Pervasive Encryption for Linux on IBM Z and LinuxOne

Ingo Franzki
ifranzki@de.ibm.com

Reinhard Buendgen
buendgen@de.ibm.com

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

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.
IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.
ITIL is a Registered Trade Mark of AXELOS Limited.
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 other countries.
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.
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.
Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
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.
UNIX is a registered trademark of The Open Group in the United States and other countries.
VMware, the VMware logo, VMware Cloud Foundation, VMware Cloud Foundation Service, VMware vCenter Server, and VMware vSphere are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.
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.

PERVASIVE ENCRYPTION

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
- Without changing applications
- Without 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 for the Linux on Z and LinuxOne Platform

Technical Foundation

IBM z14 or Emporer II - Designed for Pervasive Encryption

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

Linux on Z and LinuxOne - Full Power of Linux Ecosystem combined with IBM z14 or Emporer II Capabilities

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

z/VM – New: encrypted paging

- z/VM 6.4 APAR VM65993

PERVASIVE ENCRYPTION FOR LINUX ON Z AND LINUXONE

USE CASES

Use Case 1: Mongo DB Server

As a 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 at rest

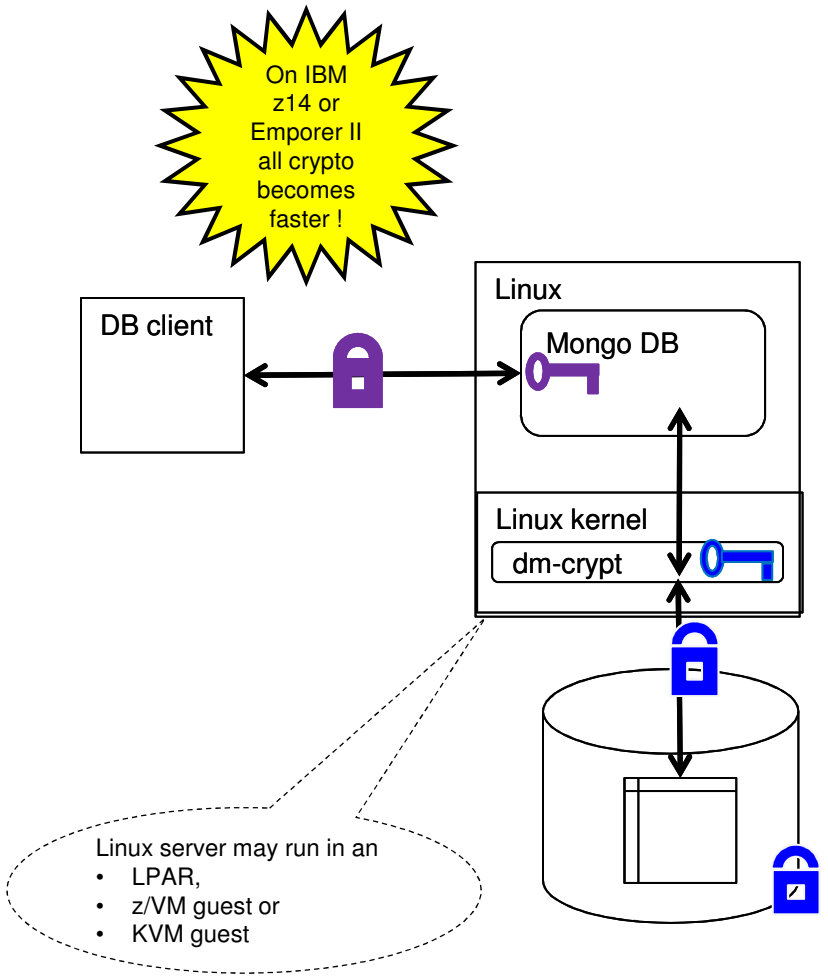
- End-to-end volume level encryption by Linux kernel (dm-crypt)
- *Transparent* usage of CPACF by Linux kernel
- Protected key option possible

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)

Secure manner of key generation

- CPACF true random numbers are fed in kernel entropy pool



Use Case 2: Mobile Server Farm in a Trusted Hypervisor

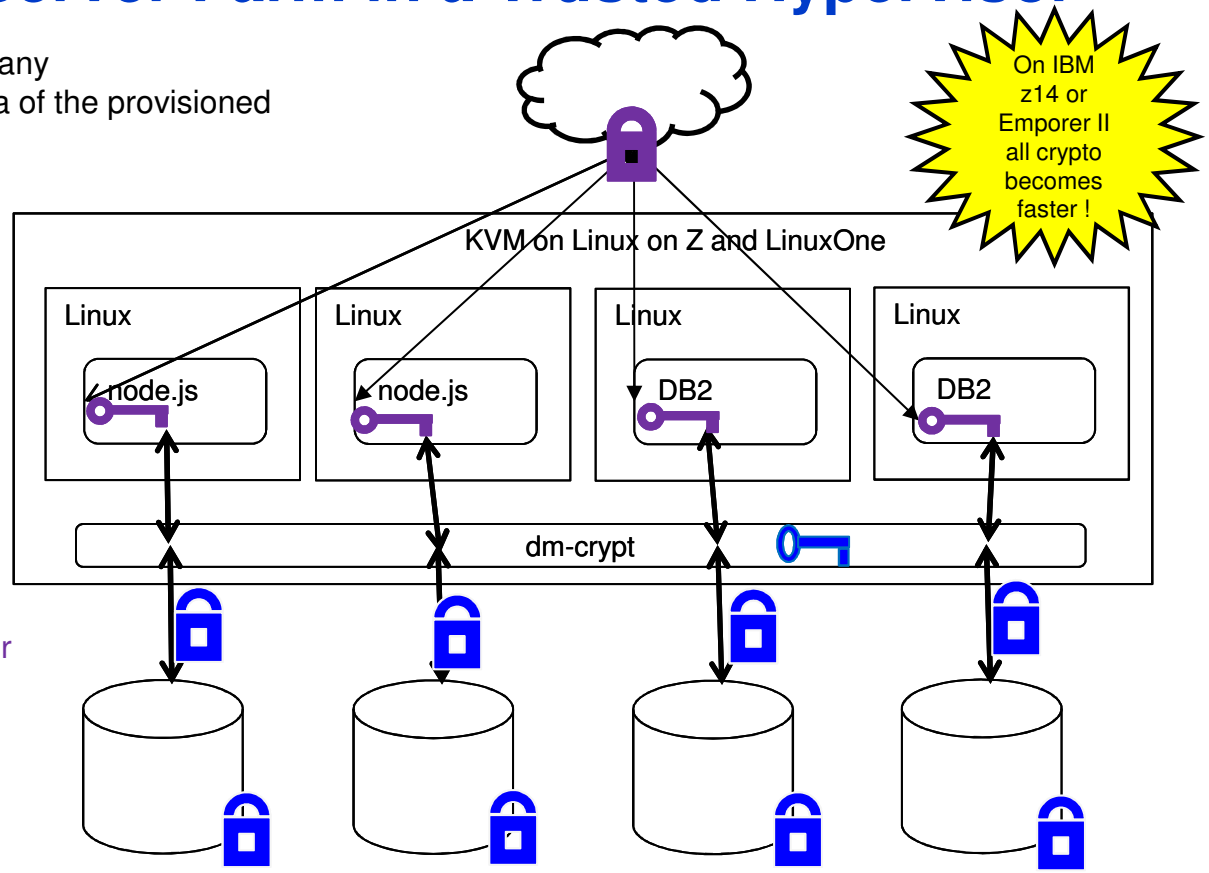
As an operator of a data center for my company
I want to host a server farm such that all data of the provisioned servers shall be transparently encrypted

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 Network encryption in node.js or apache or DB2
- Transparent usage of CPACF and SIMD via openSSL or GSKit

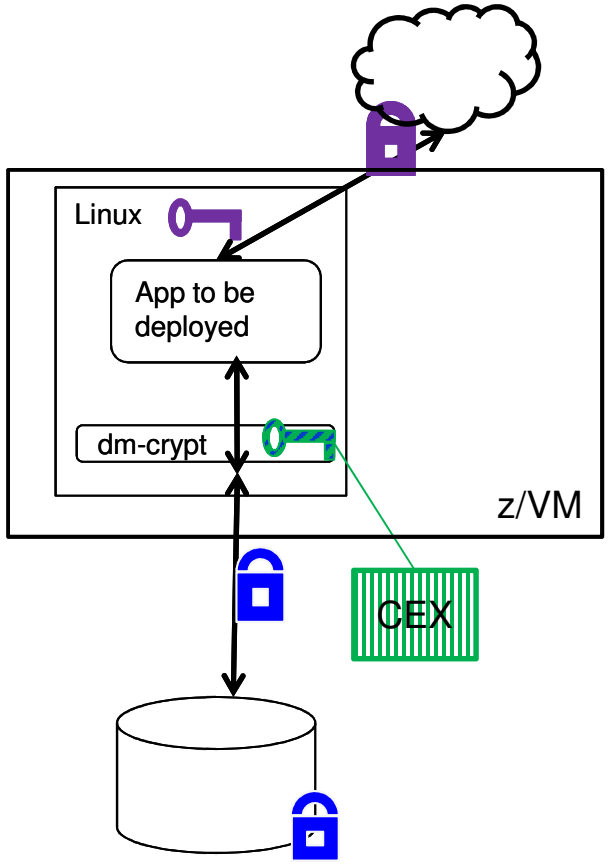


Use Case 3: PaaS for Sensitive Data

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

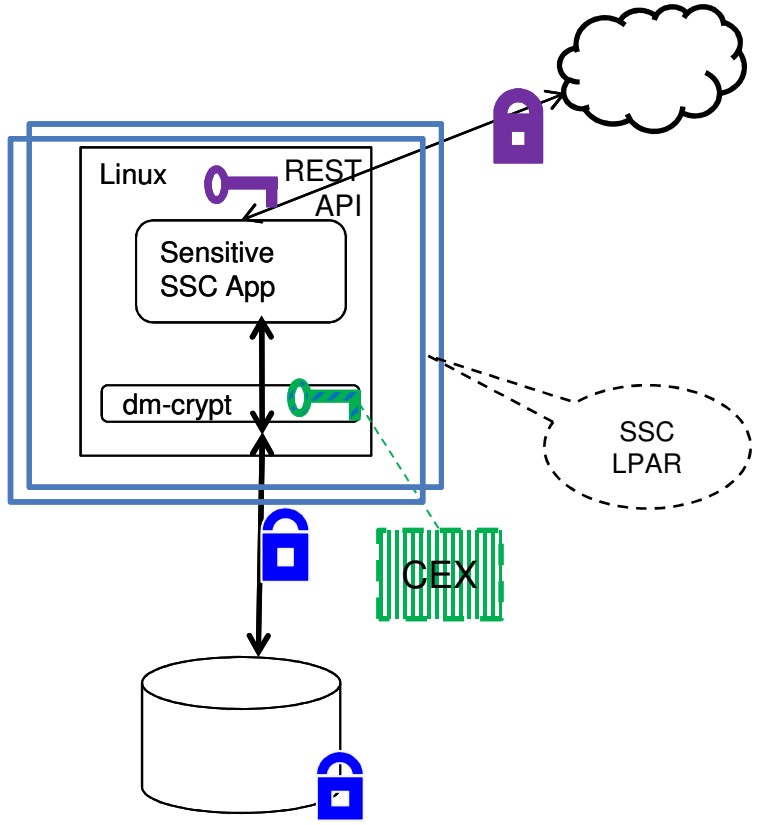
- End-to-end volume level encryption by Linux kernel (dm-crypt)
- *Transparent* usage of protected key CPACF by Linux kernel
- *Unique security and usability enhancement*
  - No clear key in memory
    - Use protected keys
    - Requires Crypto Express adapter
  - Automatic boot
- Plain text data in 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 AND LINUXONE

TECHNICAL CONTENTS

Technical Aspects of Pervasive Encryption for Linux on Z

Improved crypto performance
- Benefit from accelerated CPACF functions
- Exploit improved & new IBM z14 or Emporer II CPACF functions
- Exploit Linux on Z and LinuxOne SIMD support

Easy crypto consumability
- Linux is Linux, but exploiting Linux on Z and LinuxOne 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 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 Linux on Z and LinuxOne CPACF and SIMD code by libcrypto (w/o ibmca engine)
  - Focus on TLS 1.2 and 1.3 ciphers
  - No Linux on Z and LinuxOne specific configuration required
  - First patches (including AES-GCM support) accepted for openSSL version 1.1.1-beta
- **IPsec**
  - Bulk encryption and authentication implemented by kernel crypto
  - Transparently uses CPACF
  - Kernel 4.15 and later use new CPACF instruction for AES-GCM
- **GSKit**
  - IBM C library for TLS and crypto
  - E.g. used by IBM HTTP Server (IHS)
  - Uses Linux on Z and LinuxOne CPACF
  - Release 8.0.50.86 will use new Emperor II CPACF instructions
- **Java 8 / JCE**
  - Exploitation of Linux on Z and LinuxOne CPACF and SIMD code
  - Java 8 service refresh 5 will use z14 or Emporer II CPACF instructions

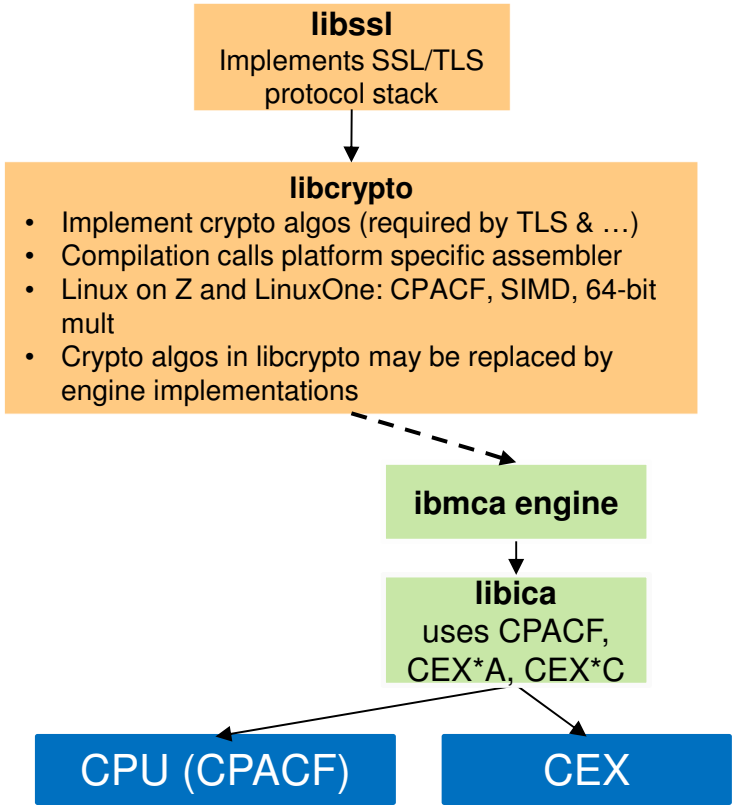
The new Linux on Z openSSL strategy

Original Linux on Z and LinuxOne strategy

- Put Linux on Z and LinuxOne specific code in the ibmca engine (only)
- Pro: all Linux on Z and LinuxOne specific user space cyrpto in libica
- 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
  - Outlook: RSA, & ECC acceleration via SIMD arithmetic
- ibmca engine config needed for
  - Offload/acceleration of RSA, DH, DSA, ECC, via Crypto Express adapters
  - Legacy crypto: 3DES
  - Configure engine to not support AES or hashes



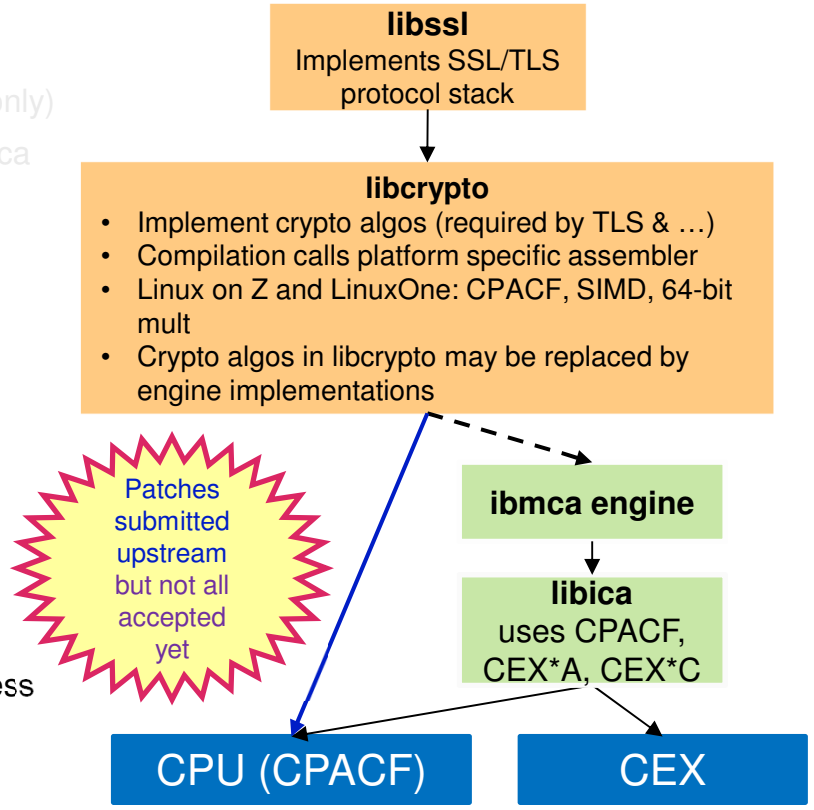
The new Linux on Z openSSL strategy

Original Linux on Z and LinuxOne strategy

- Put Linux on Z and LinuxOne specific code in the ibmca engine (only)
- Pro: all Linux on Z and LinuxOne specific user space cyrpto in libica
- Cons: engines must be configured

New Strategy

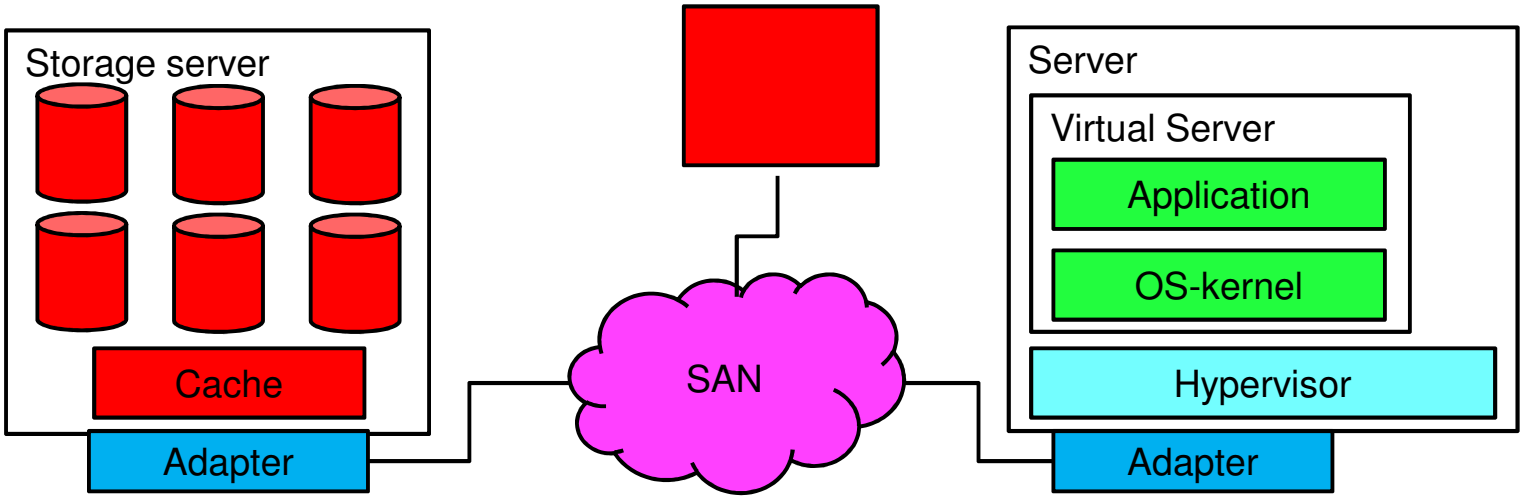
- All CPU dependent code (SIMD, CPACF) in libcrypto
- Crypto Express dependent code in ibmca
- No config needed for
  - Hashes (SHA1, SHA2,)
  - AES (ECB, CBC, OFB, CFB, XTS, CTR, GCM, CCM)
  - chacha20, poly1305
  - Outlook: RSA, & ECC acceleration via SIMD arithmetic
- ibmca engine config needed for
  - Offload/acceleration of RSA, DH, DSA, ECC, via Crypto Express adapters
  - Legacy crypto: 3DES
  - Configure engine to not support AES or hashes



PERVASIVE ENCRYPTION FOR LINUX ON Z AND LINUXONE

TECHNICAL CONTENTS: DATA AT REST

Data at Rest Encryption Considerations



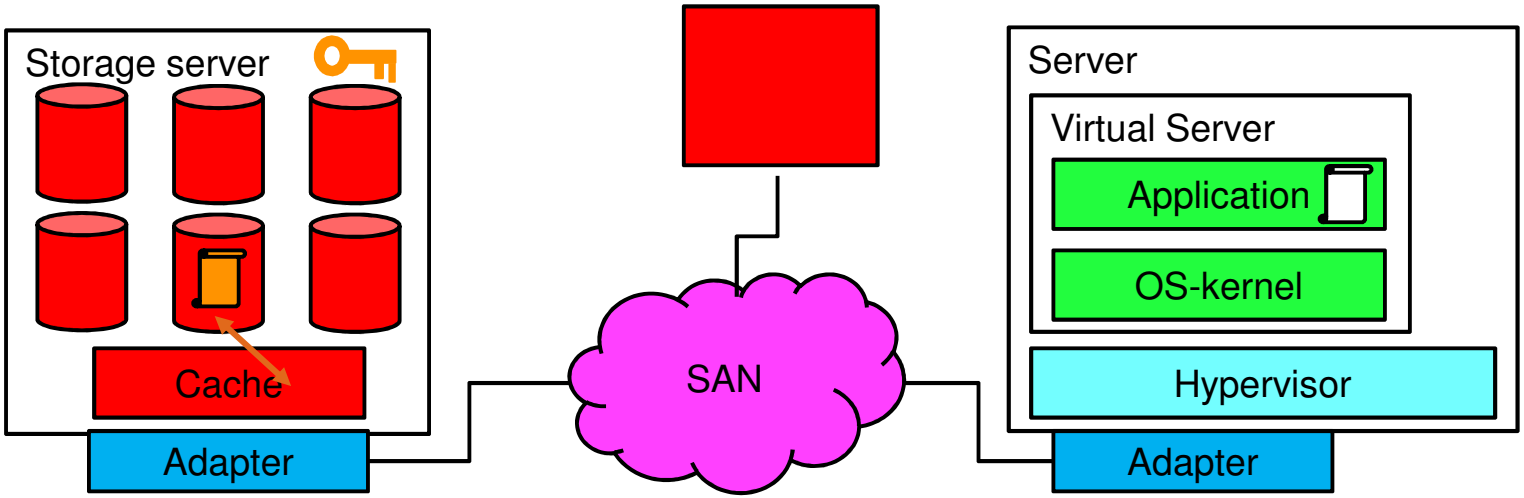
Attack points

- Storage server
- SAN
- Server / Hypervisor / Virtual Server
  - 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



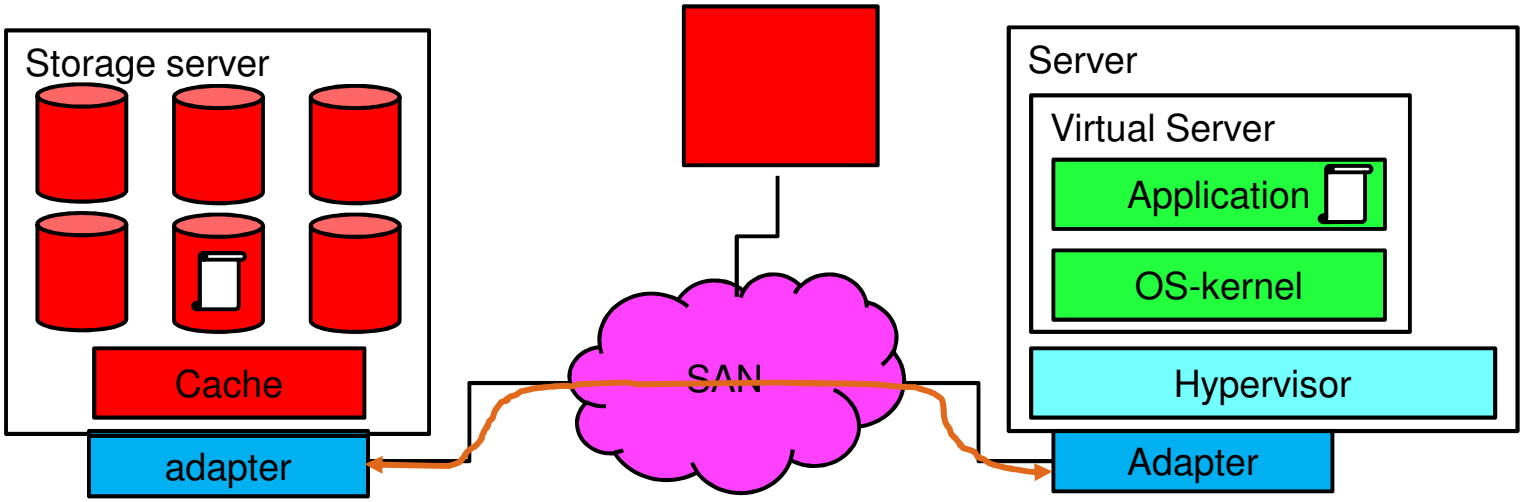
Attack points

- Storage server – (Secured)
- SAN – Not secure
- Server / Hypervisor – Not secure
- Virtual Server 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



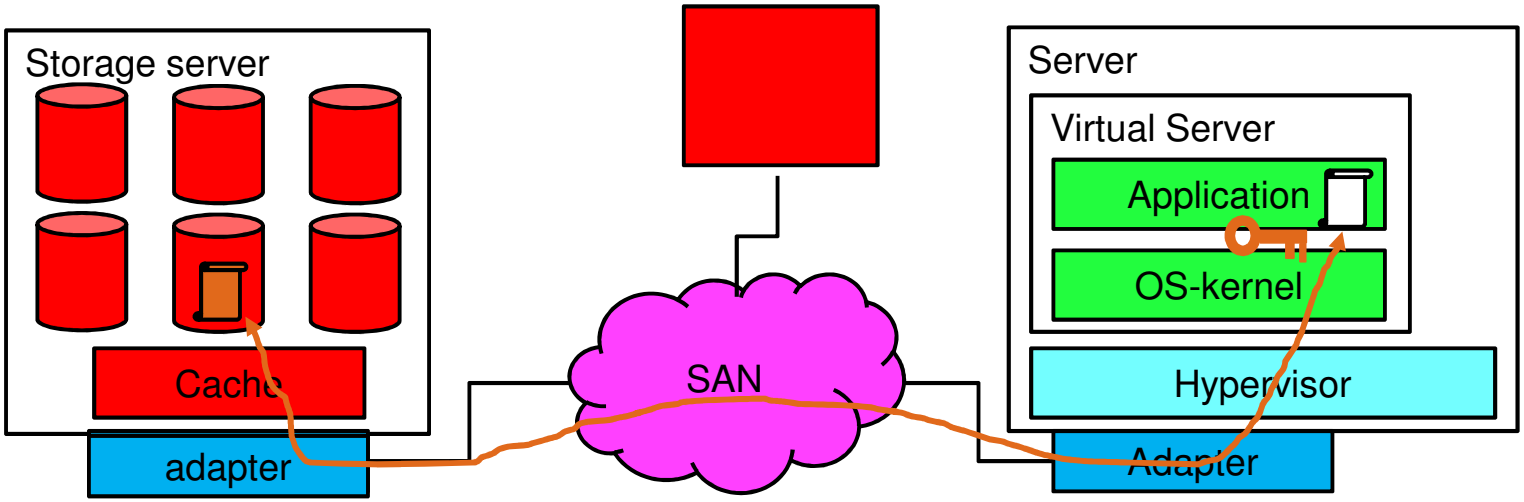
Attack points

- Storage server – Not secure
- SAN – Secured
- Server / Hypervisor – Not secure
- Virtual Server insider / outsider – Not secure

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



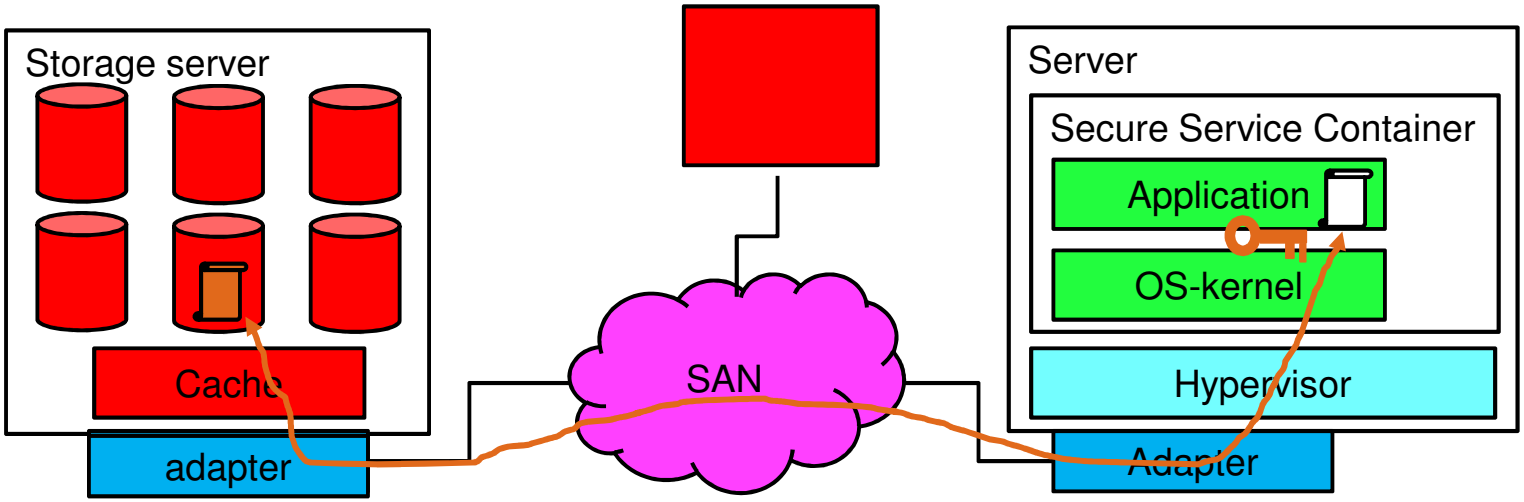
Attack points

- Storage server – Secured
- SAN – Secured
- Server / Hypervisor – Secured
- Virtual Server insider / outsider – Not secure

Questions

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

End-to-End Data Encryption from SSC



Attack points

- Storage server – Secured
- SAN – Secured
- Server / Hypervisor – Secured
- Virtual Server insider / outsider – Hardened

Questions

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

Linux on Z Encryption of Data at Rest

End-to-End Encryption

- **dm-crypt: block device / full volume encryption**
  - Uses kernel crypto
  - Granularity: disk partition / logical volume
- **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 crypto libraries
  - Granularity: file
- **DB2 native encryption: data base encryption**
  - Uses GSKit crypto library

Network Encryption

- **NFS v4 with encrytion option: encryption of file transport**
  - Uses kernel crypto
- **SMB v3.1: encryption of file transport**
  - Uses kernel crypto

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

GSKit and latest versions of Clic use CPACF for AES

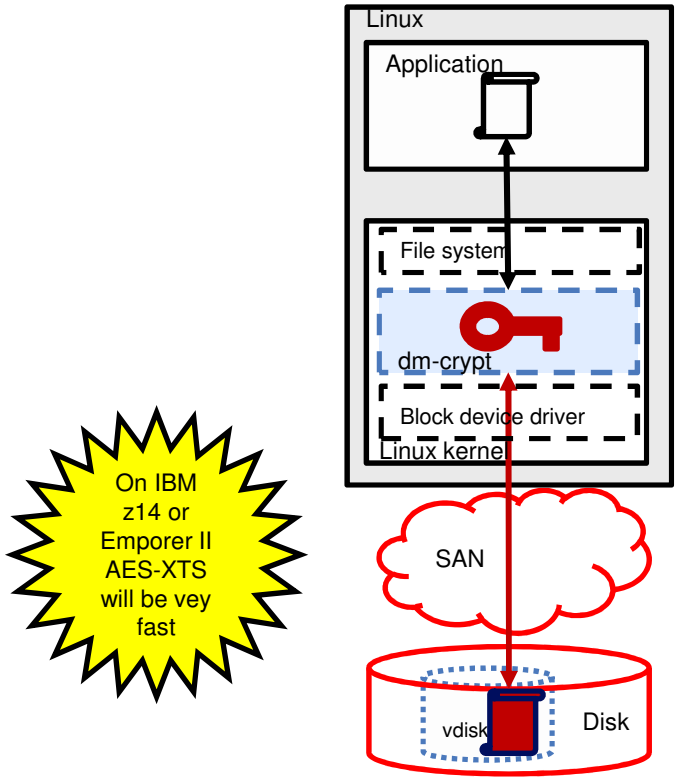
DM-CRYPT & LUKS

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

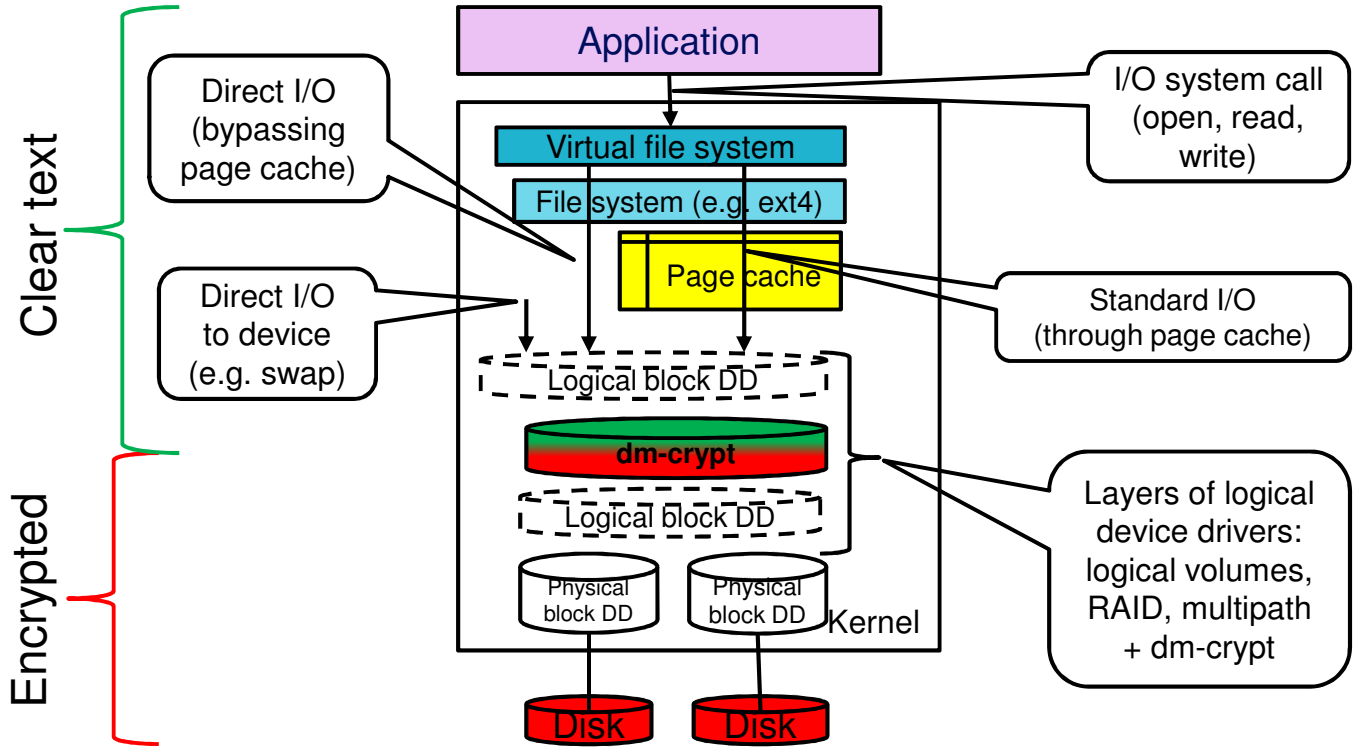
- **End-to-End data encryption**
  - The complete I/O path outside the kernel is encrypted:
    - Hypervisor, adapters, links, switches, disks

- **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 Linux on Z and LinuxOne CPACF Crypto:
      - AES-CBC
      - AES-XTS (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



dm-crypt: Linux Unified Key Setup (LUKS)

Plain

- **No header**
- **No formatting required**
- **Key & cipher name must be supplied with every open**
  - Key must be stored in a file in the file system

LUKS1

- **Header on disk**
  - Fixed binary header format
- **One time formatting required**
- **Key & cipher name is contained in the header**
  - Key is wrapped by a key derived from a passphrase
  - Up to 8 key-slots

LUKS2

- **Header on disk**
  - Flexible header format (JSON)
  - Redundancy of metadata
  - Detection of metadata corruption
  - Automatic repair from metadata copy
- **One time formatting required**
- **Key & cipher name is contained in the header**
  - Key is wrapped by a key derived from a passphrase
  - Up to 32 key-slots

How to Set-up a dm-crypt LUKS Volume

Once

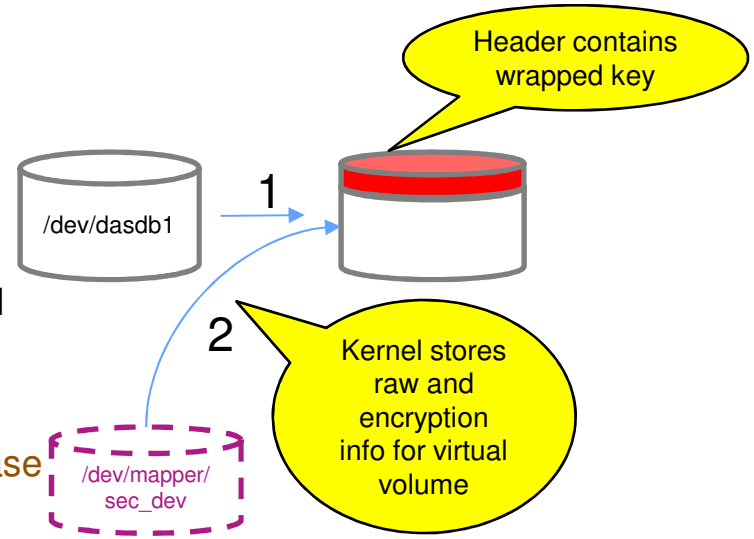
1. **Format "raw" volume as dm-crypt volume**
   - cryptsetup luksFormat …
     - cipher, key length, hash, passphrase
   - Writes LUKS header to disk

With every boot

2. **Open dm-crypt volume and assign it a virtual volume name**
   - cryptsetup luksOpen …
     - dm-crypt volume, virtual volume, passphrase
   - Creates virtual volume in **/dev/mapper**
   - Can be automated with **/etc/crypttab**

Business as usual

3. **Use virtual volume**
   - mkfs (or mkswap)
   - mount (or swapon)
   - Any kind of standard I/O
   - Do **not** use raw volume directly



On CPACF Performance

- CPACF performance depends on the size of buffers being en-/decrypted
- The larger the buffer the better the performance
  - Best performance with ≥ 4kB buffers
- dm-crypt always used 512 byte buffers
- Starting with kernel 4.12 and cryptsetup 2.0:
  - dm-crypt can use 4kB buffers
    - Plain mode
    - LUKS2
    - **NOT:** LUKS1 !

	512b	4kB
z13 or Emporer	1	1.5
z14 or Emporer II	4	13

Relative AES GCM/XTS in-memory performance measured with openSSL speed test

What Volumes can be Encrypted by dm-crypt?

- **YES: block devices**
  - Disks:
    - SCSI disks
  - Partitions of
    - ECKD DASDs
    - SCSI disks
  - Muti-path devices
  - (LVM2) Logical volumes
  - Loop back devices
  - Other device mapper devices

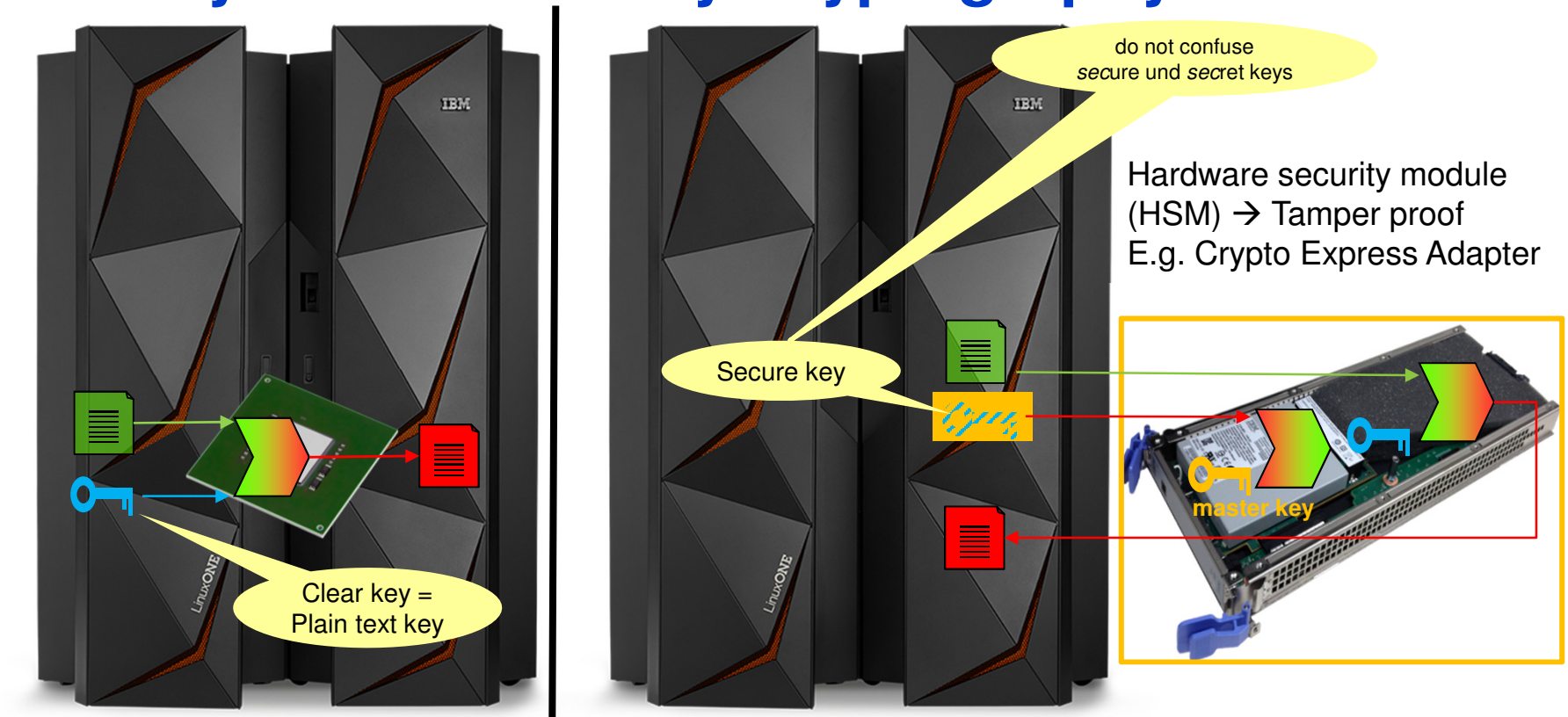
NO:

- Full ECKD DASDs
- Network file systems like NFS
  - However, you can create a loopback device based on a file in a network file system

PERVASIVE ENCRYPTION FOR LINUX ON Z AND LINUXONE

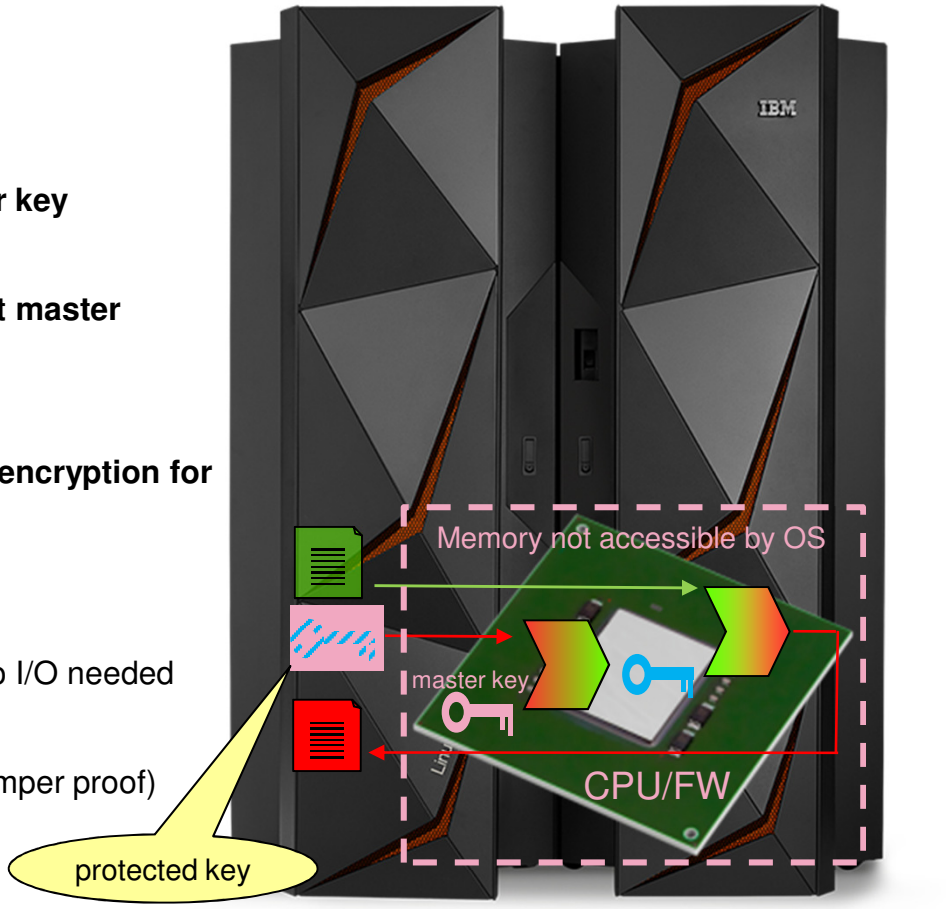
SUPPORTING CPACF PROTECTED KEY CRYPTO

Clear Key vs. Secure Key Cryptography



CPACF Protected Keys

- **Linux on Z and LinuxOne function of the CPU (CPACF)**
- **Each virtual server (LPAR or guest) has a *hidden* master key**
  - Not accessible from operating system in LPAR or guest
- ***Protected key*: a key wrapped by the hidden LPAR/guest master**
- **Protected key tokens can be generated**
  - From secure keys using CEX Adapter (secure)
- **Linux on Z and LinuxOne 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 as in HSM (not tamper proof)
    - Not certified as HSM
    - The LPAR/guest master key is not persistent
      - need to store (secure) key to derive protected key from

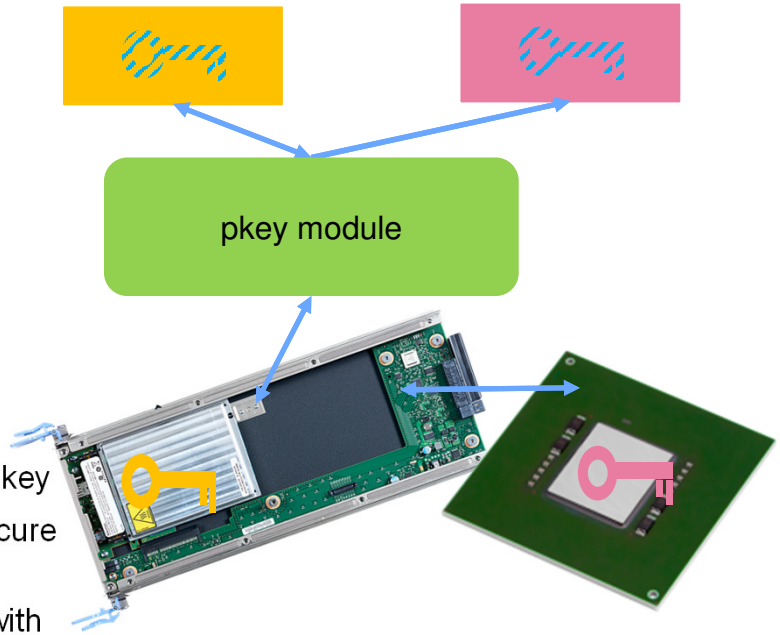


Managing Protected Keys: the pkey Kernel Module

- **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

Key repository support for the zkey tool

Upstream since s390tools 2.4.0

Enhancements to the zkey tool to store secure AES keys in a key repository

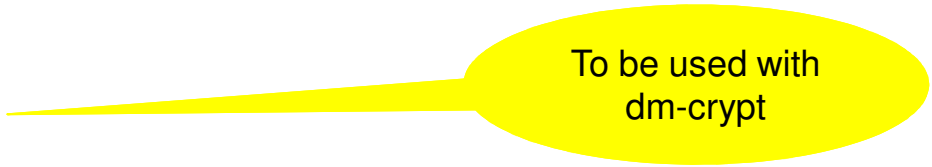
- E.g. located in /etc/zkey/repository
- Allows to associate volumes and crypto adapters (APQNs) to secure keys

Subcommands:

- **Generate** Generates an AES secure key in a file or in the key repository
- **Re-encipher** Support master key change on Crypto Express adapter for secure AES key in a file or in the key repository
- **Validate** Validate an existing secure AES key in a file or in the key repository
- **Import** Import a secure AES key from a file into the key repository
- **Export** Export a secure AES key from the key repository into a file
- **List** List keys in the key repository
- **Remove** Remove a secure AES key from the key repository
- **Change** Change the associations of a secure AES key in the key repository
- **Rename** Rename a secure AES key in the key repository
- **Copy** Copy/Duplicate a secure AES key in the key repository
- **Crypttab** Generate crypttab entries for volumes associated with secure AES keys in the key repository
- **Cryptsetup** Generate or run cryptsetup commands for volumes associated with secure AES keys in the key repository

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**
  - Transform secure key into protected key
  - Cache protected key (into encryption context aka transform)
  - Use protected key for cryptographic operations




```
# cat /proc/crypto
...
name         : xts(paes)
driver       : xts-paes-s390
module       : paes_s390
priority     : 400
refcnt       : 1
selftest     : passed
internal     : no
type         : blkcipher
blocksize    : 16
min keysize  : 128
max keysize  : 128
ivsize       : 16
geniv        : <default>

name         : cbc(paes)
driver       : cbc-paes-s390
module       : paes_s390
priority     : 400
refcnt       : 1
selftest     : passed
internal     : no
type         : blkcipher
blocksize    : 16
min keysize  : 64
max keysize  : 64
ivsize       : 16
geniv        : <default>
...
```

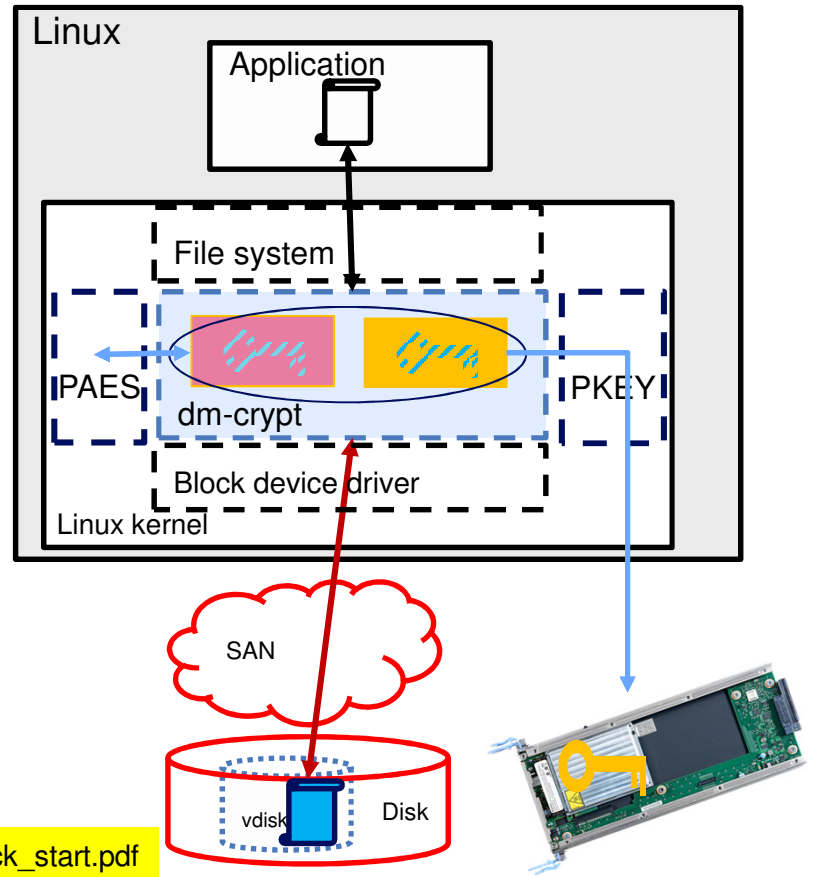
PERVASIVE ENCRYPTION FOR LINUX ON Z AND LINUXONE

SUPPORTING CPACF PROTECTED KEY CRYPTO FOR DM-CRYPT

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

- **End-to-End data encryption**
  - The complete I/O path outside the kernel is encrypted:
    - Hypervisor, 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 Linux on Z and LinuxOne CPACF protected key crypto:
      - PAES-CBC
      - PAES-XTS (recommended)
  - Encrypted volumes must be opened before usage
    - Opening provides encryption key to kernel
    - Establishes virtual volume in /dev/mapper



http://public.dhe.ibm.com/software/dw/linux390/docu/l5n1dc00_a_quick_start.pdf

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 seckey.bin -xts
```

  - Requires access to CEX[5|6]C adapter

With every boot

- **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

Business as usual

- **Use new device mapper volume**
  - (only once) Create file system:

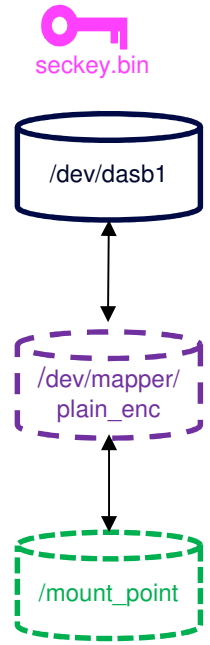
```
# mkfs.ext4 /dev/mapper/plain_enc
```

  - Mount:

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

  - Access:

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



Using the PAES with dm-crypt – LUKS2 Format

Once

0: Insert new kernel modules during boot & generate secure keys

- requires access to CEX5C or CEX6C adapter
- `zkey generate seckey.bin -xts`

Once

1: format "raw" volume as LUKS2 volume

- `# cryptsetup luksFormat --type luks2 --cipher paes-xts-plain64 --master-key-file seckey.bin --key-size 1024 /dev/dasdb1`
  - Asks for a passphrase
  - Writes LUKS2 header to disk

Requires cryptsetup 2.0.3

With every boot

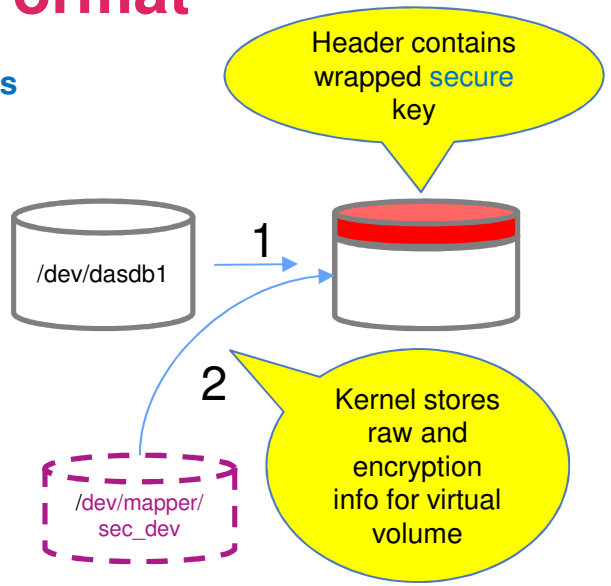
2: Open dm-crypt volume and assign it a virtual volume name

- `# cryptsetup luksOpen /dev/dasdb1 sec_dev`
  - Asks for passphrase (needs not be protected)
  - Creates virtual volume in /dev/mapper

Business as usual

3: Use virtual volume

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



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

Attention: Additional tooling to support master key change on Crypto Express adapter is being worked on

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 format)
    - Backup secure key
  - To deal with LUKS header corruption
    - Back up dm-crypt LUKS 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

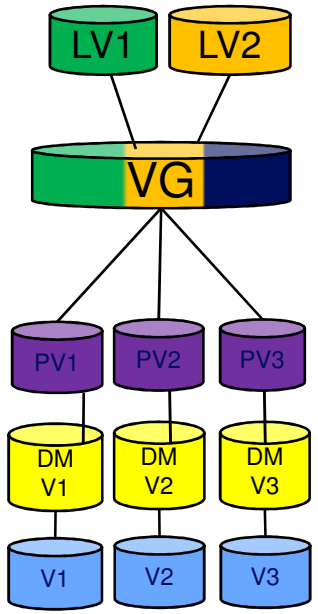
- **Configuration file to describe how dm-crypt volumes are to be opened**
- **Will be read and interpreted before /etc/fstab**
- **Format**
  - Each line describes a dm-crypt volume:
    - <dm-crypt volume> <path of block-device> <password file>|`none` [options]
      - for the plain format, the password 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
```

ATTENTION:
Currently crypttab (systemd) does not yet support to use a sector size different to 512 bytes with plain mode volumes

Using dm-crypt Volumes as LVM Physical Volumes

Use virtual dm-crypt device as input to vgcreate



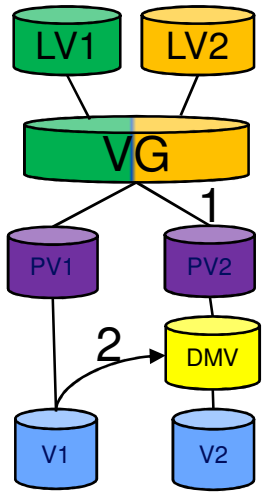
Migrate data from unencrypted volume to dm-crypt volume

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

```
# vgextend VG PV2
```

- 2: migrate data from V1 to DMV:

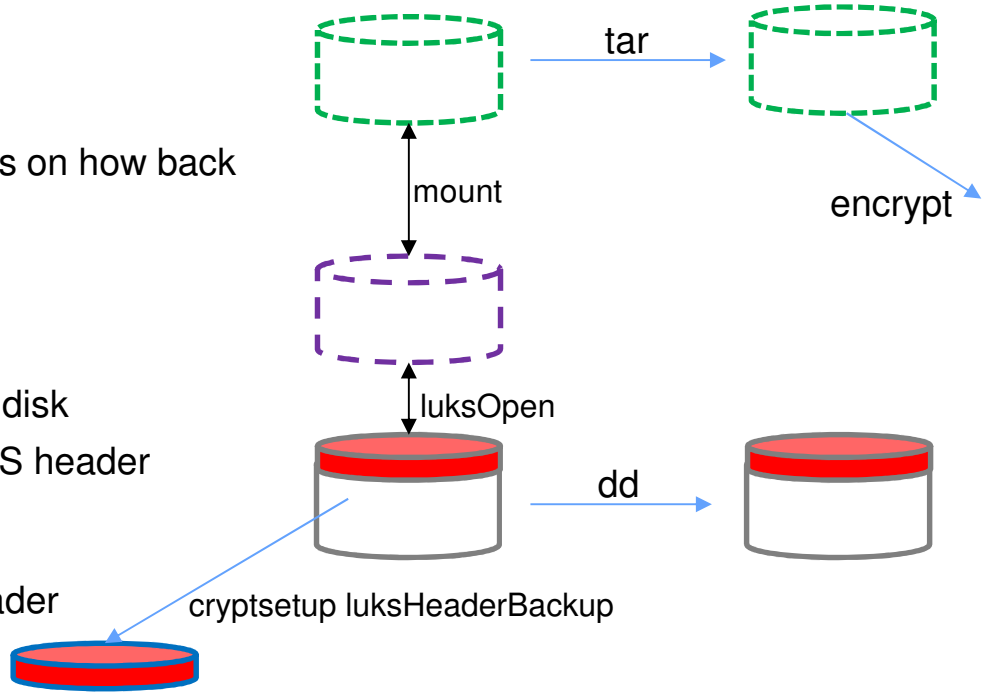
```
# pvmove V1 DMV
```



Works transparently while applications access LVs

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



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

Questions?