Encryption

Protection of data at-rest

Submitted Upstream

Client Value Proposition:
Integration of hardware accelerated Crypto into standard components for wide reach into solutions



Linux on z and LinuxONE

Focus on *Transparent* Enablement:

- *Transparent data encryption* optimized with z14 CPACF hardware performance gains
- Leverage *industry-unique* CPACF encryption which prevents raw key material from being visible to OS and applications.

Status: dm-crypt enhancements for CPACF protected-key submitted upstream

Pervasive Encryption for Data at Rest

- **dm-crypt**: block device / full volume encryption
  - uses kernel crypto
  - granularity: disk partition / logical volume
  - new protected key option
- **ext4 with encryption** option: file system encryption
  - uses kernel crypto
  - granularity: file, directory, symbolic link
- **Spectrum Scale** (GPFS) with encryption option: file encryption
  - **uses GSKit or Clic**
  - granularity: file
- **NFS v4 with encryption** option: encryption of file transport
  - uses kernel crypto
- **SMB v3.1**: encryption of file transport
  - uses kernel crypto
- **DB2 native encryption**: data base encryption
  - **uses GSKit**

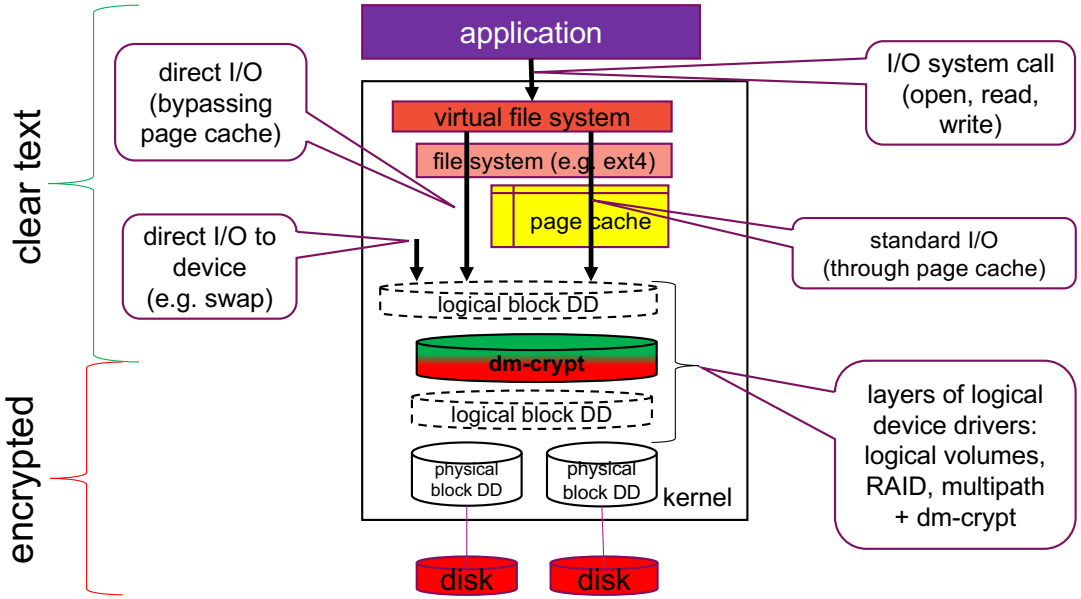
on z14 all crypto becomes faster !

kernel crypto automatically uses CPACF for AES if the module aes_s390 is loaded

GSKit and latest versions of Clic use CPACF for AES

with new Linux kernels, keys generated on z14 will be more secure

Linux File System Stack with dm-crypt



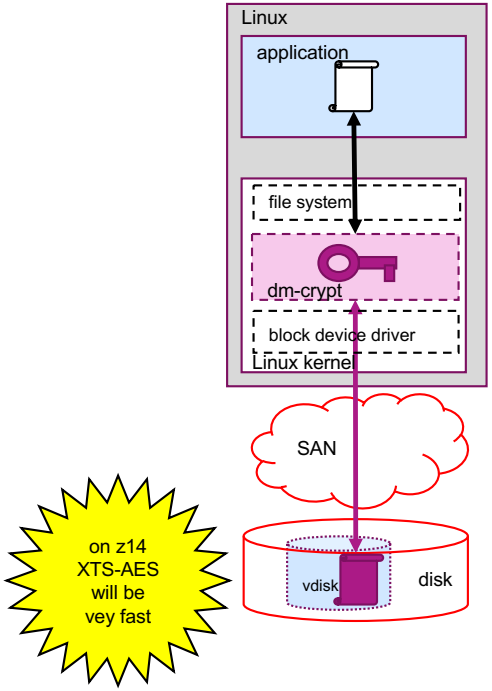
End-to-End Data at Rest Encryption

- The complete I/O path outside the kernel is encrypted: HV, adapters, links, switches, disks

- **dm-crypt**
  - a mechanism for end-to-end data encryption
  - data only appears in the clear in application

- **Linux kernel component that transparently**
  - for all applications
  - for a whole block device (partition or LV)
    - encrypts all data written to disk
    - decrypts all data read from disk

- **Uses LUKS and in-kernel crypto operations**
  - LUKS: encryption keys stored on disk (partition, LV) header
  - LUKS: encryption keys on disk are protected by passphrases
    - passphrases must be provided when disk is “opened”
  - can use IBM Z CPACF for symmetric crypto operations:
    - AES-CBC
    - XTS-AES (recommended)



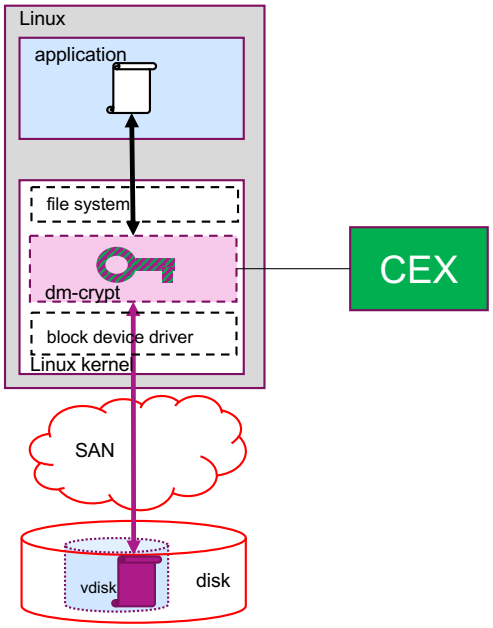
E2E Data at Rest Encryption with Protected Keys

- **Protected keys?**
  - never stored in plain text in OS memory
  - wrapped by system key accessible to CPU only
  - ephemeral since system key recycled with every IPL
  - extend lifespan of protected keys with Crypto Express adapters
  - functionally similar to secure keys but much faster
    - implemented on CPU (CPACF)
    - no I/O required

- **New kernel support for protected key**
  - module to support managing protected keys
    - transform secure key into protected key
  - module to support PAES cipher (protected key AES)
    - takes a secure key and caches the associated protected key

- **dm-crypt**
  - can use **PAES** cipher to protect data with XTS-AES

- **New tools** to manage volume encrypted using **PAES**
  - support of LUKS format (work in progress)



IBM

The PAES in-kernel Cipher

- upstream since kernel 4.11

- **paes** module implements protected key AES ciphers:
  - ecb(paes)
  - cbc(paes)
  - xts(paes)
  - ctr(paes)

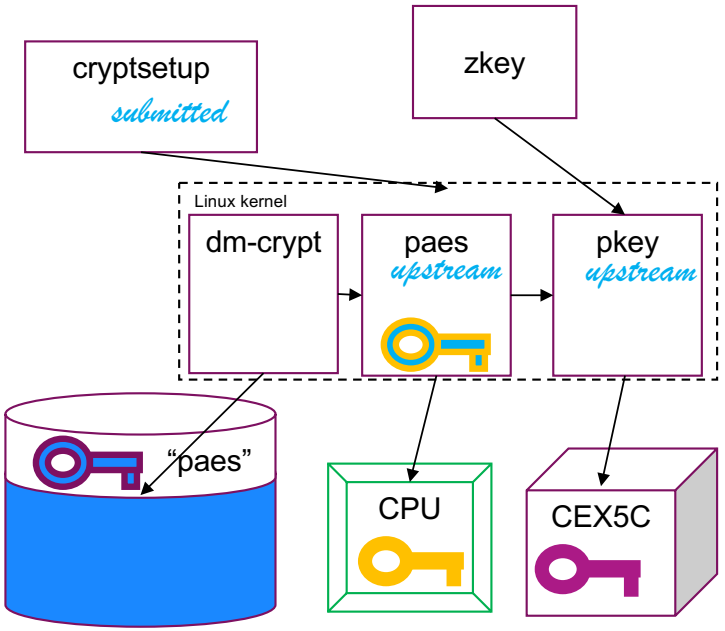
- requires the **pkey** module as prereq

- **paes** ciphers take CCA AES secure keys as key arguments
  - transforms secure key into protected key
  - caches protected key (into encryption context aka transform)
  - uses protected key for cryptographic operations

LUKS/dm-crypt with Protected Keys

Components

- new *pkey* kernel module for protected key management
  - generate secure key
  - transform secure key into protected key
- new *paes* kernel module to perform protected key encryption-decryption
  - introduces paes cipher
- *dm-crypt* kernel module (unchanged)
- extended dm-crypt management tool *cryptsetup*
  - recognizes paes cipher
  - stores secure key into LUKS header
- new *zkey* tool to
  - generate and manage secure keys
  - re-encipher secure key



Secure Key Handling: the zkey Tool

- new tool to be added to s390tools
- requires pkey module
- generate, validate, re-encipher secure AES keys to be transformed into protected keys
  - generate
    - generates file with AES secure key (AESDATA)
    - random key or from clear key
    - single key (for CBC) or two keys (for XTS)
    - size of secure keys: 64 bytes (single key), 128 bytes (XTS key) regardles of AES key size
  - validate
    - checks if input file contains valid AES secure key
    - if yes displays key attributes
  - re-encipher
    - support master key change on CryptoExpress adapter
    - transforms a valid secure key wrapped by an current (or old) HSM master key into a secure key wrapped by a new (or current) master key
    - requires installation of CCA package from
      http://www-03.ibm.com/security/cryptocards/pciecc2/lonzsoftware.shtml

Using the PAES with dm-crypt – Plain Format

- the plain dm-crypt format does not have a header describing the disk encryption: no formatting required

once

- generate a file with a secure key

```
# zkey generate seckey2.bin -xts
```

  – requires access to CEX[4|5|6]C adapter

with every boot

- open block device as device mapper volume

```
# cryptsetup open --type plain --key-file seckey2.bin \\
--key-size 1024 --cipher paes-xts-plain64 \\
/dev/dasdd1 plain_enc
```

  – new virtual device mapper volume /dev/mapper/plain_enc will be created
  – requires access to CEX[5|6]C adapter

BAU

- use new device mapper volume
  – (only once) create file system:

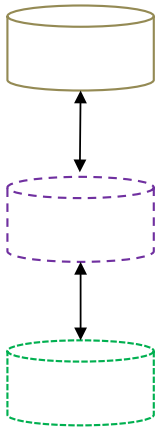
```
# mkfs.ext4 /dev/mapper/plain_enc
```

  – mount:

```
# mount /dev/mapper/plain_enc /mount_point
```

  – access:

```
# echo "hello world" > /mount_point/myfile
```



Best Practices with (Protected Key) dm-crypt

- use /etc/cryptsetup
  - to configure automated opening of volumes

- use dm-crypt volumes as LVM physical volumes
  - allows transparent data migration

- for production use
  - Back up the dm-crypt superblock
    - to deal with superblock corruption
  - Utilize back-up adapters with same master keys
    - to deal with HSM loss
    - Alternately:
      - generate secure key from clear key in clear room environment
      - and store clear key in safe

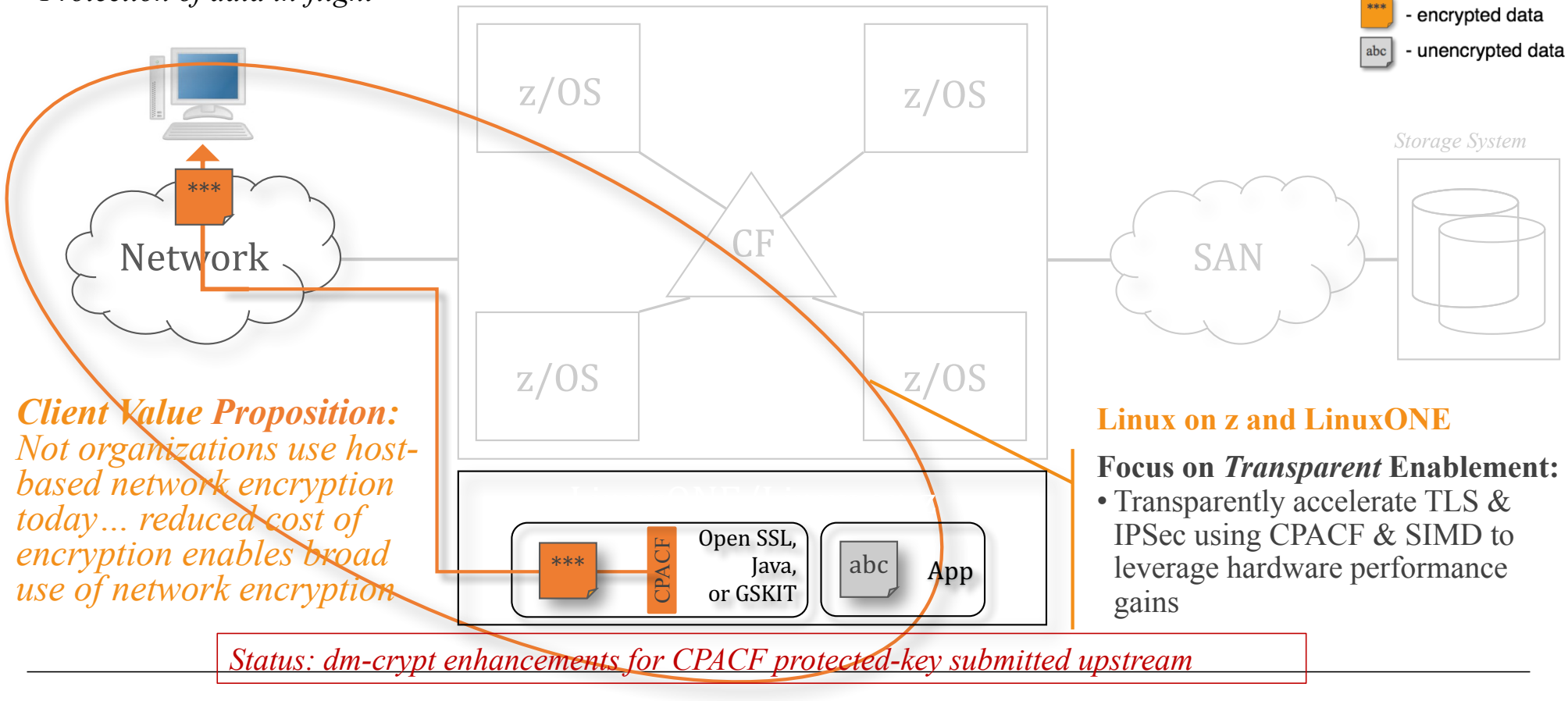
Data in Flight for Linux on z

Remember, no virtual machine is an island.

Data Protection // Linux on z Network Security

Protection of data in-flight

Submitted Upstream



Client Value Proposition:
Not organizations use host-based network encryption today… reduced cost of encryption enables broad use of network encryption

Linux on z and LinuxONE

Focus on *Transparent* Enablement:

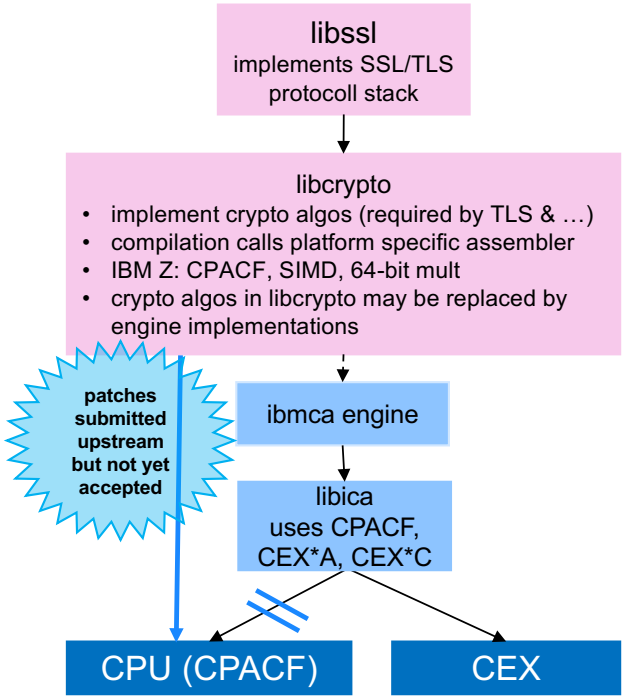
- Transparently accelerate TLS & IPSec using CPACF & SIMD to leverage hardware performance gains

Status: dm-crypt enhancements for CPACF protected-key submitted upstream

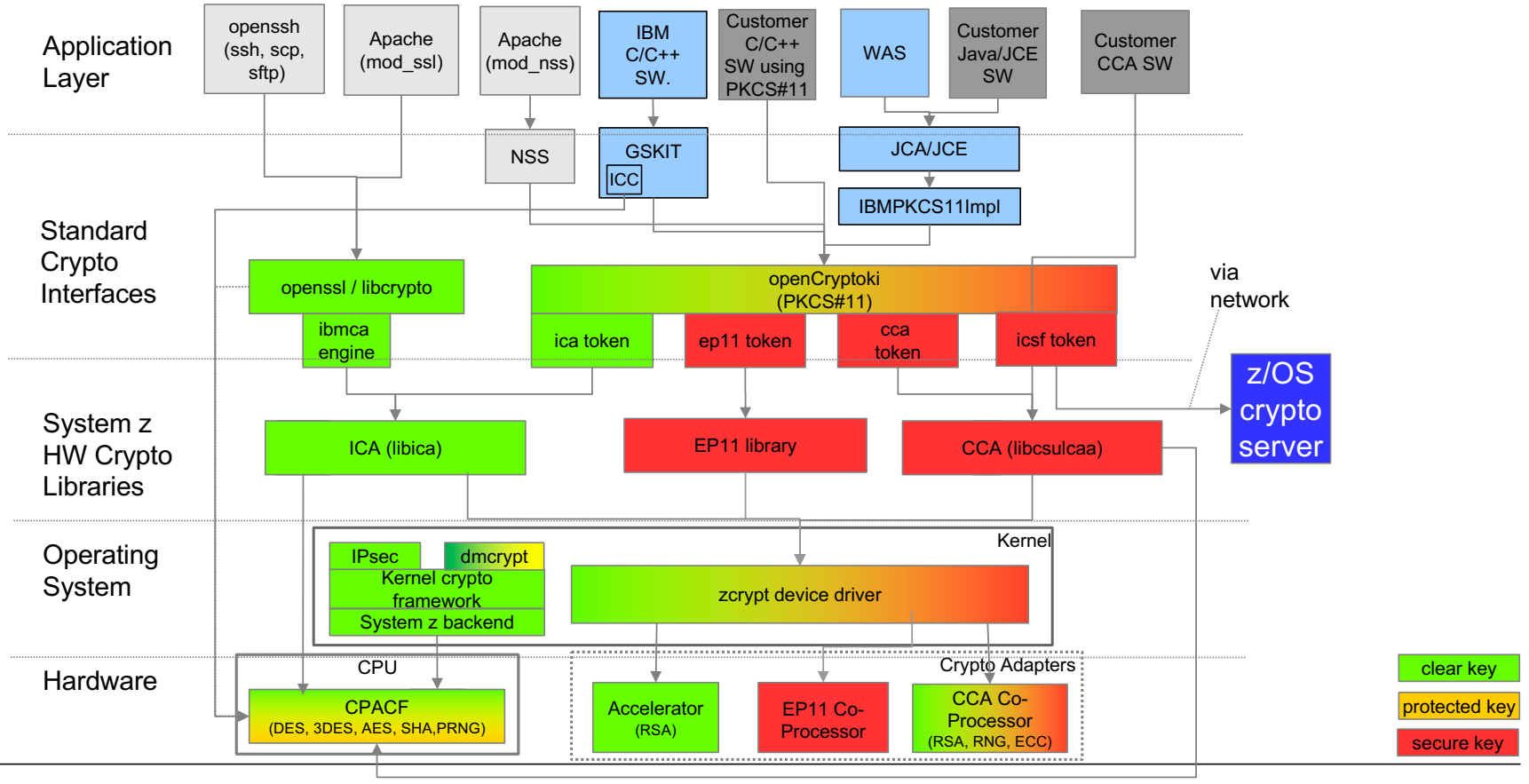
The new Linux on z openSSL Strategy

Original Linux on z strategy
- put Z specific code in the ibmca engine (only)
- pro: all Z specific user space cyrpto in libica
- cons: engines must be configured

New Strategy
- all CPU dependent code (SIMD, CPACF) in libcrypto
- CryptoExpress dependent code in ibmca
- **no config needed for**
  - hashes (SHA1, SHA2, poly1305)
  - AES (ECB, CBC, OFB, CFB, XTS, CTR, GCM, CCM)
  - chacha20,
  - RSA, & ECC acceleration via SIMD arithmetic
- ibmca engine config needed for
  - offload/acceleration of RSA, DH, DSA, ECC, (3DES) via CryptoExpress adapters
  - configure engine to not support AES or hashes



Linux on z Systems Crypto Infrastructure (Refresher)



Pervasive Encryption: Data in Flight

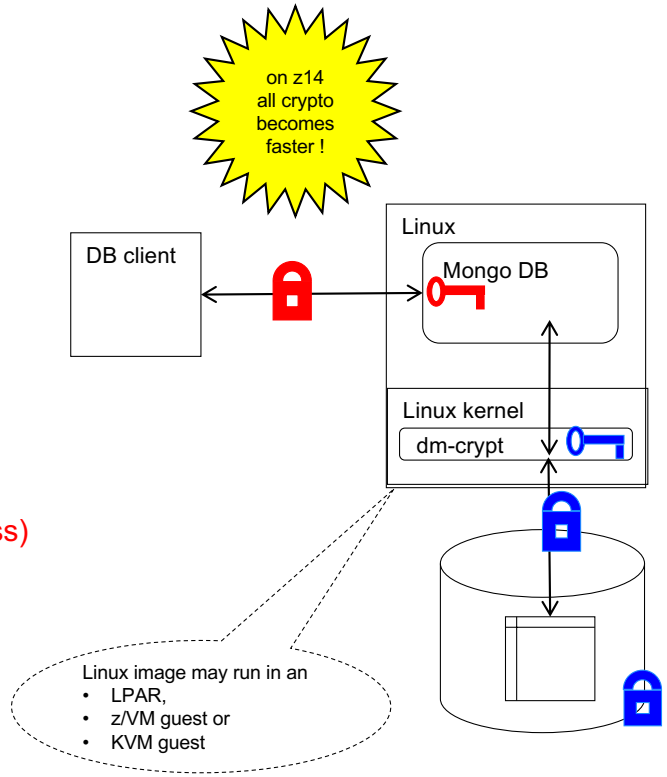
- openSSL and libcrypto
  - de-facto standard TLS & crypto libraries
  - used by many open source projects (including Apache, node.js, MongoDB)
  - exploitation of IBM Z CPACF and SIMD code by libcrypto (w/o ibmca engine)
  - focus on TLS 1.2 ciphers
  - no Z-specific configuration required
- IPsec
  - bulk encryption and authentication implemented by kernel crypto
  - transparantly uses CPACF
- GSKit
  - IBM C library for TLS and crypto
  - e.g. used by IBM HTTP Server (IHS)
  - uses IBM Z CPACF *& SIMD code submitted to openSSL*
- Java 8 / JCE
  - exploitation of IBM Z CPACF and SIMD code

Use-Cases for Linux on z

What can you do with all this stuff?

Use Case 1: Mongo DB Server

As a **Linux system administrator**, I **want to run a no-SQL DB service** using an existing open source DB **where all data in flight and at rest is transparently encrypted**



data in flight:

- encrypted connection by DB server (-> openSSL)
- encrypted Linux sessions via ssh (-> openSSL)
- *transparent* usage of CPACF by openSSL
- symmetric (CPACF) and asymmetric encryption (SIMD or CryptoExpress)

data at rest

- **end-to-end volume level encryption by Linux kernel (dm-crypt)**
- ***transparent* usage of CPACF by Linux kernel**
- **protected key option possible**

secure manner of key generation

- CPACF true random numbers are fed in kernel entropy pool

Use Case 2: Mobile Server Farm in a Trusted HV

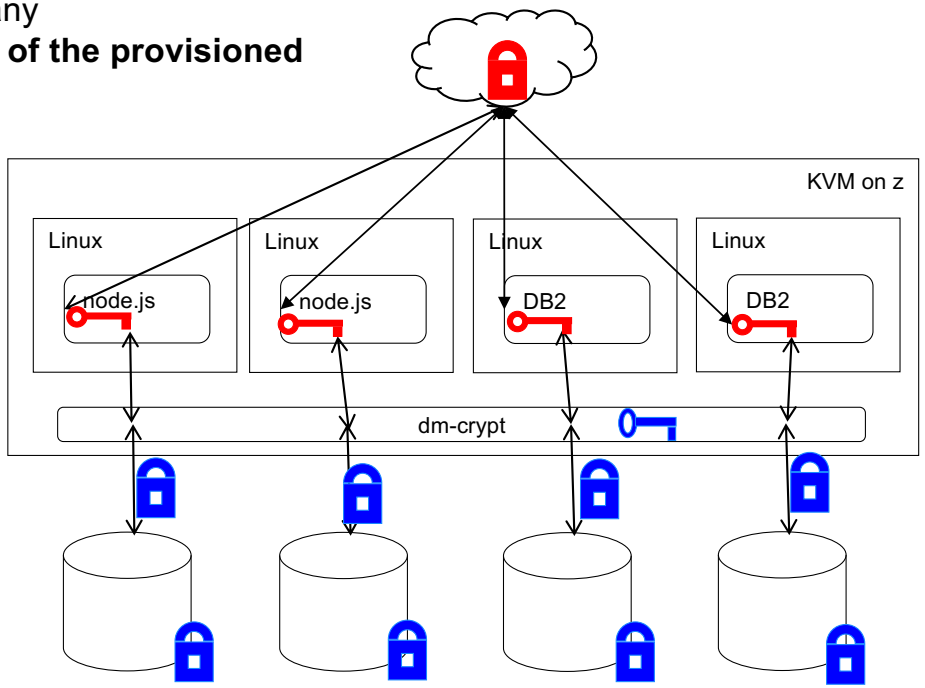
As **an operator of a data center** for my company
I want to **host a server farm** such that **all data of the provisioned servers shall be transparently encrypted**

data at rest:

- end to end encryption of all real volumes by KVM
- transparent usage of CPACF via kernel and dm-crypt
- protected key option possible

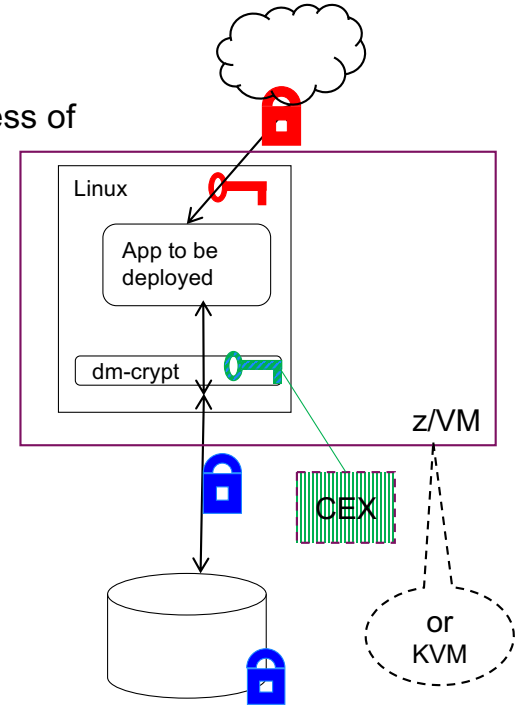
data in flight:

- per guest NW encryption in node.js or apache or DB2
- transparent usage of CPACF and SIMD via openSSL or GSKit



Use Case 3: PaaS for Sensitive Data

As a provider for PaaS I want to address
- customers with sensitive data (HR, Health, insurances, ...) and
- provide systems where all data at rest is transparently encrypted regardless of the storage location such that the encryption key cannot be stolen.



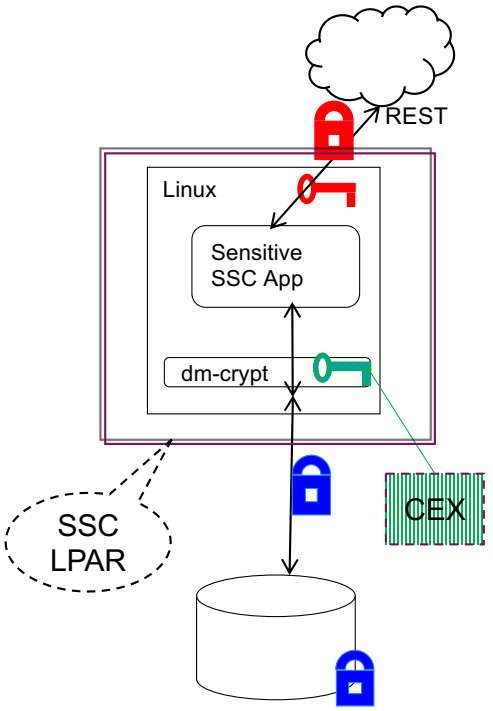
data at rest

- end-to-end volume level encryption by Linux kernel (dm-crypt)
- *transparent* usage of protected key CPACF by Linux kernel
- *unique security and usability enhancement*
  - no clear key in memory
    - use protected keys
    - requires CryptoExpress adapter
  - autonomous boot
- clear text data in volumes can only be accessed by the system that created the data

Use Case 4: Secure Service Container

As **a service provider** I want to be able host ultra sensitive appliances (e.g. block chain nodes) as a black box that cannot be inspected by my operator team

- let customer & IBM build SSC image
- install signed & partially encrypted SSC image in SCC LPAR
- no access of from SE/HMC into SSC LPAR
- restricted secure connectivity through REST APIs
- all SSC Data E2E encrypted (dm-crypt)
- option: protected key dm-crypt provides additional layer of security



Questions?

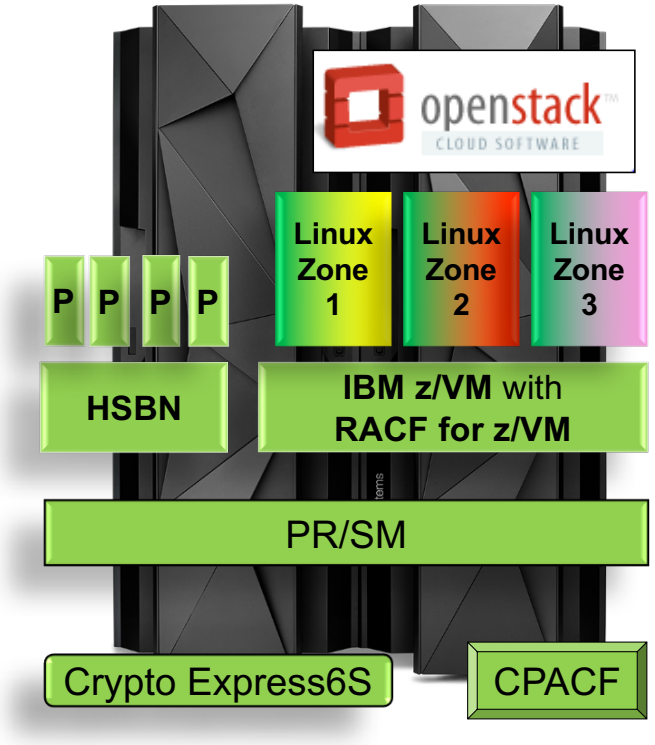
This is where I pause for breath and let you do the talking.

Summary

- **The z14 brings Pervasive Encryption to mainframe workloads – including, yes, z/VM and Linux on z**
  - It's not merely doing what we've always done, but faster
  - Goals are ease-of-use and ubiquity of encryption for partitions, hypervisors & guests
  - Remember, it's not "Linux on z is more secure," it's **"Linux is more secure on Z."** Especially with a z14.

- **Aim is to deliver encryption everywhere, with ease of use, at scale**
  - A lot of moving parts
  - Customers will need to enable the parts that work best for their security requirements … and their performance goals

- **IBM will work with distribution partners to bring new Linux capabilities into open source projects for future distro releases**



For More Information …

- **IBM z14 Technical Guide:**
  http://www.redbooks.ibm.com/redpieces/abstracts/sg248451.html?Open
- **IBM Z Hardware Crypto Synopsis:**
  https://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100810
- **IBM Z Crypto Education Community**:
- https://www.ibm.com/developerworks/community/groups/community/crypto
- **z/VM Security:**
  http://www.vm.ibm.com/security
- **Linux on z Security:**
  https://www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/security.html

Contact Information:

CISSP®

Brian W. Hugenbruch
IBM Z Security for Virtualization & Cloud
bwhugen at us dot ibm dot com
@Bwhugen