WARP MECHANICS®

HGST
a Western Digital brand

ORNL / DoD Lustre Ecosystem 2017

Managing self-encrypting HDDs with Lustre/ZFS

Q2-2017

Josh Judd, CTO

OAK RIDGE National Laboratory

Agenda

- Self-Encrypting Drive concepts
- Required hardware
- Open software design theory
- WARP's implementation as example
- Underlying code which makes WARP's implementation work
  - Pointers on where to start
  - Actually *implementing* it would be *way* longer than this session timeslot :)

WARP MECHANICS®

Overview

- Data at rest encryption →
  - Good concept; has tradeoffs
- Yesterday →
  - Software encryption = performance penalties
  - Hardware encryption = vendor-locked and expensive
- Today →
  - COTS hardware self-encrypting HDDs and SSDs
  - Balance high performance hardware encryption with low cost open software
  - But… er… *what* software, exactly…?
  - "Off the shelf" open software is lacking
  - Options include: buy it, build it, or (balance) get somebody friendly to help

WARP MECHANICS®

Data at Rest Encryption

- Several options for encrypting disks
- An open <u>software-only</u> approach could be something like WARP's Ecosystem presentation from last year:

```
cd /dev/disk/by-vdev
cryptsetup create eXXpAdYY eXXpAdYY
cryptsetup luksFormat /dev/mapper/eXXpAdYY
cryptsetup luksOpen eXXpAdYY eXXpAdYY
mkfs [ ... ] /dev/mapper/eXXpAdYY
```

PROBLEM: "Substantial" performance impact for SSDs (e.g. 50%) *and* any other CPU-, IOPS-, or latency-intensive workloads

*(** Note: e_p_d_ is WARP's meaningful UDEV scheme for disk names)*

WARP MECHANICS®

Hardware Data at Rest Encryption

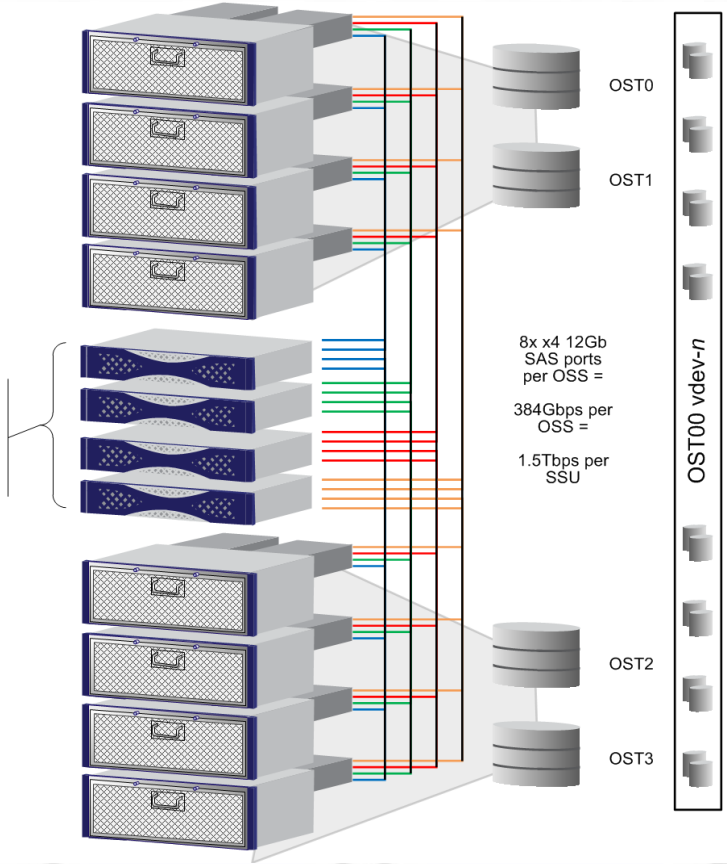
- **Solves** performance problems with software approach
- Historically, required expensive proprietary systems
- Now, can be done with standard hardware at low incremental cost
  - E.g., +1% or so system-level cost vs. equivalent non-encrypted drives

- NEW PROBLEM: Open software lags far behind for managing keys, lock states, and other encryption-specific features

- **New Solution**: DIY tools are not all that difficult to write... Or just send us a note and we'll help :)

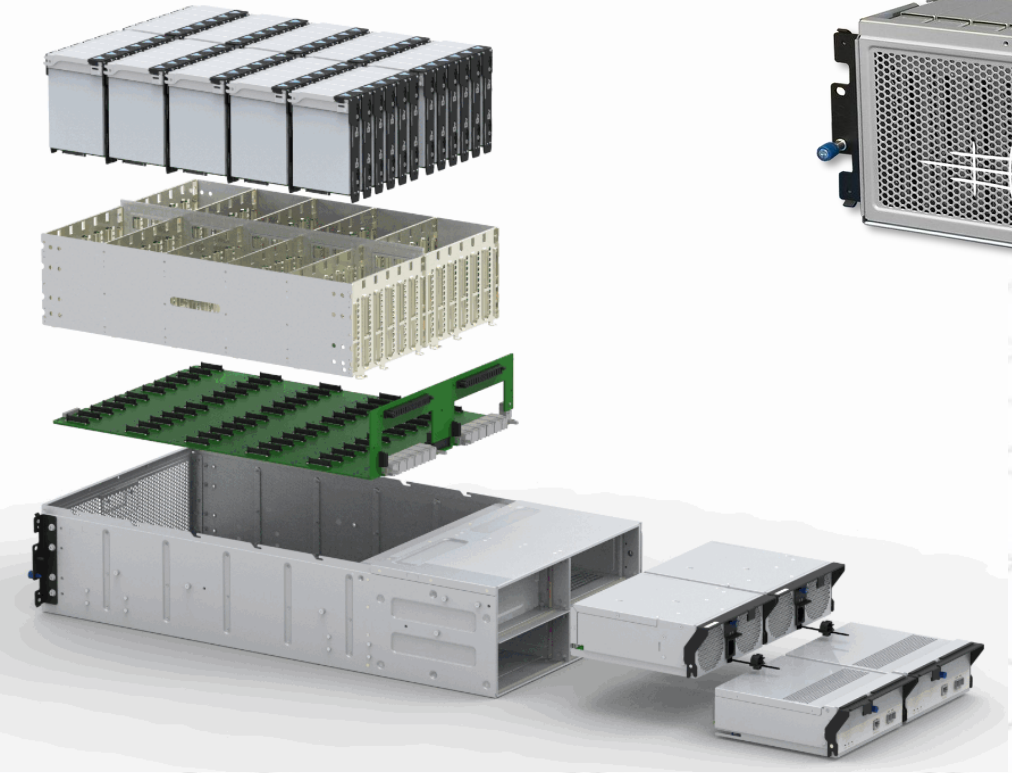
WARP MECHANICS®

Reference Design for Discussion: WARP*hpc* 41000



WARP MECHANICS®

Reference Design for Discussion (cont.): WARP WDS-9460 (HGST)



- **720TB per 4u**
- ~38 GBytes/s raw per shelf
- Enterprise "helium" SAS PMR
- WARP feature enhanced
- Zero performance penalty TCG HDD encryption

WARP MECHANICS®

Nomenclature

- BDE: Bulk Data Encryption – standard for SED encryption
- HGST: Premium division of WD = Gold standard HDDs and SSDs
- Password: Opens the encrypted/portable secure_keys container, which contains PINs
- PIN: A key used to authenticate to a TCG drive, which allows you to send it commands such as “lock” or “unlock”
- SED: Self Encrypting Drive
- TCG: Trusted Computing Group – standard for SED encryption
- WARP: “Wicked-fast Application Resource Platform” which runs all this nice encryption management software :)

HGST TCG SAS Helium HDDs and SSDs

- Underlying hardware in reference solution: HGST TCG drives
- Helium = highest density, best reliability, lightest weight, PMR
- PMR = "Normal" HDDs, as opposed to "SMR"
- TCG = "Trusted Computing Group" standard for "Self Encrypting Drives", which provides multiple benefits:
  - Transparency: No OS or app modifications required
  - Re-encryption: With SED, there is no need to ever re-encrypt data
  - Performance: No degradation in SED performance; hardware-based
  - Standardization: Whole industry is building to the TCG/SED specifications
  - Safety: Drives can be unlocked with multiple keys – can cancel keys known by one specific admin without effecting organization's ability to access data
- BDE = "Bulk Data Encryption" is a similar standard, but strictly for lower end SATA drives, with fewer features and lower security

WARP MECHANICS®

“Open SED Management” Functional Requirements

- Drives encrypt themselves, all the time, no matter what
- However, to be *useful*, you need to *manage* the encryption
- At a high level, tools must handle cases such as:
  - Detect if a drive supports encryption, and if so, whether it is TCG vs. BDE
  - Manage PINs for all drives collectively and securely
    - Admins don’t have to manually unlock 1000s of drives in a single rack
    - PINs can be easily replicated and backed up
  - Turn drive locking on and off for individual drives or full systems
  - Allow all running directly-connected servers to “see” all drives, for HA
  - Allow drives to remain unlocked when OSS/MDS/MGS reboot or switchover
  - Manage PINs when replacing a failed drive
  - Handle lock status changes for re-seating drives
  - Display status of locking

Open SED Management Software Design

- The WARP Way:
  - CLI utilities to manage
    - Encryption is changed *very rarely,* and should not be changed by Jr Admins
    - OK... GUI *can* manage it if desired... But not going to use that for examples
  - Store the (large number of) drive PINs in a encrypted container
  - Utilities accept a single password to unlock that file, then it can *stay* unlocked, and utilities manage PINs on the drives for you
  - If you copy the container file (backup or replicate) you'll get <u>all</u> the PINs copied securely
  - If an admin quits, you can change just one password, and not have to re-key drives or re-encrypt any data

WARP MECHANICS®

Open SED "WARP Implementation" Walk Through

- Initial power on: All TCG drives are encrypting, but unlocked
  - They look just like any other drive, and are accessible to all attached servers
  - However, *internally*, they are already using 256-bit encryption
- Initialize drive locking with WARP's "`wmsedisk`" tool
  - Creates encrypted "`secure_keys_container`" file, which contains all drive PINs, and can be backed up and/or copied to other WARP servers
  - Drives are now encrypted *and* protected: unreadable if powered off
  - But they are currently unlocked and thus visible to all SAS-attached WARPs
- Create pools and filesystems, <u>if</u> they didn't already exist
  - Unlike software encryption method, this step actually *can* be performed *first*
  - You can create filesystems and load data, then enable encryption, or vice versa

WARP MECHANICS®

Open SED Walk Through *(continued)*

- Test to ensure that locking is working as expected
  - Completely power down all servers and JBODs, or cycle just the JBODs
  - Power on servers then JBODs
  - All drives should be locked and *not usable* by any of the servers
- Log into any attached server which has a "`secure_keys_container`"
  - If you rebooted the server or manually locked the container, "`secvol`" unlocks
- Send command to unlock drives with "`wmsedisk`"
  - Prompts for your PIN container password, and makes drive PINs available
  - The server you're on will now see all drives as mountable
  - Run "partprobe" on all other directly-attached servers to get them to notice
- Import all zpools to their associated servers, and start Lustre
- Until the next cold boot of the JBODs, or ejection of HDDs/SSDs, it should work like any other Lustre system

Specific Command Examples

Whether you use WARP or “roll your own”, you need commands such as the following to manage SED. Note that this example shows one drive. The tools also accept drive ranges, so you can toggle everything with one command.

```
# wmsedisk init e00pAd57

No user key file found.
Would you like to create your key file now [Y/n]?: y

Enter password: *********
Re-enter password: ********
Done: 100%

The VeraCrypt volume has been successfully created.

Enter password for /opt/warpmech/tcg/secure_key_container: ********
Please enter new user PIN: ********
Please confirm password: ********

Set authority PINs (SID PIN, Band0 PIN, Erase PIN) and enabled read/write lock on reset for e00pAd57.
```

Specific Command Examples *(cont.)*

```
# wmsedisk getlock e00pAd57

Enter password for /opt/warpmech/tcg/secure_key_container: ********

Lock state of 'e00pAd57':
            band 0
            lock_on_reset 00
            range_uid 0000000000000000
            read_lock 0
            read_lock_enabled 1
            write_lock 0
            write_lock_enabled 1
```

** NOTE: At this point, the key container file is unlocked, and will stay that way until you say otherwise

Specific Command Examples *(cont.)*

```
# wmsedisk lock e00pAd57

Locked band0 for e00pAd57.

# lsvdev -v | grep -E ”VDEV|e00pAd57"
 VDEV            SERIAL                 SIZE      [H:C:T:L]    TYPE    VENDOR    MODEL            REV   DEV        GEN
 e00pAd57        2EG1DP6G           _unset_sdk_   [13:0:2:0]   disk    HGST      HUH728080AL4201  B907  /dev/sdk   /dev/sg10

# wmsedisk unlock e00pAd57

Unlocked band0 for e00pAd57.

# lsvdev -v | grep -E "VDEV|e00pAd57"
 VDEV            SERIAL                 SIZE      [H:C:T:L]    TYPE    VENDOR    MODEL            REV   DEV        GEN
 e00pAd57        2EG1DP6G           8001.6 GB     [13:0:2:0]   disk    HGST      HUH728080AL4201  B907  /dev/sdk   /dev/sg10
```

Specific Command Examples *(cont.)*

With drives unlocked, you can treat them just like any other drives:

```
# zpool create volume-00 e00pAd57

# zpool status volume-00
  pool: volume-00
 state: ONLINE
  scan: none requested
config:

            NAME        STATE     READ WRITE CKSUM
            volume-00   ONLINE       0     0     0
              e00pAd57  ONLINE       0     0     0

errors: No known data errors
```

Specific Command Examples *(cont.)*

Test: Export zpool, power cycle JBOD, then check to see if drives come back locked or unlocked

```
# zpool export volume-00
```

(power cycle)

```
# wmsedisk getlock e00pAd57
Lock state of 'e00pAd57':
            band 0
            lock_on_reset 00
            range_uid 0000000000000000
            read_lock 1
            read_lock_enabled 1
            write_lock 1
            write_lock_enabled 1
```

```
# lsvdev -v | grep -E ”VDEV|e00pAd57"
 VDEV          SERIAL              SIZE      [H:C:T:L]    TYPE    VENDOR    MODEL            REV   DEV        GEN
 e00pAd57      2EG1DP6G        _unset_sdk_   [13:0:2:0]   disk    HGST      HUH728080AL4201  B907  /dev/sdk   /dev/sg10
```

Specific Command Examples *(cont.)*

```
# wmsedisk unlock e00pAd57

Unlocked band0 for e00pAd57.

# zpool import volume-00 -d /dev/disk/by-vdev

# zpool status volume-00
  pool: volume-00
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        volume-00   ONLINE       0     0     0
          e00pAd57  ONLINE       0     0     0
        cache
          e00pAd59  ONLINE       0     0     0

errors: No known data errors
```

Specific Command Examples *(cont.)*

You also have to manage the “secure volume” file that contains the drive PINs.

E.g., unlock it if you reboot the servers, or re-lock it if you want to ensure root-level admins cannot see the PINs.

```
# secvol open $warpdir/tcg/secure_key_container $warpdir/tcg/secure_keys

Mounting /opt/warpmech/tcg/secure_key_container to /opt/warpmech/tcg/secure_keys...
Are you sure [y/N]?: y
Enter password for /opt/warpmech/tcg/secure_key_container: ********

# secvol list

1: /opt/warpmech/tcg/secure_key_container /dev/mapper/veracrypt1 /opt/warpmech/tcg/secure_keys
```

Specific Command Examples *(cont.)*

```
# secvol info $warpdir/tcg/secure_keys –v

Slot: 1
Volume: /opt/warpmech/tcg/secure_key_container
Virtual Device: /dev/mapper/veracrypt1
Mount Directory: /opt/warpmech/tcg/secure_keys
Size: 9.8 MB
Type: Normal
Read-Only: No
Hidden Volume Protected: No
Encryption Algorithm: AES
Primary Key Size: 256 bits
Secondary Key Size (XTS Mode): 256 bits
Block Size: 128 bits
Mode of Operation: XTS
PKCS-5 PRF: HMAC-SHA-512
Volume Format Version: 2
Embedded Backup Header: Yes
```

Specific Command Examples *(cont.)*

```
# secvol info $warpdir/tcg/secure_keys

1: /opt/warpmech/tcg/secure_key_container /dev/mapper/veracrypt1 /opt/warpmech/tcg/secure_keys

# ll $warpdir/tcg/secure_keys/
total 15K
-rw------- 1 root root  33 Jul 17 09:08 band0_pin
-rw------- 1 root root  33 Jul 17 09:08 erase_pin
drwx------ 2 root root 12K Jul 17 09:08 lost+found
-rw------- 1 root root  33 Jul 17 09:08 sid_pin

# secvol close $warpdir/tcg/secure_keys

Dismounting /opt/warpmech/tcg/secure_keys...
Are you sure [y/N]?: y

# secvol list

Error: No volumes mounted.
```

Under the covers coding

- Say that you don't want to buy a WARP. (Big sad face.)
- No problem! You can roll your own, or ask us for (limited=free) help.
- You just have to implement the same kinds of *functions* we built – fields etc are all in the TCG standards docs
- That said, it does get a *liiiiii-ttle* complicated if you want to totally do it on your own...

```
1. Open a session to the Admin SP
    a. In start_admin_sess.bin
        i. Set byte #83 to 0x00 in start session file to set it as Admin SP
    b. StartSession
            # sg_raw -s 512 -i start_admin_sess.bin /dev/sg2 B5 01 07 FE 80 00 00 00 00 01 00 00
    c. SyncSession
            # sg_raw -r 512 -o start_admin_resp.bin /dev/sg2 A2 01 07 FE 80 00 00 00 00 01 00 00
    d. Read index #76 for HSN (e.g. 0x00) and #78-79 for TSN (e.g. 0xD501). TSN is required for payloads sent to
the device for this session and typically increments for each subsequent session.

2. Invoke the Get method to retrieve the Storage Device's MSID.
    a. In getmsid.bin
        i. Set bytes #22-23 as TSN, byte #27 as HSN
    b. Execute Get command
            # sg_raw -s 512 -i getmsid.bin /dev/sg2 B5 01 07 FE 80 00 00 00 00 01 00 00
```

...and so on

Summary

- Hardware-enabled SED drives aren't much more expensive than equivalent non-SED drives, apples-apples
- Managing encryption is relatively simple using open tools...
- As long as you have or can create "wrappers" for those tools
- With the commands in this presentation, you can script your own
- ...But coding sg_raw, while doable, is rather obscure...
- So send an email to **lustre@warpmech.com**, and we can help

WARP MECHANICS®

WARP® MECHANICS

HGST
a Western Digital brand

Questions?

Q2-2017

Josh Judd, CTO

OAK RIDGE National Laboratory