Pervasive Encryption for Linux on z Systems and LinuxONE

Reinhard Buendgen -- buendgen@de.ibm.com

Crypto Architect for Linux on z

Trademarks

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

CICS*	Global Business Services*	MQ*	SPSS*	XIV*	z/VSE*
Cognos*	IBM*	Parallel Sysplex*	System Storage*	zEnterprise*	
DataStage*	IBM (logo)*	QualityStage	System x*	z/OS*	
DB2*	InfoSphere	Rational*	Tivoli*	z Systems*	
GDPS	Maximo*	Smarter Cities	WebSphere*	z/VM*	

* Registered trademarks of IBM Corporation

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.

PERVASIVE ENCRYPTION FOR IBM Z

The Value of Data …

Today data is one of the most valuable assets of many companies.

In particular sensitive data must be protected against unauthorized access to avoid

- losing customer trust
- losing competitive advantages
- being subject to fines and regression claims

Data encryption is the most effective way to protect data outside your system be it in flight or at rest.

But encrypting data is not easy

- requires the introduction of new policies
- complicates data management
- requires to securely manage keys
- costs computing resources

Here is a Dream …

what if you could just encrypt all data in-flight and at-rest

- at no cost
- w/o changing applications
- w/o changing data management
- by pushing a single button

Well, that will remain to be a dream.

But with **pervasive encryption** we want to make a large step in that direction.

Pervasive Encryption with IBM Z

Technical Foundation – Linux on z related

IBM z14 -- Designed for Pervasive Encryption

- CPACF – Dramatic advance in bulk symmetric encryption performance
- Crypto Express6S– Doubling of asymmetric encryption performance for TLS handshakes
- Coupling Facility – E2E encryption

z/OS -- New approach to encryption of in-flight and at-rest data

- z/OS data set encryption – Transparent encryption of data at-rest
- z/OS CF encryption –Transparent end-to-end encryption of CF data

Linux on z and LinuxONE -- Full Power of Linux Ecosystem combined with IBM z14 Capabilities

- dm-crypt – Transparent volume encryption using industry unique CPACF protected-keys
- Network Security – Enterprise scale encryption and handshakes using z14 CPACF and SIMD
- Secure Service Container – Automatic protection of data and code for virtual appliances

z/VM – New: Encrypted paging support

PERVASIVE ENCRYPTION FOR LINUX ON Z SYSTEMS AND LINUXONE

USE CASES

Use Case 1: Mongo DB Server

As user 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:

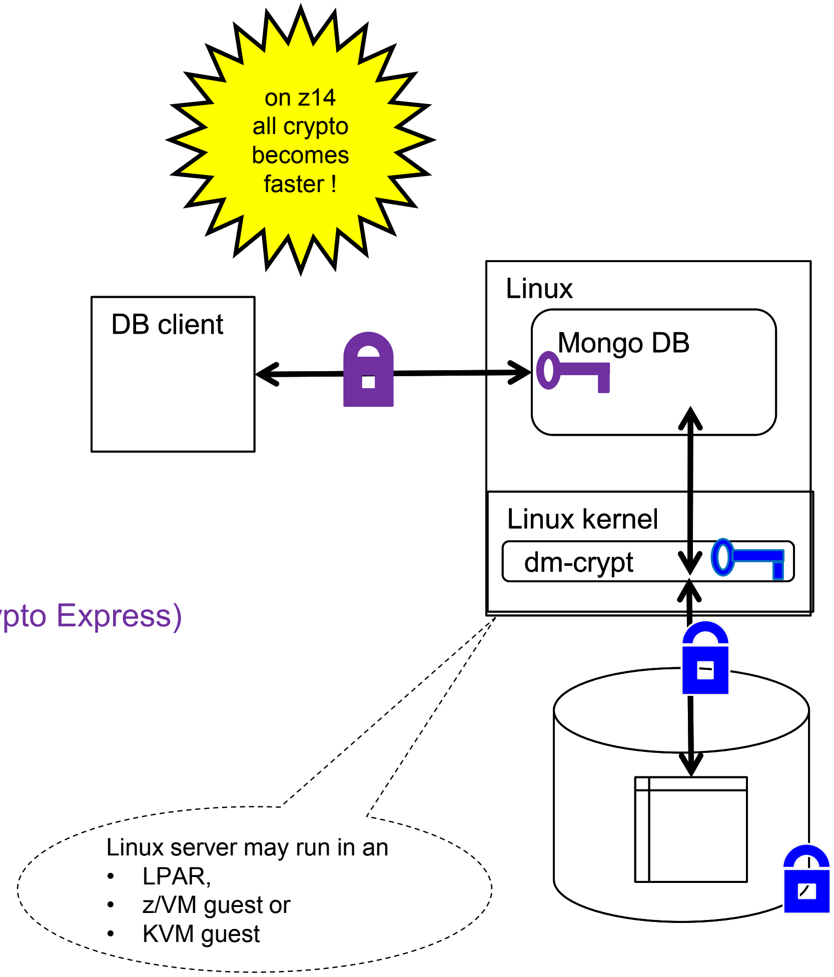
- encrypted connection by DB server (-> openSSL)
- encrypted Linux sessions via ssh (-> openSSL)
- *transparent* usage of CPACF by openSSL
- symmetric (CPACF) and asymmetric encryption (SIMD or Crypto Express)

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

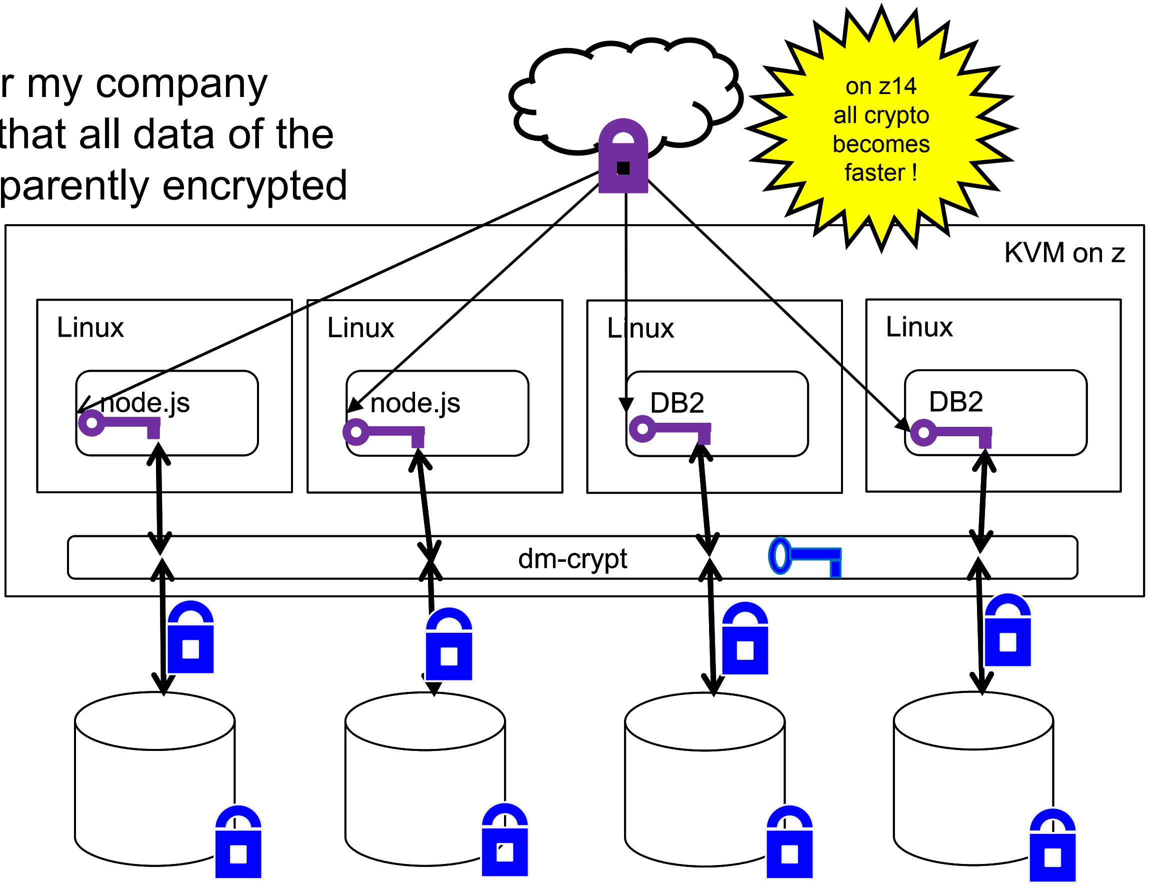
As an operator of a data center for my company
I want to host a server farm such that all data of the provisioned servers shall be transparently encrypted

data in flight:

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

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



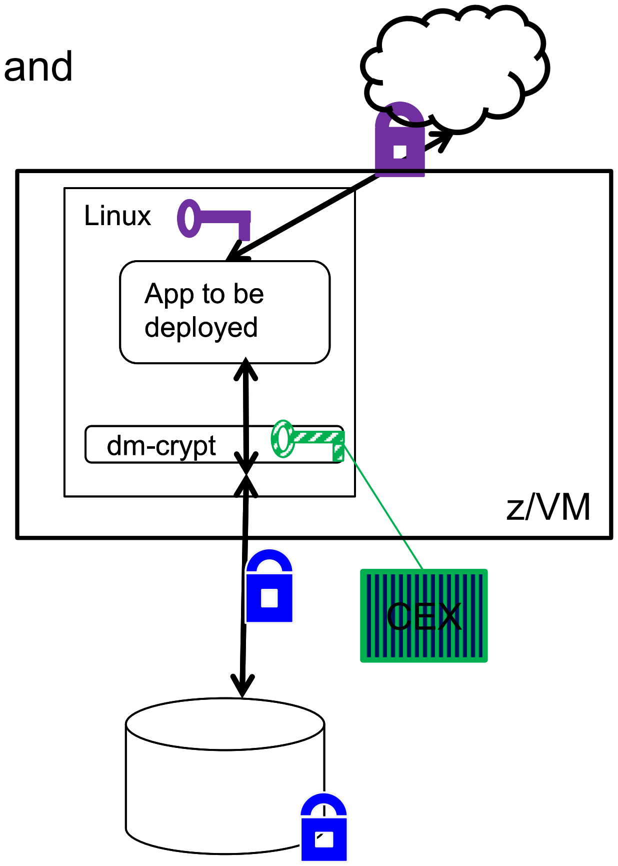
Use Case 3: PaaS for Sensitive Data

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

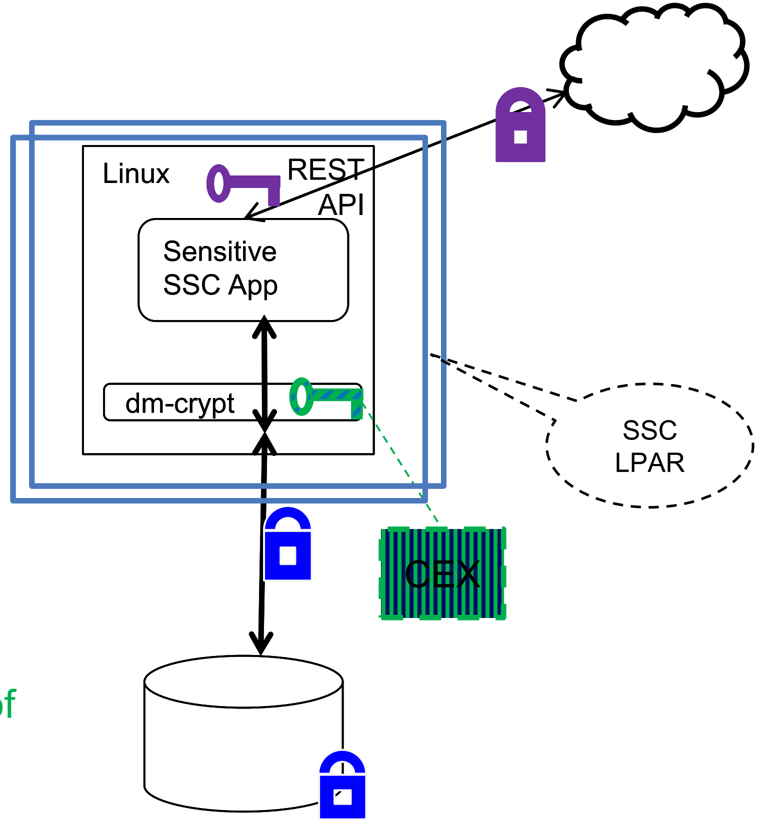
- end-to-end volume level encryption by Linux kernel (dm-crypt)
- *transparent* usage of protected key CPACF by Linux kernel
- *unique security and usability enhancement*
  - no clear key in memory
    - use protected keys
    - requires Crypto Express adapter
  - autonomous boot
- clear text data in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



Use Case 4: Secure Service Container (SSC)

As service provider I want to be able host ultra sensitive appliances (e.g. blockchain 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 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



PERVASIVE ENCRYPTION FOR LINUX ON Z SYSTEMS AND LINUXONE

TECHNICAL CONTENTS

Technical Aspects of Pervasive Encryption for Linux on z Systems and LinuxONE

Improved crypto performance

- benefit from accelerated CPACF functions
- exploit improved & new z14 CPACF functions
- exploit z13 & z14 SIMD support

Easy crypto consumability

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

Improved security

- abundant entropy
  - to generate good and strong keys
  - feed CPACF true random numbers into kernel entropy pool
- *unique security enhancement* for dm-crypt:
  - no plain text key in memory
    - use protected keys
    - requires Crypto Express adapter
- Secure Service Container
  - tamper protected and confidential appliance container

PERVASIVE ENCRYPTION FOR LINUX ON Z SYSTEMS AND LINUXONE

TECHNICAL CONTENTS: DATA IN FLIGHT

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 and 1.3 ciphers
  - no IBM Z specific configuration required

openssl patches submitted upstream but not yet accepted

- **IPsec**
  - bulk encryption and authentication implemented by kernel crypto
  - transparently uses CPACF
- **GSKit**
  - IBM C library for TLS and crypto
  - e.g. used by IBM HTTP Server (IHS)
  - uses IBM Z CPACF
  - release 8.0.50.82 will use new z14 CPACF instructions
- **Java 8 / JCE**
  - exploitation of IBM Z CPACF and SIMD code
  - Java 8 service refresh 5 will use z14 CPACF instructions

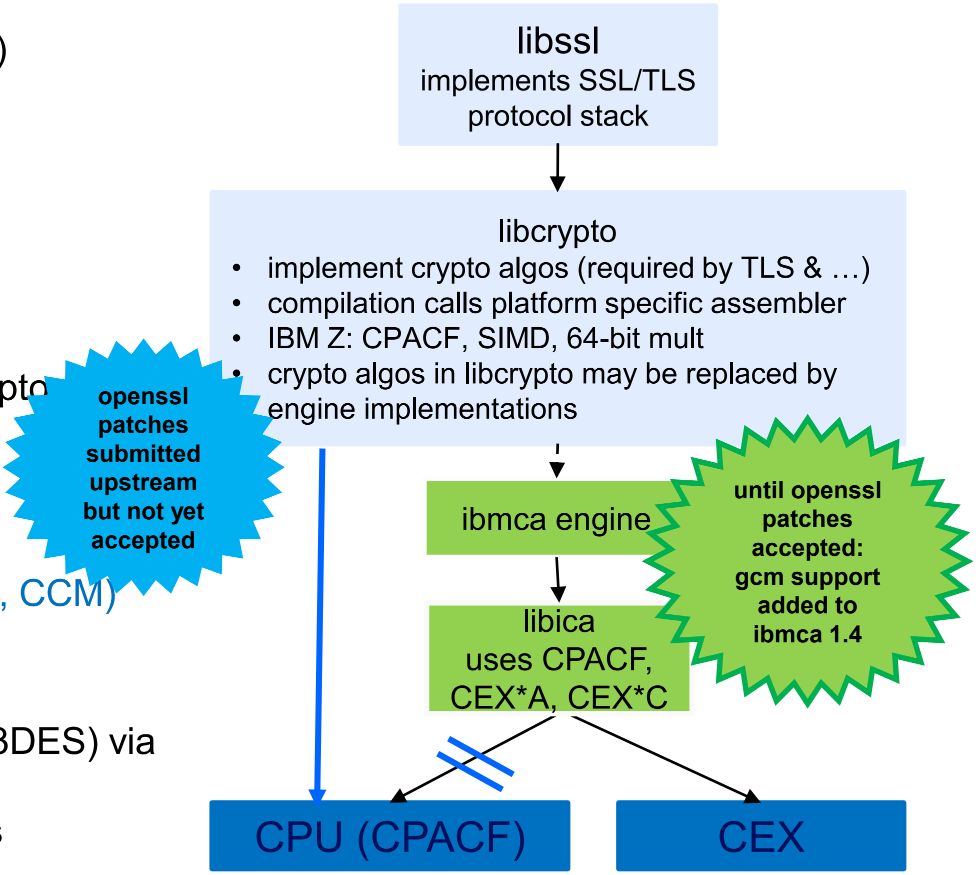
The new Linux on z openSSL Strategy

Original Linux on z strategy

- put IBM Z specific code in the ibmca engine (only)
- pro:
  - all IBM Z specific user space cyrpto in libica
  - IBM maintains ibmca engine
- cons: engines must be configured

New Strategy

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



libssl
implements SSL/TLS protocol stack
libcrypto
• implement crypto algos (required by TLS & …)
• compilation calls platform specific assembler
• IBM Z: CPACF, SIMD, 64-bit mult
• crypto algos in libcrypto may be replaced by engine implementations
openssl patches submitted upstream but not yet accepted
ibmca engine
libica uses CPACF, CEX*A, CEX*C
until openssl patches accepted: gcm support added to ibmca 1.4
CPU (CPACF)
CEX

PERVASIVE ENCRYPTION FOR LINUX ON Z SYSTEMS AND LINUXONE

TECHNICAL CONTENTS: DATA AT REST

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
- ext4fs 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

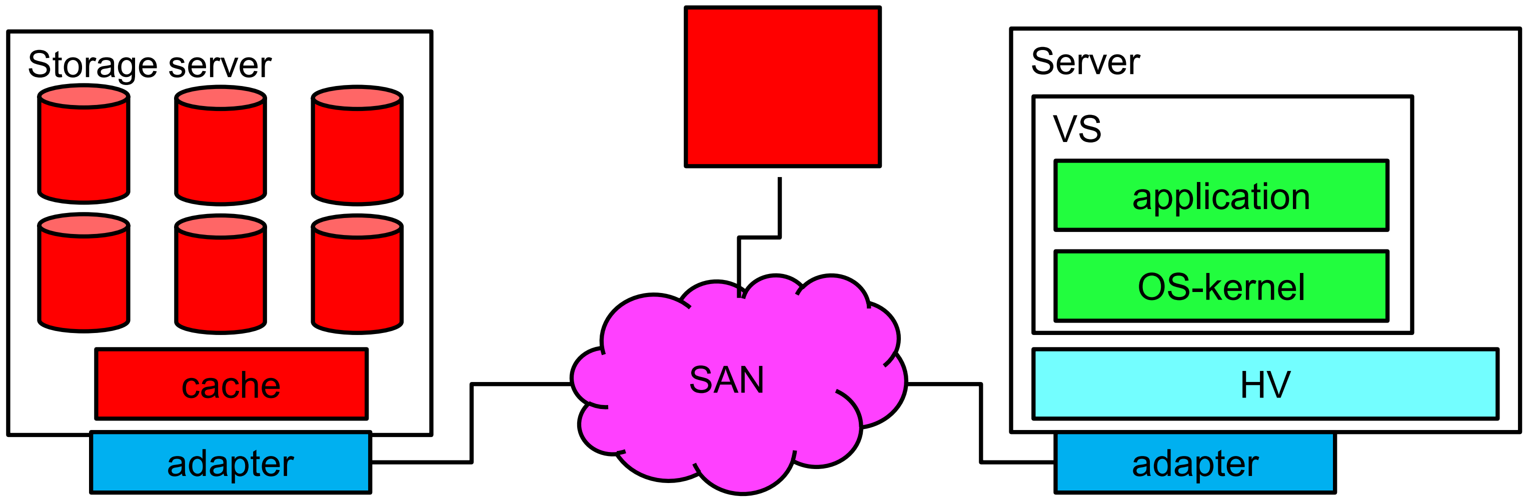
on z14 all crypto becomes faster !

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

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

GSKit and latest versions of Clic use CPACF for AES

Data at Rest Encryption Considerations



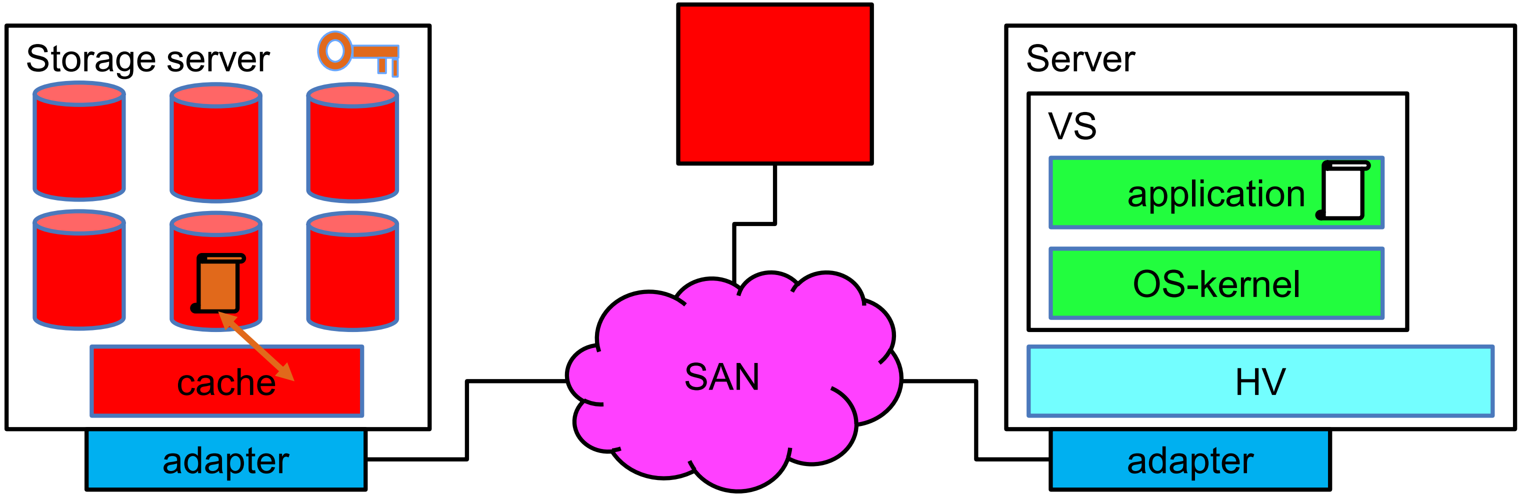
Attack points

- storage server
- SAN
- Server / HV / VS
  - insider / outsider

Questions

- Where is data decrypted/encrypted?
- Who generates keys?
- Who owns (can access) the keys?
- Backups? Data migration?

Data Encryption on Storage Subsystem



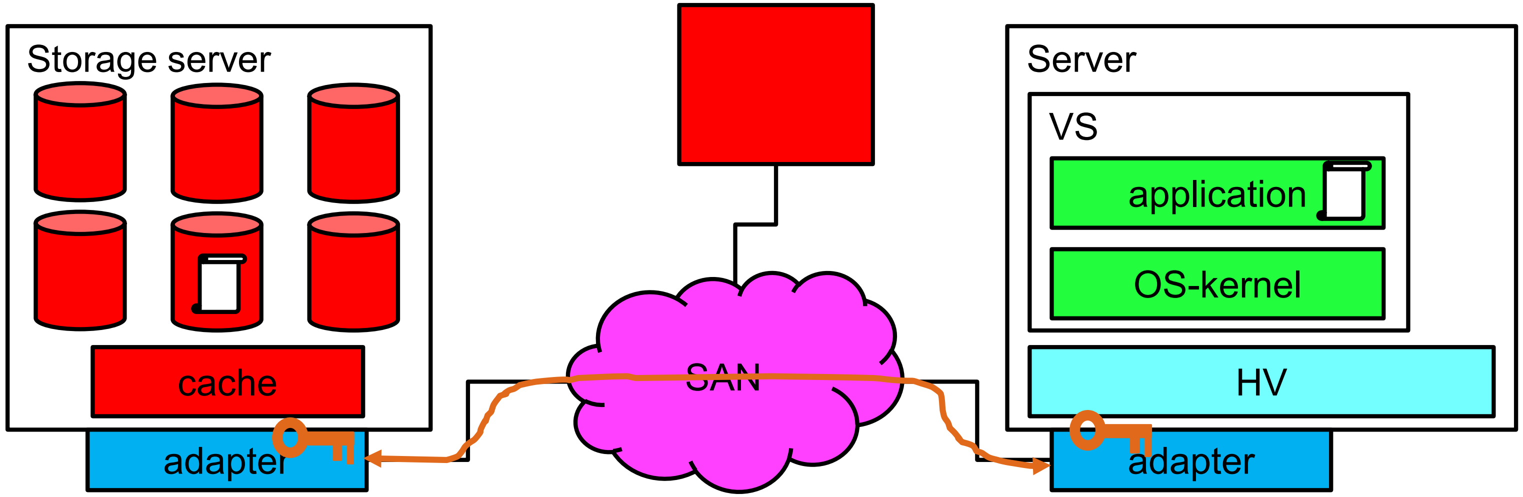
Attack points

- storage server -- (secured)
- SAN – not secure
- Server / HV – not secure
- VS insider / outsider – not secure

Questions

- Where is data decrypted/encrypted? storage server
- Who generates keys? storage/OS
- Who owns (can access) the keys? storage/OS admin

End-to-End SAN/Network Encryption



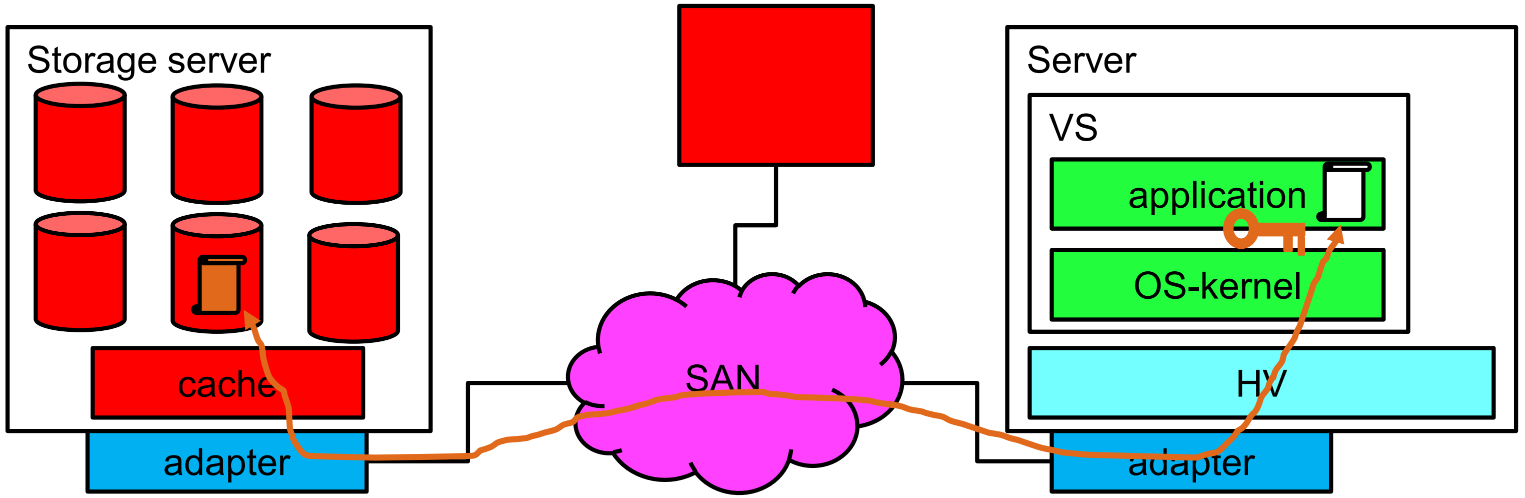
Attack points

- storage server -- not secured
- SAN -- secured
- Server / HV -- not secured
- VS insider / outsider – not secured

Questions

- Where is data decrypted/encrypted? adapter
- Who generates keys? system admin
- Who owns (can access) the keys? system admins

End-to-End Data Encryption



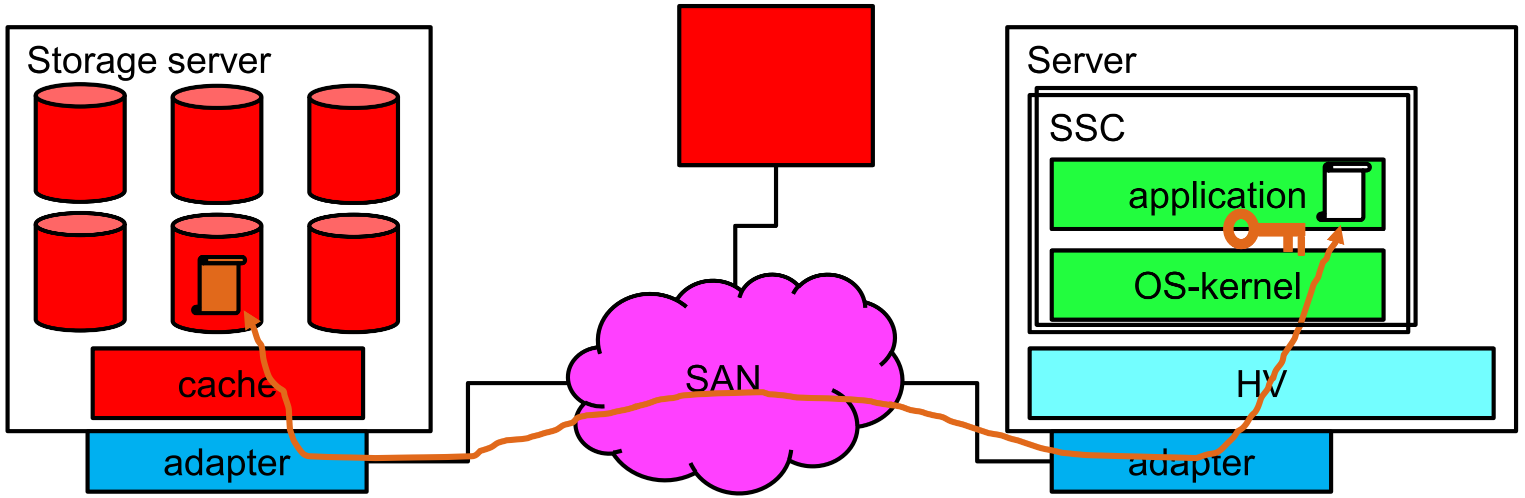
Attack points

- storage server -- secured
- SAN -- secured
- Server / HV – (secured)
- VS insider / outsider -- not secured

Questions

- Where is data decrypted/encrypted? application or kernel
- Who generates keys? app or OS admin
- Who owns (can access) the keys? app or OS admin

End-to-End Data Encryption from SSC



Attack points

- storage server -- secured
- SAN -- secured
- Server / HV – secured
- VS insider / outsider -- hardened

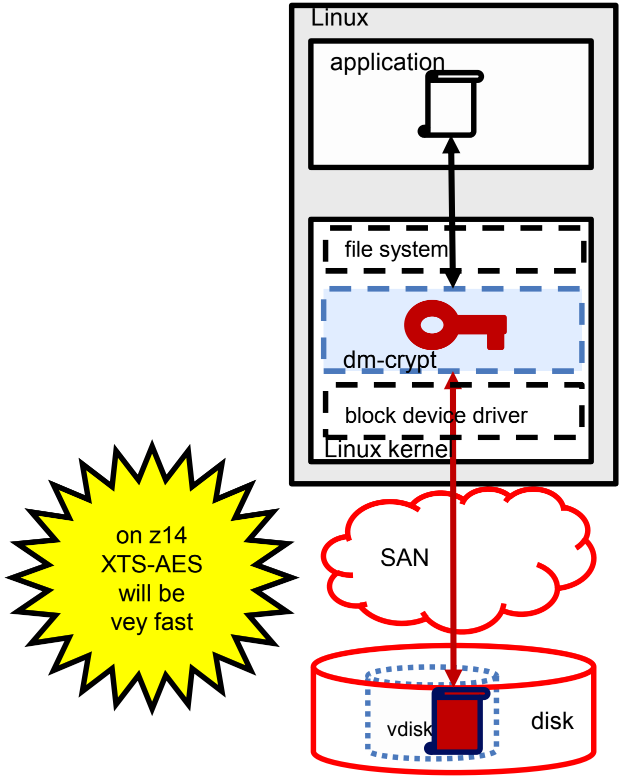
Questions

- Where is data decrypted/encrypted? application or kernel
- Who generates keys? app or OS admin
- Who owns (can access) the keys? app or OS admin

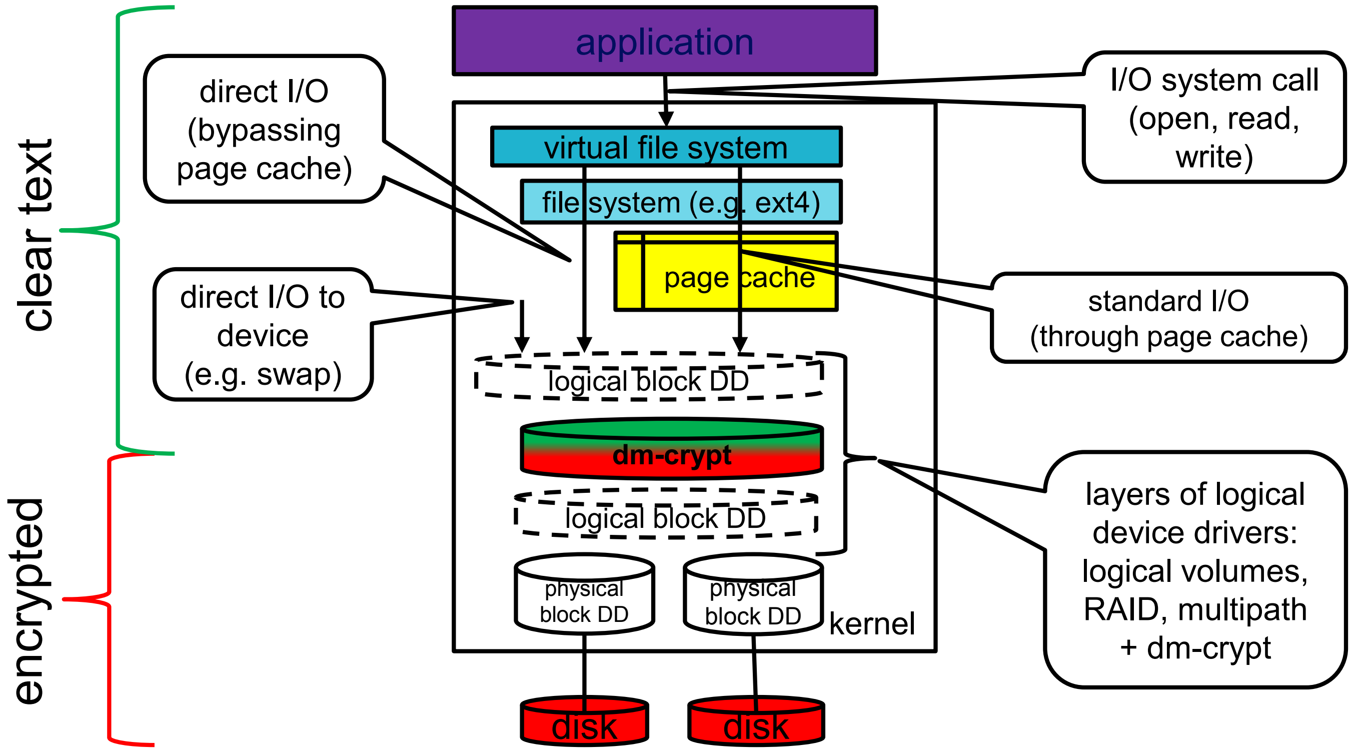
DM-CRYPT & LUKS

End-to-End Data at Rest Encryption with dm-crypt

- **E2E data 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
- **How it works:**
  - uses in kernel-crypto
    - can use IBM Z CPACF Crypto:
      - AES-CBC
      - XTS-AES (recommended)
  - encrypted volumes must be opened before usage
    - opening provides encryption key to kernel
    - establishes virtual volume in /dev/mapper



Linux File System Stack with dm-crypt



How to Set-up a dm-crypt LUKS Volume

once

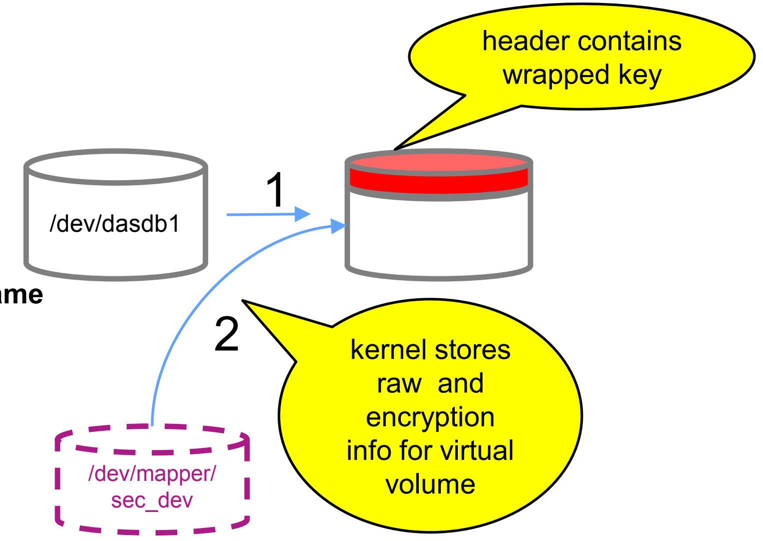
- **1: format “raw” volume as dm-crypt volume**
  - cryptsetup luksFormat …
    - cipher, key length, hash, passphrase
  - writes dm-crypt header to disk

with every boot

- **2: open dm-crypt volume and assign it a virtual volume name**
  - cryptsetup luksOpen …
    - dm-crypt volume, virtual volume, password
  - creates virtual volume in /dev/mapper
  - can be automated with /etc/crypttab

BAU

- **3: use virtual volume**
  - mkfs (or mkswap)
  - mount (or swapon)
  - any kind of standard I/O
  - do **not** use raw volume directly

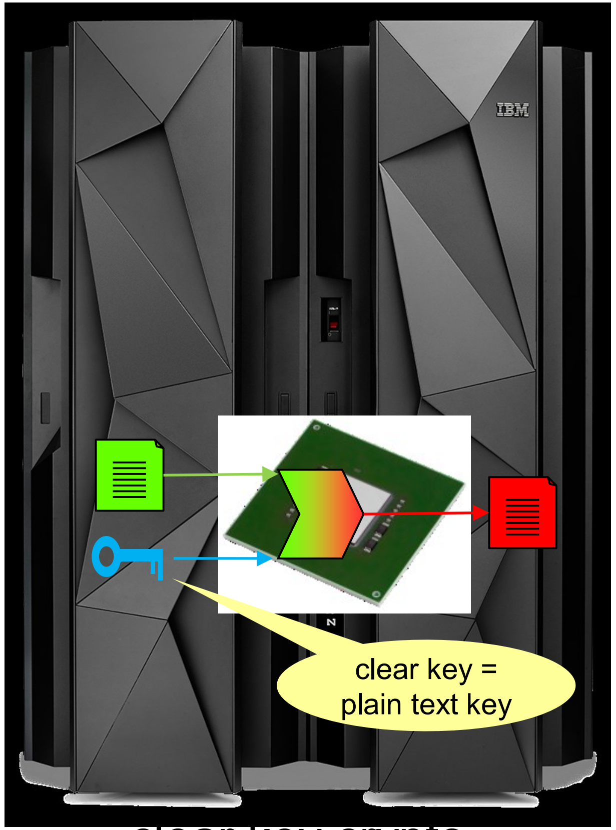

header contains wrapped key
/dev/dasdb1
1
2
kernel stores raw and encryption info for virtual volume
/dev/mapper/ sec_dev

kernel translates I/O to virtual volume into I/O to raw volume and performs decryption/encryption

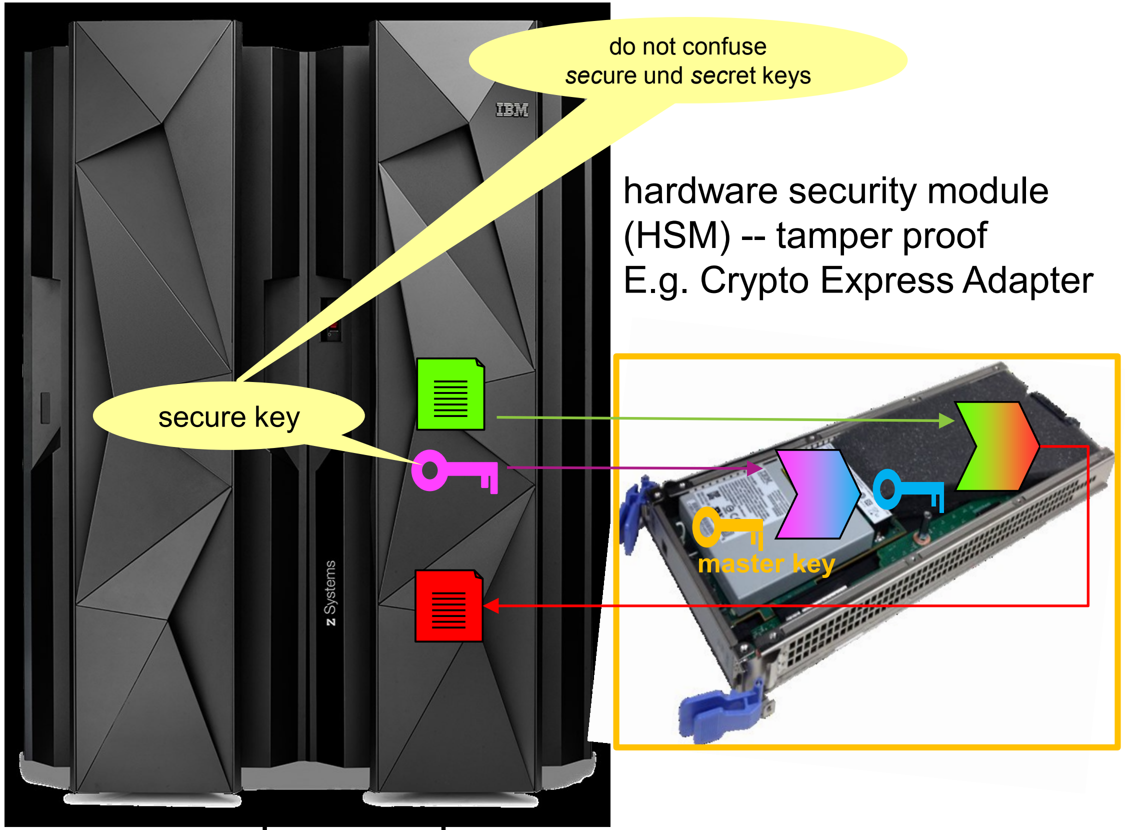
PERVASIVE ENCRYPTION FOR LINUX ON Z SYSTEMS AND LINUXONE

SUPPORTING CPACF PROTECTED KEY CRYPTO

Clear Key vs. Secure Key Cryptography



clear key crypto

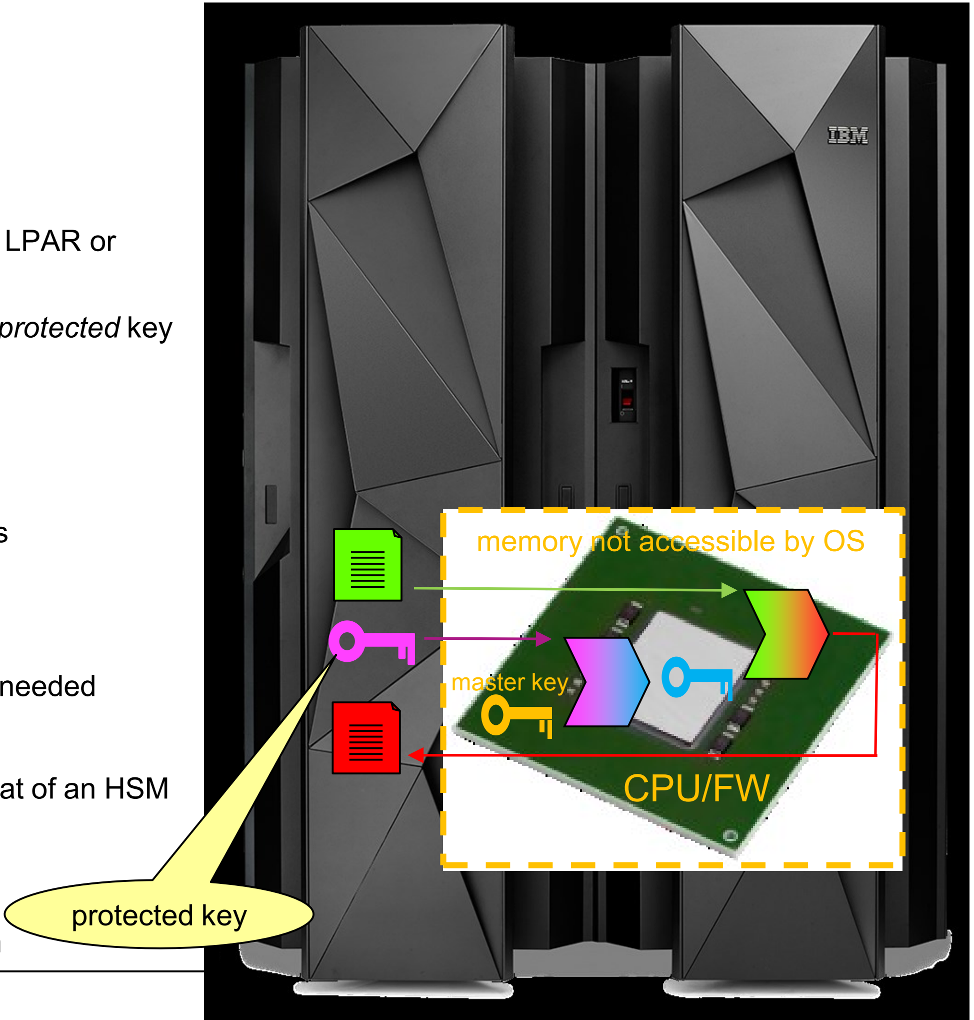


hardware security module (HSM) -- tamper proof
E.g. Crypto Express Adapter

secure key crypto

CPACF Protected Keys

- IBM Z function of the CPU (CPACF)
- each virtual server (LPAR or guest) has a *hidden* master key
  - hidden master key is not accessible from operating system in LPAR or guest
- a key wrapped by the hidden LPAR/guest master key is called a *protected* key
- protected key tokens can be generated
  - from clear keys (insecure)
  - from secure keys using CEX Adapter (secure)
- IBM Z CPU can compute symmetric encryption for protected keys
  - pro
    - no clear keys in operating system memory
    - fast, encryption/decryption at CPU/CPACF speed, no I/O needed
  - cons
    - „hiding“ of master key not as good and tamper proof as that of an HSM
    - not certified as HSM
    - the LPAR/guest master key is not persistent
      - need to store (secure) key to derive protected key from



Managing Protected Keys: the pkey Kernel Module

- **upstream since kernel version 4.11**
- **activate with**

```
# modprobe pkey
```

- **implements misc device: `/dev/pkey`**
- **provides functions (IOCTLs) to**

– `PKEY_GENSEC`:	generate a random CCA secure key
– `PKEY_CLR2SEC`:	generate a CCA secure key from a clear key
– `PKEY_SEC2PROT`:	generate a protected key from a CCA secure key
– `PKEY_FINDCARD`:	find an adapter and domain associated with a given secure key
– `PKEY_SECK2PROTK`:	first `PKEY_FINDCARD` then `PKEY_SEC2PROT`

  – `PKEY_GENSEC`, `PKEY_CLR2SEC`, `PKEY_SEC2PROT` have target adapter&domain as well as key type as arguments

- **Secure keys are CCA secure keys of type AESDATA**

Secure Key Handling: the zkey Tool

- new tool in s390tools 1.39
- 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) regardless 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 Crypto Express adapter
    - transforms a valid secure key wrapped by a 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

The PAES in-kernel Cipher

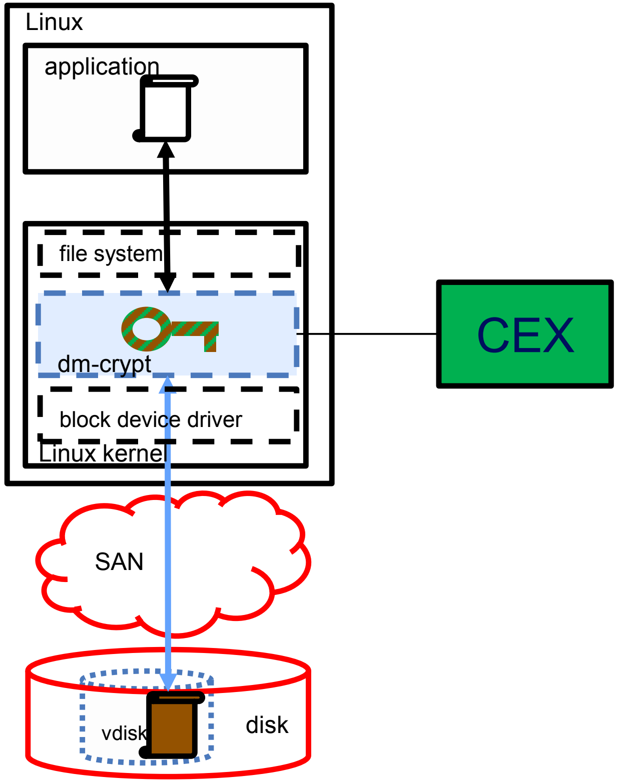
- **upstream since kernel 4.11**
- **paes module implements protected key AES ciphers:**
  - ecb(paes)
  - cbc(paes)
  - xts(paes)
  - ctr(paes)
- **requires the pkey module as prereq**
- **paes ciphers take CCA AES secure keys as key arguments**
  - transforms secure key into protected key
  - caches protected key (into encryption context aka transform)
  - uses protected key for cryptographic operations

PERVASIVE ENCRYPTION FOR LINUX ON Z SYSTEMS AND LINUXONE

SUPPORTING CPACF PROTECTED KEY CRYPTO FOR DM-CRYPT

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
  - volatile since system key recycled with every IPL
  - extend “life span of protected keys” with the help of 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)



Using the PAES with dm-crypt – Plain Format

once

with every boot

BAU

- **the plain dm-crypt format does not have a header describing the disk encryption: no formatting required**
- **generate a file with a secure key**

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

  – requires access to CEX[5|6]C adapter
- **open block device as device mapper volume**

```
# cryptsetup open --type plain --key-file seckey.bin \\
   --key-size 1024 --cipher paes-xts-plain64/dev/dasdb1 \\
   plain_enc
```

  – new virtual device mapper volume /dev/mapper/plain_enc will be created
  – requires access to CEX[5|6]C adapter
- **use new device mapper volume**
  – (only once) create file system:

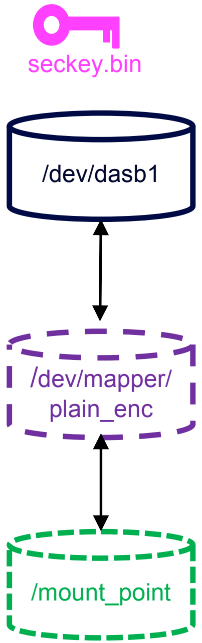
```
   # mkfs.ext4 /dev/mapper/plain_enc
```

  – mount:

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

  – access:

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



Using the PAES with dm-crypt – LUKS Format

once

0: insert new kernel modules during boot

- requires access to CEX5C or CEX6C adapter

once

1: format raw volume as dm-crypt volume

- cryptsetup luksFormat …
  - cipher, key length, hash, password
  - use "paes" instead of "aes"
- writes dm-crypt header to disk

with every boot

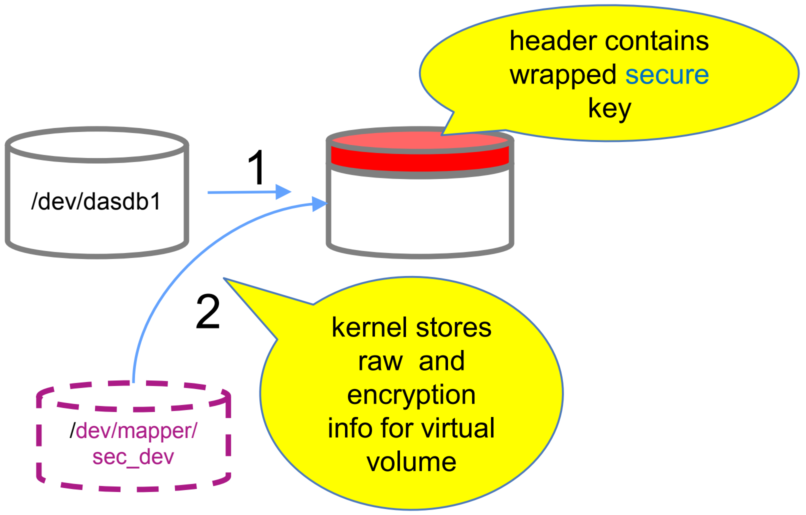
2: open dm-crypt volume and assign it a virtual volume name

- cryptsetup luksOpen …
  - dm-crypt volume, virtual volume, passphrase (needs not be protected)
- creates virtual volume in /dev/mapper

BAU

3: use virtual volume

- mkfs (or mkswap)
- mount (or swapon)
- any kind of standard I/O
- do not use raw volmue directly


header contains wrapped secure key
/dev/dasdb1
1
2
/dev/mapper/ sec_dev
kernel stores raw and encryption info for virtual volume

kernel translates I/O to virtual volume into I/O to raw volume and performs decryption/encryption

modification of cryptsetup submitted upstream but not yet accepted

BEST PRACTICES WITH ENCRYPTED VOLUMES

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**
  - to deal with key loss (plain formant)
    - backup secure key
  - to deal with LUKS header corruption
    - back up dm-crypt header
  - to deal with HSM loss
    - make sure you have a back-up adapter with same master key as primary adapter
    - or put the smart cards with the master key parts into a set of safes.

/etc/crypttab

- configuration file to describe how dm-crypt volumes are to be opened
- will be read and interpreted before /etc/fstab
- format
  - each line describes a dm-crypt volume:
    - <dm-crypt volume> <path of block-device> <password file>|`none` [options]
      - for the plain format, the password file contains a key
    - options may be set to describe volatile swap and tmp volumes
  - example lines:

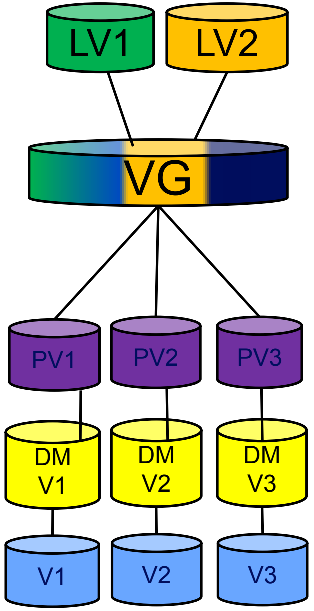
```
luks_vol /dev/dasdb1 sec_dev /root/PWs/sec_dev.pw

sec_swap /dev/dasdb2 sec_swap none cipher=aes-xts-plain64,size=512,hash=sha512,swap

plain_vol /dev/dasdd1 /root/seckey.bin plain,cipher=paes-xts-plain64,hash=plain, \\
  size=1024
```

Using dm-crypt Volumes as LVM Physical Volumes

Use virtual dm-crypt device as input to vgcreate



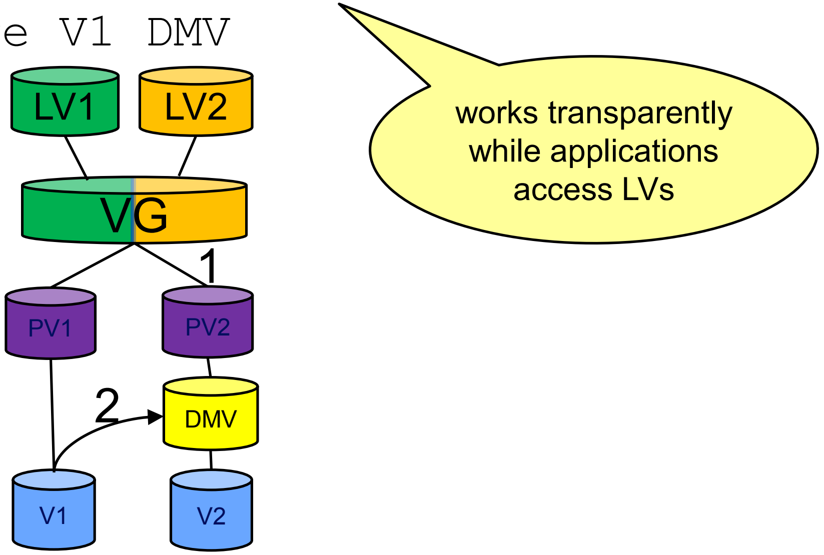
Migrate data from unencrypted volume to dm-crypt volume

– 1: add dm-crypt based physical volume to volume group:

```
# vgextend VG PV2
```

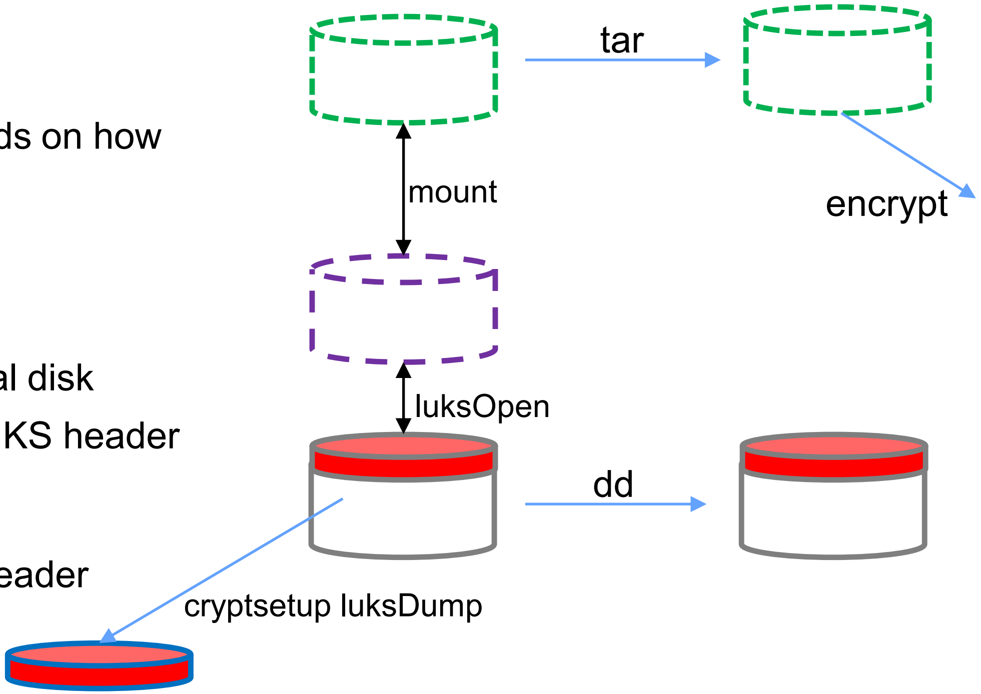
– 2: migrate data from V1 to DMV:

```
# pvmove V1 DMV
```



Back-ups of Encrypted Data

- **Back-ups at file system level**
  - e.g. with tar
  - back-up is plain text encryption depends on how back up is stored
- **Raw disk image**
  - e.g. dd from /dev/dasdb1
  - back-up is encrypted exactly as original disk
  - LUKS: self-contained as it contains LUKS header
- **Always a good idea:**
  - LUKS: make a back-up of the LUKS header
    - protects against header corruption
  - plain format: always back-up key file



Secure Service Containers

Secure Service Container (SSC)

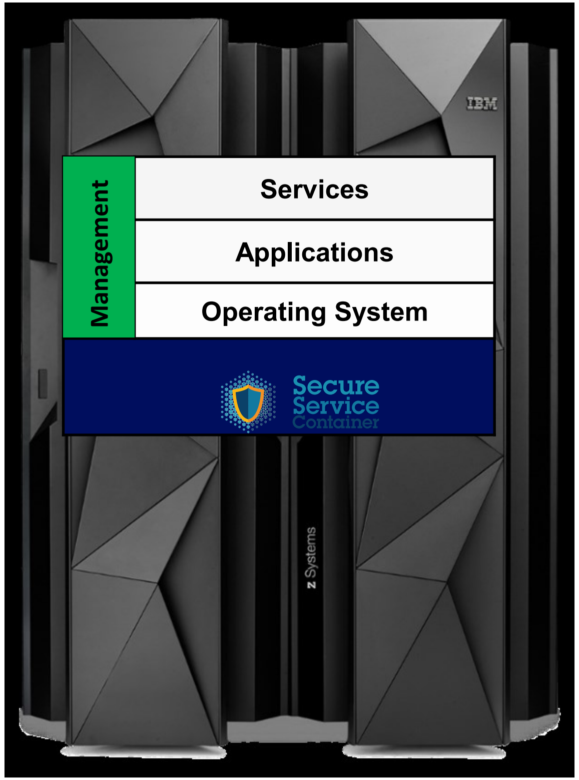
The Base Infrastructure to Host and Build Software Appliances

Simplicity

- Provides simplified mechanism for fast deployment and management of packaged solutions
- Management provided via remote APIs (RESTful) and web interfaces
- Enables appliances to be delivered via distribution channels

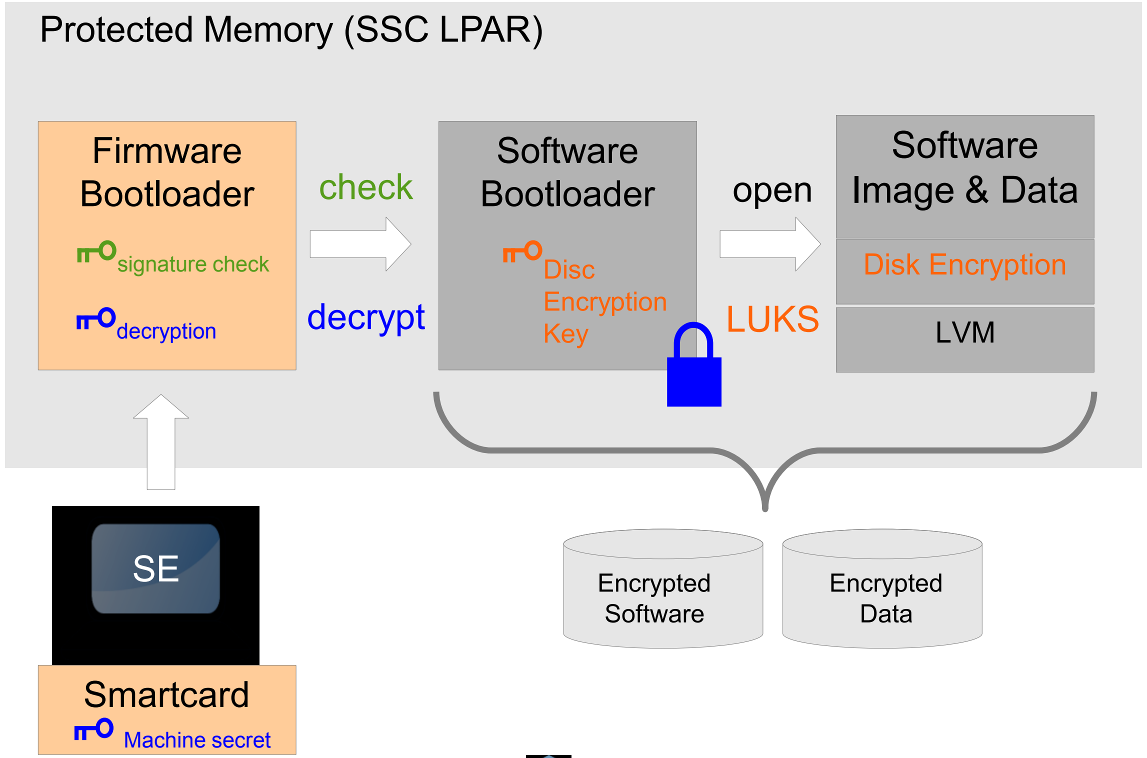
Security

- Provides tamper protection during appliance installation and runtime
- Ensure confidentiality of data and code running within the appliance – both at flight and at rest



SSC Security Concept

Encrypted, Signed, Tamper Resistant, Protected



Boot sequence:

1. Firmware bootloader is loaded in memory
2. Firmware loads the software bootloader from disk
   1. Check integrity of software bootloader
   2. Decrypt software bootloader
3. Software bootloader activate encrypted disks
   1. Key stored in software bootloader (encrypted)
   2. 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

THE CRYPTO EXPRESS6S ADAPTER

The Crypto Express6S Adapter

At z14 GA CEX6S adapters will be tolerated

- on all supported distribution releases
- supported release may need ”toleration” patch
- CCA and EP11 require a package update from
  - http://www.ibm.com/security/cryptocards/pciecc2/lonzsoftware.shtml

“Toleration” means

- CEX6S adapters will be accepted by Linux releases with CEX6S toleration support.
- CEX5S function will be available on CEX6S adapters.
- The kernel will “pretend” a CEX6S adapter is a CEX5S adapter.
- Performance improvements of CEX6S adapters will be available to Linux.

Pervasive Encryption for Linux on z Systems -- When?

- z14 CPACF HW performance improvements
  - dm-crypt, libica, openSSL, openCryptoki, Java, GSKit: all distributions at z14 GA
- z14 CPACF new CPACF instructions (GCM, TRNG)
  - Java, GSKit: future release update
  - kernel (TRNG): distribution releases to pick up the new code
  - openSSL
    - once patches accepted upstream distribution releases to pick up the new code
    - until then GCM via ibmca 1.4: distribution releases to pick up the new code
- Protected key support
  - basic support (pkey & paes modules, zkey tool): distribution releases to pick up the new code
  - dm-crypt with plain format: implicit with basic protected key support
  - dm-crypt with LUKS format: once cryptsetup patches accepted upstream distribution release to pick up the new code
- SSC
  - GA for some appliances (BC, VSENA)
  - more appliances to follow

Summary

Pervasive Encryption for Linux provides

- fast and consumable data protection for data in-flight and data at-rest
- transparent fast encryption for data at rest
- extended security for data at-rest with protected key dm-crypt
- a trusted computing environment for sensitive appliances through Secure Service Containers

IBM will work with its distribution partners to include the new function code being accepted by open source projects in future distribution releases.

Questions?