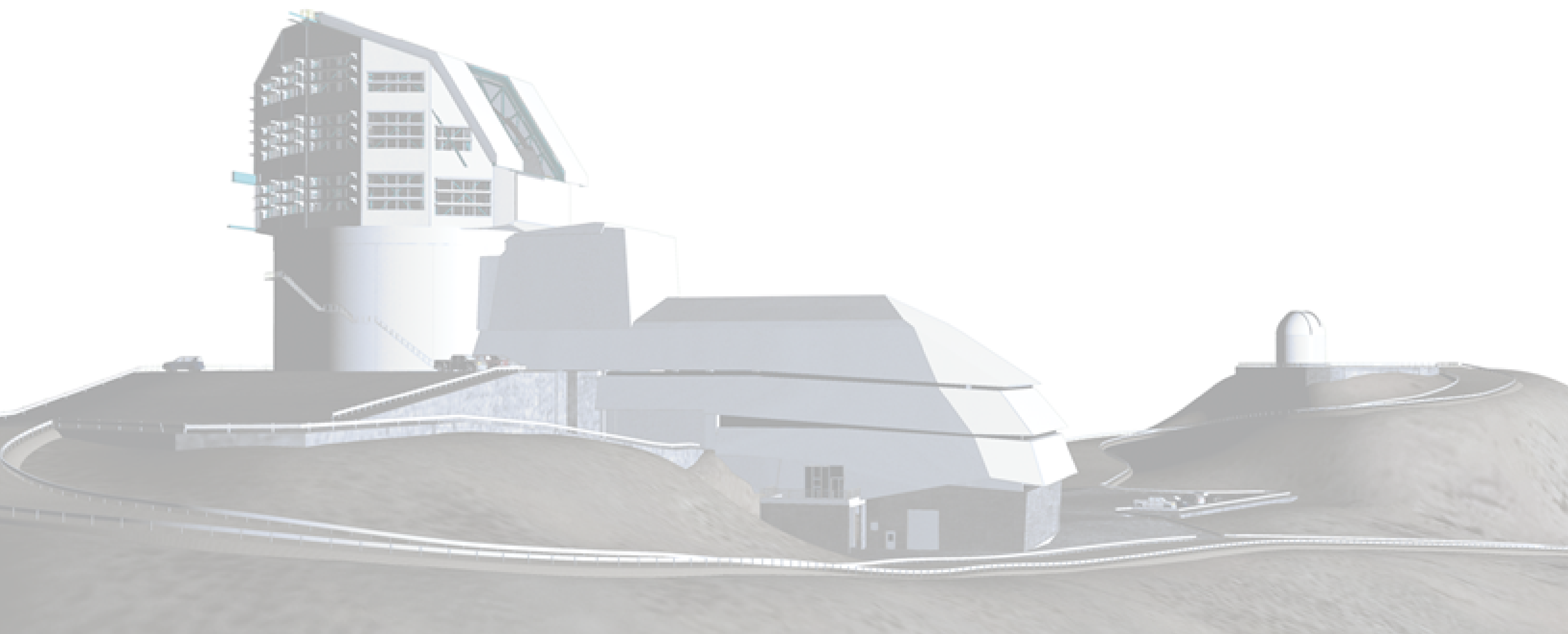
Vera C. Rubin Observatory
Rubin Observatory Project Office

CentOS System Disk Encryption

Heinrich Reinking

ITTN-048

Latest Revision: 2021-07-27



Abstract

Disk encryption rises due to the need of protecting the information by converting it into unreadable code that cannot be deciphered easily. It uses hardware or software to encrypt bit by bit all data that goes on a disk. The Linux Unified Key Setup (LUKS) is a disk encryption software, that implements a platform-independent standard on-disk format for use in various tools.

The Policy-Based Decryption (PBD) is a collection of technologies that enable unlocking encrypted root and secondary volumes of hard drives on physical and virtual machines using different methods like a user password, a Trusted Platform Module (TPM) device, a PKCS11 device connected to a system, for example, a special network server. The PBD as technology allows combining different unlocking methods into a policy creating an ability to unlock the same volume in different ways. The current implementation of the PBD in Red Hat Enterprise Linux consists of the Clevis framework and plugins called pins. Each pin provides a separate unlocking capability. For now, the only two pins available are the ones that allow volumes to be unlocked with TPM or with a network server.

The Network Bound Disk Encryption (NBDE) is a subcategory of the PBD technologies that allows binding the encrypted volumes to a special network server. The current implementation of the NBDE includes a Clevis pin for the Tang server and the Tang server itself.

Based on these tools, the Servers System Disk will be encrypted and when they boot, they will request decryption to a centralized server that withholds the Decryption key, avoiding the password prompt at boot.

Change Record

Version	Date	Description	Owner name
1	2021-05-09	First Release	Heinrich Reinking
2	2021-06-01	First Commit Concluded	Heinrich Reinking
3	2021-06-10	Performance Test	Heinrich Reinking
4	2021-07-27	Performance Test Virtual Drives over GPFS, SSD and NVMe	Heinrich Reinking

Document source location: `https://github.com/lsst-it/ittn-048`

Contents

CentOS System Disk Encryption

1 System Disk Encryption

An encrypted system disk prevents that the data contained in it can be cloned or replicated without the passphrase or authentication server. For this design, the disks will be encrypted through a kickstart passphrase and then removed once the remote Tang server is reached. If a non-authorized user gains physical access to the server:

- If halted and attempted to change the root password, the encryption passphrase prompt will be requested - which was deleted.
- If booted through a Live USB OS, the encrypted partitions remain unreadable.
- If the drive is removed/stolen, the disk's data remains cyphered.

1.1 LUKS - Linux Unified Key Setup

According to a paper subscribed by Danut Anton and Emil Simion [1], LUKS is one of the most common FDE solutions for Linux-based systems. FDE works by encrypting every single bit on a storage device, so if the user doesn't have the password, data cannot be recovered. The most common problem for FDE solutions is password management, which at what concerns this implementation, will be handled by a two-level key hierarchy. A strong master key is generated by an OS, which is used to encrypt/decrypt the hard drive. That key has to be split and encrypted with a secret user key and stored on the device, at the beginning of the memory. The advantage of this approach is that you can have multiple systems with multiple keys, allowing you to have multiple decryption Servers.

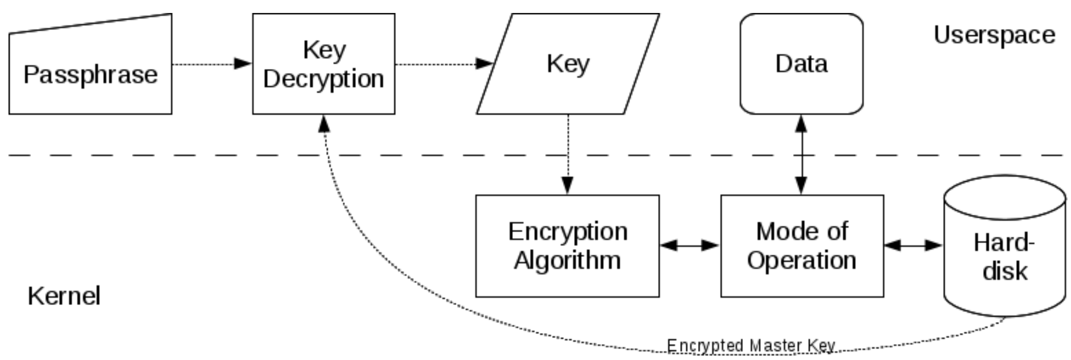


FIGURE 1: LUKS Operational Diagram

[1] https://ieeexplore-ieee-org.usm.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=8678978

1.2 Clevis

Clevis is a pluggable framework for automated decryption. It can be used to provide automate decryption of data or even automated unlocking of LUKS volumes [2]. Once Clevis has subscribed the decryption to a server, the encryption passphrase is removed, which means in a lost communication event, the server won't be able to decrypt, not even with the passphrase. To prevent this Clevis can subscribe up to 8 keys to 8 different servers/users and it can be restricted to how many of them are required as a minimum. If you set a value t=2, means that at least 2 servers have to be available at the moment of decryption.

1.3 Clevis Puppet Profile

```
#Clevis Profile
class profile::core::clevis() {
  $packages = [
    'clevis',
    'clevis-luks',
    'clevis-dracut'
  ]

  ##Add require packages
  package { $packages:
    ensure => 'present',
  }
  ->exec { '/sbin/dracut -f --regenerate-all':
    path    => ['/usr/bin', '/sbin'],
    onlyif => 'test ! -f /usr/lib/dracut/modules.d/60clevis/clevis-hook.sh'
  }
}
```

This profile installs the clevis packages needed to encrypt and manage the LUKS encryption drives. This is not quite required, because the clevis packages are being installed during provisioning, but, it grants some useful tools like 'cryptosetup' to check the subscribed Tang servers.

[2] https://github.com/latchset/clevis

2 Tang Server - Decryption Server

2.1 Tang Service

Tang [3] is a server for binding data to network presence. In simple terms: you have some data, but you only want it to be available when the system containing the data is on a certain, usually secure, network, This is where Tang comes in. First, the client gets a list of the Tang server's advertised asymmetric keys. This can happen online by a simple HTTP GET. Second, the client uses one of these public keys to generate a unique, cryptographically strong encryption key. The data is then encrypted using this key. Once the data is encrypted, the key is discarded. Some small metadata is produced as part of this operation which the client should store in a convenient location. This process of encrypting data is the provisioning step. Third, when the client is ready to access its data, it simply loads the metadata produced in the provisioning step and performs an HTTP POST to recover the encryption key. This process of encrypting data is the provisioning step.

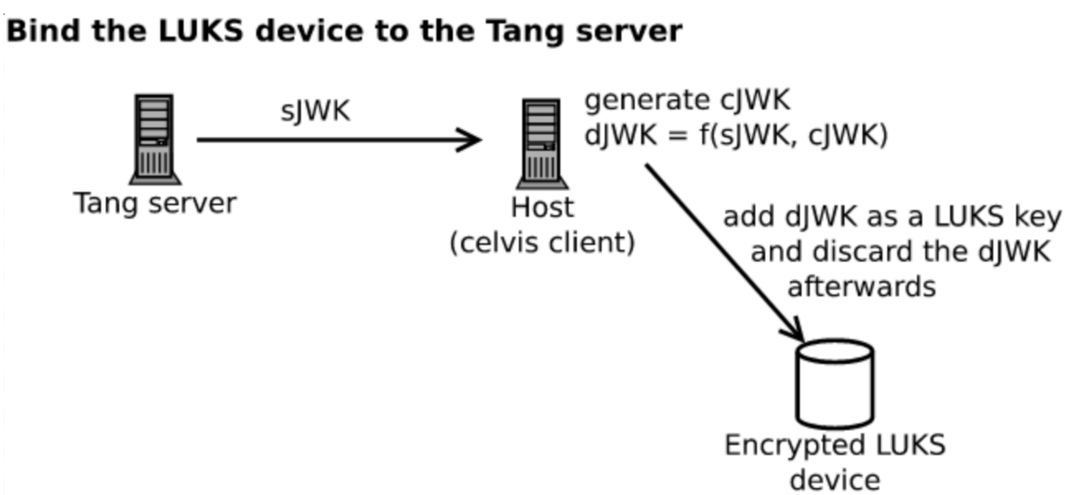


FIGURE 2: LUKS device interaction with Tang Server

[3] https://github.com/latchset/tang

2.2 Tang Puppet Profile and Role

```
# Tang Server Encryption Module

class profile::core::tang() {
  #Variables
  $packages = [
    'tang'
  ]

  #Add require packages
  package { $packages:
    ensure => 'present',
  }

  systemd::dropin_file {'override.conf':
    unit    => 'tangd.socket',
    content => @(OVERRIDE/L)
      [Socket]
      ListenStream=7500
      | OVERRIDE
  }
  # Ensure tang service is running
  ->service { 'tangd.socket':
    ensure  => 'running',
    require => Package[$packages],
  }
}
```

Tang profile handles the installation of the tangd.socket service, and then modifies it so it listens on port 7500 for incoming connections from clevis-dracut.

3 Lab Testing - Proof of Concept

3.1 Kickstart Modifications - Use of Encryption in Provisioning Template

Since the drive must be encrypted with LUKS early during the provisioning, new Kickstart Provisioning Template and Partition Tables had to be created at Foreman.

```
# Encrypted VDA - Partition Table
ignoredisk --only-use=${BOOT_DEV}
zerombr
clearpart --drives=${BOOT_DEV} --all --initlabel
part /boot     --size=1024 --asprimary --ondrive=${BOOT_DEV}
part /boot/efi --size=200  --asprimary --ondrive=${BOOT_DEV} --fstype=efi
# Use an easy passphrase, it will be removed one step later
part pv.boot   --size=1 --grow  --encrypted --passphrase=******** --ondisk=${BOOT_DEV}
volgroup ${BOOT_VG} pv.boot
logvol /              --vgname=${BOOT_VG} --size=1 --grow --name=root
```

"Encrypted VDA" initialize the System disk with two regular partitions - /boot and /boot/efi - and then a PV, a VG and a LV, been the LV encrypted through LUKS with a temporary password

```
##Kickstart - Encrypted Provisioning Template
#Packages Section
%packages
clevis-dracut
#Post Section - At ******** use the same passphrase written at the Partition Table
%post --log=/mnt/sysimage/root/install.post.log
curl -sfg http://tang01.cp.lsst.org/adv -o adv1.jws
clevis luks bind -f -k- -d /dev/vda3 \
tang '{"url":"http://tang01.cp.lsst.org","adv":"adv1.jws"}' <<< "********"
curl -sf http://tang02.cp.lsst.org/adv -o adv2.jws
clevis luks bind -f -k- -d /dev/vda3 \
tang '{"url":"http://tang02.cp.lsst.org","adv":"adv2.jws"}' \ <<< "********"
cryptsetup luksRemoveKey /dev/vda3 <<< "********"
```

In the packages section, clevis-dracut is installed, to then be used at post to communicate with a Tang server(s), subscribe to them and remove the temporary password.

3.2 Test Environment

- Two Tang servers using the tang puppet profile.
- A client with the clevis puppet profile.
- The client VM (clevis01.cp.lsst.org) is provisioned through PXE with 'Encrypted VDA' Partitioning Table and 'Kickstart Encrypted Provisioning Template'.
- During partition creation, clevis01 root partition is encrypted through LUKS with a passphrase.
- Then at packages, clevis-dracut is installed to then communicate with the Tang servers at post section.
- At post, clevis01 subscribes to the Tang servers (tang01.cp.lsst.org and tang02.cp.lsst.org) and the temporary passphrase encryption key is removed as a decryption mechanism.

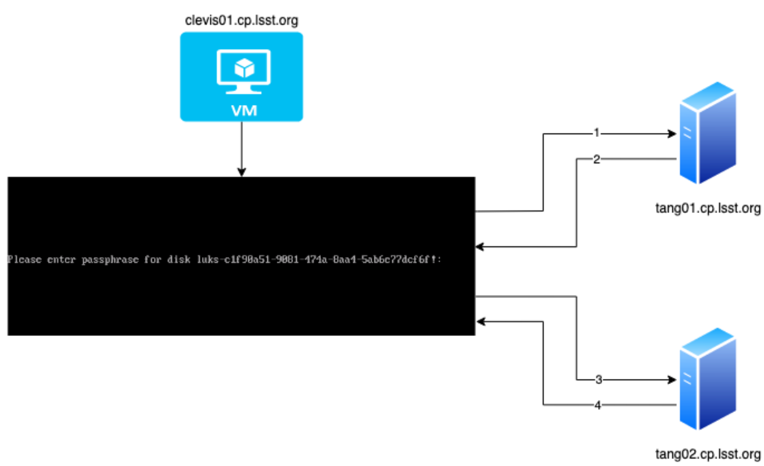


FIGURE 3: Booting procedure for an enrolled or newly enrolled client.

1. During boot, the client machine attempts to reach the first Tang server.
2. If reached, the decryption server hands over the decryption key.
3. If the first Tang server wasn't reachable, it attempts with the next one in the key slot.
4. The second Tang server sends the decryption key.

3.3 Lab Results

- The encrypted client clevis01 successfully decrypt during dracut by reaching tang01.
- The primary Tang server (tang01) was powered off and the client was able to decrypt through tang02.
- Both Tang servers were powered off and the server remains on hold requesting a passphrase (which doesn't exist) until at least one of the Tang servers is back online (Figure 3).
- For the scope of this PoC, the deletion and recreation of one or both Tang servers was not done, but presumably the client decryption would not happened and the content would be irrecoverable.
- One way of handling the loss of all Tang servers, is to add the keys to lsst-private repo, but key rotation is suggested by the documentation to increase safety.

```
2021-05-10T20:39:36.774868+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:37.365081+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:39.781504+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:40.368893+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:42.787335+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:43.373976+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:45.793299+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:46.379501+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:48.799358+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:49.386871+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:51.806121+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:52.395495+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:54.811291+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:55.398743+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:57.817439+00:00 clevis01.cp.lsst.org dracut-initqueue: Error communicating with the server!
2021-05-10T20:39:58.502717+00:00 clevis01.cp.lsst.org systemd-cryptsetup: Set cipher aes, mode xts-plain64, key size 512 bits
 for device /dev/disk/by-uuid/c1f90a51-9081-474a-8aa4-5ab6e77dcf6f.
2021-05-10T20:40:00.587670+00:00 clevis01.cp.lsst.org systemd: Started Cryptography Setup for luks-c1f90a51-9081-474a-8aa4-5a
b6e77dcf6f.
2021-05-10T20:40:00.588665+00:00 clevis01.cp.lsst.org systemd: Reached target Local Encrypted Volumes.
2021-05-10T20:40:00.589349+00:00 clevis01.cp.lsst.org systemd: Reached target System Initialization.
2021-05-10T20:40:00.589667+00:00 clevis01.cp.lsst.org systemd: Reached target Basic System.
2021-05-10T20:40:00.710597+00:00 clevis01.cp.lsst.org systemd: Found device /dev/mapper/centos_clevis01-root.
2021-05-10T20:40:00.711402+00:00 clevis01.cp.lsst.org systemd: Starting File System Check on /dev/mapper/centos_clevis01-root
...
2021-05-10T20:40:00.726521+00:00 clevis01.cp.lsst.org systemd: Started dracut initqueue hook.
2021-05-10T20:40:00.727855+00:00 clevis01.cp.lsst.org systemd: Reached target Remote File Systems (Pre).
2021-05-10T20:40:00.728090+00:00 clevis01.cp.lsst.org systemd: Reached target Remote File Systems.
2021-05-10T20:40:00.732683+00:00 clevis01.cp.lsst.org systemd-fsck: /sbin/fsck.xfs: XFS file system.
2021-05-10T20:40:00.733725+00:00 clevis01.cp.lsst.org systemd: Started File System Check on /dev/mapper/centos_clevis01-root.
2021-05-10T20:40:00.734543+00:00 clevis01.cp.lsst.org systemd: Mounting /sysroot...
2021-05-10T20:40:00.851616+00:00 clevis01.cp.lsst.org kernel: SGI XFS with ACLs, security attributes, no debug enabled
2021-05-10T20:40:00.853330+00:00 clevis01.cp.lsst.org kernel: XFS (dm-1): Mounting V5 Filesystem
2021-05-10T20:40:00.883530+00:00 clevis01.cp.lsst.org kernel: XFS (dm-1): Ending clean mount
2021-05-10T20:40:00.887378+00:00 clevis01.cp.lsst.org systemd: Mounted /sysroot.
2021-05-10T20:40:00.888099+00:00 clevis01.cp.lsst.org systemd: Reached target Initrd Root File System.
2021-05-10T20:40:00.889124+00:00 clevis01.cp.lsst.org systemd: Starting Reload Configuration from the Real Root...
2021-05-10T20:40:00.894051+00:00 clevis01.cp.lsst.org systemd: Reloading.
2021-05-10T20:40:00.960568+00:00 clevis01.cp.lsst.org systemd: Started Reload Configuration from the Real Root.
2021-05-10T20:40:00.960828+00:00 clevis01.cp.lsst.org systemd: Reached target Initrd File Systems.
2021-05-10T20:40:00.961068+00:00 clevis01.cp.lsst.org systemd: Reached target Initrd Default Target.
2021-05-10T20:40:00.961493+00:00 clevis01.cp.lsst.org systemd: Starting dracut pre-pivot and cleanup hook...
```

FIGURE 4: Access to LUKS encrypted drive while Tang server is rebooting

3.4 Performance Test - Virtual Drive over HDD

Besides securing your data, Encryption has an impact on performance as well. Every written bit of data has to be encrypted before written on disk, which impacts both the CPU and the Disk I/O. In modern CPU architectures, the impact is not as much as it was in the past, but disks do suffer consequences. To test the performance impact of encryption, we are going to use ***sysbench***, an open-source tool that performs series of tests to verify systems under intensive load. To have a baseline, two CentOS VM, with the same specs, were deployed: one encrypted with LUKS and a regular not encrypted one.

3.4.1 CPU Benchmark

Test: ***sysbench –test=cpu –cpu-max-prime=20000 run***

Encrypted

```
sysbench 1.0.17 (using system LuaJIT 2.0.4)
Running the test with following options:
Number of threads: 1
Initializing random number generator
from current time
Prime numbers limit: 20000
Initializing worker threads...
Threads started!

CPU speed:
  events per second:   314.14

General statistics:
  total time:                 10.0028s
  total number of events:     3143

  Latency (ms):
    min:                      3.16
    avg:                      3.18
    max:                      5.40
    95th percentile:          3.19
    sum:                      10000.49
Threads fairness:
  events (avg/stddev):           3143.0000/0.00
  execution time (avg/stddev):   10.0005/0.00

```

Not-Encrypted

```
sysbench 1.0.17 (using system LuaJIT 2.0.4)
Running the test with following options:
Number of threads: 1
Initializing random number generator
from current time
Prime numbers limit: 20000
Initializing worker threads...
Threads started!

CPU speed:
  events per second:   314.05

General statistics:
  total time:                 10.0025s
  total number of events:     3142

  Latency (ms):
      min:                    3.16
      avg:                    3.18
      max:                    5.21
      95th percentile:        3.19
      sum:                    9995.36
Threads fairness:
  events (avg/stddev):           3142.0000/0.00
  execution time (avg/stddev):   9.9954/0.00

```

The results help us know that the CPU architecture, threads, and processing are the same for both VM, which will help us determine the accuracy for the following tests.

3.4.2 Disk Benchmark

Test: ***time sysbench –test=fileio –file-total-size=30G –file-num=24 prepare***

Encrypted

```
sysbench 1.0.17 (using system LuaJIT 2.0.4)

24 files, 1310720Kb each, 30720Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
Creating file test_file.20
Creating file test_file.21
Creating file test_file.22
Creating file test_file.23
32212254720 bytes written in 244.25 seconds (125.77 MiB/sec).

real	4m4.275s
user	0m1.205s
sys	0m46.108s

```

Not-Encrypted

```
sysbench 1.0.17 (using system LuaJIT 2.0.4)

24 files, 1310720Kb each, 30720Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
Creating file test_file.20
Creating file test_file.21
Creating file test_file.22
Creating file test_file.23
32212254720 bytes written in 105.24 seconds (291.91 MiB/sec).

real	1m45.253s
user	0m1.201s
sys	0m50.978s

```

First, there is a preparation stage, in which several files are created, so they can be then moved, synced, copied, and deleted. Based on this operations, ***sysbench*** will reflect the Disk IO times. Yet, it is important to notice that for only writing the files, the Encrypted vs Not-Encryption rates are significantly impact: while the Not-Encrypted had a rate of 291.91 [MiB/sec], the Encrypted one was only 125.77 [MiB/sec], which result in almost tripling the amount of time required to write 30 GB.

Test: ***sysbench fileio –file-total-size=30G –file-num=24 –file-test-mode=rndrw –time=1800 –file-rw-ratio=1 –threads=16 –max-requests=0 run***

Encrypted

```
Number of threads: 16
24 files, 1.25GiB each
30GiB total file size
Block size 16KiB
Read/Write ratio for combined random IO test: 1.00
Periodic FSYNC enabled, calling fsync() each 100
    requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...
Threads started!

File operations:
    reads/s:                     2755.75
    writes/s:                    2755.75
    fsyncs/s:                    1322.96

Throughput:
    read, MiB/s:                 43.06
    written, MiB/s:              43.06

General statistics:
    total time:                  1800.0206s
    total number of events:      12301839
Latency (ms):
    min:                    0.00
    avg:                    2.34
    max:                    429.07
    95th percentile:        8.74
    sum:                    28757517.82

Threads fairness:
  events (avg/stddev):          768864.9375/3513.22
  execution time (avg/stddev):  1797.3449/0.09
```

Not-Encrypted

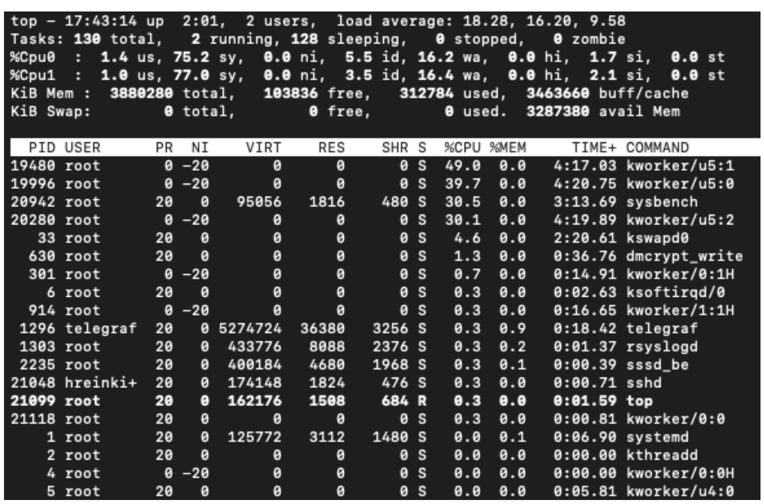
```
Number of threads: 16
24 files, 1.25GiB each
30GiB total file size
Block size 16KiB
Read/Write ratio for combined random IO test: 1.00
Periodic FSYNC enabled, calling fsync() each 100
    requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...
Threads started!

File operations:
    reads/s:                     6842.16
    writes/s:                    6842.16
    fsyncs/s:                    3284.45

Throughput:
    read, MiB/s:                 106.91
    written, MiB/s:              106.91

General statistics:
    total time:                  1800.0121s
    total number of events:      30543674
Latency (ms):
    min:                    0.00
    avg:                    0.94
    max:                    536.87
    95th percentile:        3.36
    sum:                    28764555.92

Threads fairness:
  events (avg/stddev):          1908979.6250/7247.20
  execution time (avg/stddev):  1797.7847/0.04
```



```
top - 17:43:14 up  2:01,  2 users,  load average: 18.28, 16.20, 9.58
Tasks: 130 total,   2 running, 128 sleeping,   0 stopped,   0 zombie
%Cpu0  :  1.4 us, 75.2 sy,  0.0 ni,  5.5 id, 16.2 wa,  0.0 hi,  1.7 si,  0.0 st
%Cpu1  :  1.0 us, 77.0 sy,  0.0 ni,  3.5 id, 16.4 wa,  0.0 hi,  2.1 si,  0.0 st
KiB Mem :  3880280 total,   103836 free,   312784 used,  3463660 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  3287380 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
19480 root       0 -20       0      0      0 S  49.0  0.0   4:17.03 kworker/u5:1
19996 root       0 -20       0      0      0 S  39.7  0.0   4:20.75 kworker/u5:0
20942 root      20   0   95056   1816    480 S  30.5  0.0   3:13.69 sysbench
20280 root       0 -20       0      0      0 S  30.1  0.0   4:19.89 kworker/u5:2
   33 root      20   0       0      0      0 S   4.6  0.0   2:20.61 kswapd0
  630 root      20   0       0      0      0 S   1.3  0.0   0:36.76 dmcrypt_write
  301 root       0 -20       0      0      0 S   0.7  0.0   0:14.91 kworker/0:1H
    6 root      20   0       0      0      0 S   0.3  0.0   0:02.63 ksoftirqd/0
  914 root       0 -20       0      0      0 S   0.3  0.0   0:16.65 kworker/1:1H
 1296 telegraf  20   0 5274724  36380   3256 S   0.3  0.9   0:18.42 telegraf
 1303 root      20   0  433776   8088   2376 S   0.3  0.2   0:01.37 rsyslogd
 2235 root      20   0  400184   4680   1968 S   0.3  0.1   0:00.39 sssd_be
21048 hreinki+  20   0  174148   1824    476 S   0.3  0.0   0:00.71 sshd
21099 root      20   0  162176   1508    684 R   0.3  0.0   0:01.59 top
21118 root      20   0       0      0      0 S   0.3  0.0   0:00.81 kworker/0:0
    1 root      20   0  125772   3112   1480 S   0.0  0.1   0:06.90 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    4 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    5 root      20   0       0      0      0 S   0.0  0.0   0:05.81 kworker/u4:0
```

FIGURE 5: CPU Load while running sysbench

To get a more clear view of the performance results, let's arrange them in a more comfortable way:

Section	Metric	Encrypted	Not-Encrypted	Percentage
File Operations	reads/s	2755.75	6842.1600	Decreased 59.72
	writes/s	2755.75	6842.1600	Decreased 59.72
	fsyncs/s	1322.96	3284.45	Decreased 59,72
Throughput	reads MiB/s	43.06	106.91	Decreased 59.72
	written MiB/s	43.06	106.91	Decreased 59.72
General Stadistics	total time	1800.0206	1800.0121	Decreased 0.000472
	total number of events	12301839	30543675	Increased 249.29
Latency	min	0	0	0
	avg	2.34	0.94	Decreased 59.83
	max	429.07	536.87	Increased 20.08
	95th Percentile	8.74	3.36	Decreased 38.44
	sum	28757517.82	28764555.92	Increased 0.02
Threads fairness	events (avg/stddev)	768864.9375/3513.22	1908979.6250/7247.2	Increased 40.28/Increased 106.28
	execution time (avg/stddev)	1797.3449/0.09	1797.7847/0.04	Increased 0.02/Decreased 55.56

As shown in Figure 5, during the performance test at the Encrypted VM, the load on the CPU was so high (18.28) that the ssh connection was terminated and the network connection lost. As far as the results goes:

- **Files Operations:** Reads, Writes, and File Sync operations suffered a 49.72 percent due to encryption.
- **Throughput:** lost a 59.72 percentage both at write and read operations while Encrypting.
- **General Statistics:** The number of events compared with the total time for them to occur, shows that, in the same amount of time, 249.29 percentage more events happened at the Not-Encrypted VM.
- **Latency:** In the same time period, the average for the Encrypted vs Not-Encrypted suffered a 59.83 percentage more latency, while the Max increased in a 20.08 percent.
- **Threads Fairness:** The number of events per thread, were 40.28 percent higher in a Not-Encrypted system than in an Encrypted one.

3.5 Virtual Drive over HDD Conclusions - Pros and Cons

- Encrypting a system, impacts in average a 60 percent payload over the read/write operations.
- The encrypted client clevis01 successfully decrypt during dracut by reaching tang01.
- The primary Tang server (tang01) was powered off and the client was able to decrypt through tang02.
- For the scope of this PoC, the deletion and recreation of one or both Tang servers was not done, but presumably, the client decryption would not happen and the content would be irrecoverable.
- One way of handling the loss of all Tang servers is to add the keys to lsst-private repo, but key rotation is suggested by the documentation to increase safety.

3.6 Performance Test - Virtual Drive over GPFS/Gluster

General Parallel File System [4] is a high-performance clustered file system software developed by IBM. It provides concurrent, high-speed file access to applications executing on multiple nodes of clusters. Gluster [5] is a scalable network filesystem suitable for data-intensive tasks such as cloud storage and media streaming. It is free and open-source and can utilize standard hardware. Since GPFS is not an open-source solution, it can not be tested for encryption performance, but GlusterFS offers the same suitable performance attributes from GPFS, and it can be easily mounted and tested. Then, for the GPFS Encryption Test, we are going to try a VM over a GlusterFS.

3.6.1 Environment Setup - GlusterFS

- Over one ESXi hypervisor, with HDD storage, three VMs were deployed: gluster01, gluster02, and gluster03.
- Each VM has a 50GB storage partition and a replica three configuration, which means that meanwhile, one VM is still running, the data will be accessible.
- A glusterFS pool was generated and then mounted into another ESXi hypervisor through NFS.
- The performance tests were performed simultaneously in both VMs (encrypted and not-encrypted) to hit as much as possible the gluster storage.

[4] https://es.wikipedia.org/wiki/General_Parallel_File_System
[5] https://docs.gluster.org/en/latest/

3.6.2 CPU Benchmark

Test: ***sysbench –test=cpu –cpu-max-prime=20000 run***

Encrypted

```
Number of threads: 1
Prime numbers limit: 20000
Initializing worker threads...
Threads started!

CPU speed:
    events per second:   289.40
General statistics:
    total time:                          10.0012s
    total number of events:              2895
Latency (ms):
    min:                                    3.05
    avg:                                    3.45
    max:                                    4.31
    95th percentile:                        3.62
    sum:                                 9999.21
Threads fairness:
    events (avg/stddev):           2895.0000/0.00
    execution time (avg/stddev):   9.9992/0.00

```

Not-Encrypted

```
Number of threads: 1
Prime numbers limit: 20000
Initializing worker threads...
Threads started!

CPU speed:
    events per second:   289.35
General statistics:
    total time:                          10.0032s
    total number of events:              2895
Latency (ms):
    min:                                    2.94
    avg:                                    3.45
    max:                                    3.95
    95th percentile:                        3.62
    sum:                                10000.67
Threads fairness:
    events (avg/stddev):           2895.0000/0.00
    execution time (avg/stddev):   10.0007/0.00

```

3.6.3 Disk Benchmark

Test: ***time sysbench –test=fileio –file-total-size=30G –file-num=24 prepare***

Encrypted

```
24 files, 1310720Kb each, 30720Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
#
#OMITTING OUTPUT FROM
#FILE2 TO FILE21
#
Creating file test_file.22
Creating file test_file.23
32212254720 bytes written in 79.41 seconds (386.86 MiB/sec).

real    1m19.840s
user    0m0.810s
sys     0m16.460s

```

Not-Encrypted

```
24 files, 1310720Kb each, 30720Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
#
#OMITTING OUTPUT FROM
#FILE2 TO FILE21
#
Creating file test_file.22
Creating file test_file.23
32212254720 bytes written in 75.20 seconds (408.50 MiB/sec).

real  1m16.082s
user  0m0.833s
sys 0m19.254s

```

Test: ***sysbench fileio –file-total-size=30G –file-num=24 –file-test-mode=rndrw –time=1800 –file-rw-ratio=1 –threads=16 –max-requests=0 run***

Encrypted

```
Number of threads: 16
Initializing random number generator from current
    time
24 files, 1.25GiB each
30GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.00
Periodic FSYNC enabled, calling fsync() each 100
    requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!
File operations:
    reads/s:                      276.79
    writes/s:                     276.79
    fsyncs/s:                     133.06

Throughput:
    read, MiB/s:                  4.32
    written, MiB/s:               4.32

General statistics:
    total time:                          1800.0429s
    total number of events:              1235604

Latency (ms):
    min:                                    0.00
    avg:                                   23.31
    max:                                  964.45
    95th percentile:                      125.52
    sum:                             28798820.02

Threads fairness:
    events (avg/stddev):          77225.2500/438.15
    execution time (avg/stddev):  1799.9263/0.01
```

Not-Encrypted

```
Number of threads: 16
Initializing random number generator from current
    time
24 files, 1.25GiB each
30GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.00
Periodic FSYNC enabled, calling fsync() each 100
    requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!
File operations:
    reads/s:                      309.04
    writes/s:                     309.04
    fsyncs/s:                     148.54

Throughput:
    read, MiB/s:                  4.83
    written, MiB/s:               4.83

General statistics:
    total time:                          1800.0651s
    total number of events:              1379587

Latency (ms):
    min:                                    0.00
    avg:                                   20.88
    max:                                 1263.24
    95th percentile:                      110.66
    sum:                             28799044.27

Threads fairness:
    events (avg/stddev):          86224.1875/802.36
    execution time (avg/stddev):  1799.9403/0.02
```

Comparison Table

Section	Metric	Encrypted	Not-Encrypted	Percentage
File Operations	reads/s	276.79	308.04	Decreased 10.44
	writes/s	279.79	309.04	Decreased 10.44
	fsyncs/s	133.06	148.54	Decreased 10.42
Throughput	reads MiB/s	4.32	4.83	Decreased 10.56
	written MiB/s	4.32	4.83	Decreased 10.56
General Stadistics	total time	1800.0429s	1800.0651s	Increased 10.42
	total number of events	1235604	1379587	Increased 11.65
Latency	min	0.00	0.00	0
	avg	23.31	20.88	Decreased 10.42
	max	964.45	1263.24	Increased 23.65
	95th Percentile	125.52	110.66	Decreased 11.84
	sum	287944820.02	28799044.27	Increased 0.00078
Threads fairness	events (avg/stddev)	77225.25/438.15	86224.1875/802.36	Increased 10.44/Increased 83.12
	execution time (avg/stddev)	1799.9263/0.01	1799.9403/0.02	Increased 0.00078/Increased 100

Impact

- **Files Operations:** Reads, Writes, and File Sync operations suffered a 10.44 percent due to encryption.
- **Throughput:** Lost a 10.56 percentage both at write and read operations while Encrypting.
- **General Statistics:** The number of events compared with the total time for them to occur shows that 11.65 percentage more events happened at the Not-Encrypted VM in the same amount of time.
- **Latency:** In the same period, the average for the Encrypted vs. Not-Encrypted suffered a 10.42 percentage more latency, while the Max increased by 23.65 percent.
- **Threads Fairness:** The number of events per thread was 10.44 percent higher in a Not-Encrypted system than in an Encrypted one.

3.7 Virtual Drive over GPFS/Gluster Conclusions - Pros and Cons

- The Read/Write operations are significantly lower due to the cluster network-based storage compared to the direct disk RW operations (HDD, SDD, and NVMe).
- Since the RW decreased operations, the performance impact is not as high as in the direct disk but still has an effect between Encrypted vs. Not-Encrypted (in favor of Not-Encrypted).

3.8 Performance Test - Virtual Drive over SDD

3.8.1 CPU Benchmark

Test: ***sysbench –test=cpu –cpu-max-prime=20000 run***

Encrypted

```
[root@encrypted ~]# sysbench --test=cpu --cpu-max-prime=20000 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.17 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   279.21

General statistics:
    total time:                          10.0006s
    total number of events:              2794

Latency (ms):
         min:                                    3.04
         avg:                                    3.58
         max:                                   21.07
         95th percentile:                        9.06
         sum:                                 9997.23

Threads fairness:
    events (avg/stddev):           2794.0000/0.00
    execution time (avg/stddev):   9.9972/0.00

```

Not-Encrypted

```
[root@not-encrypted ~]# sysbench --test=cpu --cpu-max-prime=20000 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.17 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   327.59

General statistics:
    total time:                          10.0011s
    total number of events:              3277

Latency (ms):
         min:                                    3.04
         avg:                                    3.05
         max:                                    4.47
         95th percentile:                        3.07
         sum:                                 9998.90

Threads fairness:
    events (avg/stddev):           3277.0000/0.00
    execution time (avg/stddev):   9.9989/0.00

```

3.8.2 Disk Benchmark

Test: ***time sysbench –test=fileio –file-total-size=30G –file-num=24 prepare***

Encrypted

```
24 files, 1310720Kb each, 30720Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
Creating file test_file.20
Creating file test_file.21
Creating file test_file.22
Creating file test_file.23
32212254720 bytes written in 444.62 seconds (69.09
    MiB/sec).

real    7m24.647s
user    0m1.065s
sys     0m24.865s
```

Not-Encrypted

```
24 files, 1310720Kb each, 30720Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
Creating file test_file.20
Creating file test_file.21
Creating file test_file.22
Creating file test_file.23
32212254720 bytes written in 308.17 seconds (99.69
    MiB/sec).

real  5m8.190s
user  0m1.051s
sys 0m24.147s
```

Test: ***sysbench fileio –file-total-size=30G –file-num=24 –file-test-mode=rndrw –time=1800 –file-rw-ratio=1 –threads=16 –max-requests=0 run***

Encrypted

```
Running the test with following options:
Number of threads: 16
Initializing random number generator from current
    time


Extra file open flags: (none)
24 files, 1.25GiB each
30GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.00
Periodic FSYNC enabled, calling fsync() each 100
    requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                    2514.25
    writes/s:                   2514.25
    fsyncs/s:                   1207.04

Throughput:
    read, MiB/s:                39.29
    written, MiB/s:             39.29

General statistics:
    total time:                          1800.0455s
    total number of events:              11223964

Latency (ms):
    min:                                      0.00
    avg:                                      2.56
    max:                                     54.50
    95th percentile:                         10.09
    sum:                               28767929.92

Threads fairness:
    events (avg/stddev):          701497.7500/2374.58
    execution time (avg/stddev):  1797.9956/0.07
```

Not-Encrypted

```
Running the test with following options:
Number of threads: 16
Initializing random number generator from current
    time


Extra file open flags: (none)
24 files, 1.25GiB each
30GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.00
Periodic FSYNC enabled, calling fsync() each 100
    requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                    5861.03
    writes/s:                   5861.03
    fsyncs/s:                   2813.50

Throughput:
    read, MiB/s:                91.58
    written, MiB/s:             91.58

General statistics:
    total time:                          1800.0202s
    total number of events:              26163995

Latency (ms):
    min:                                      0.00
    avg:                                      1.10
    max:                                    258.42
    95th percentile:                          5.57
    sum:                               28768693.60

Threads fairness:
    events (avg/stddev):          1635249.6875/3619.10
    execution time (avg/stddev):  1798.0434/0.03
```

Comparison Table

Section	Metric	Encrypted	Not-Encrypted	Percentage
File Operations	reads/s	2514.25	5861.03	Increased 57.10
	writes/s	2514.25	5861.03	Increased 57.10
	fsyncs/s	1207.04	2813.50	Increased 57.10
Throughput	reads MiB/s	39.29	91.58	Increased 57.10
	written MiB/s	39.29	91.58	Increased 57.10
General Stadistics	total time	1800.0455	1800.0202	Decreased 0.001406
	total number of events	11223964	26163995	Increased 233.11
Latency	min	0.00	0.00	0.00
	avg	2.56	1.10	Decreased 57.03
	max	54.5	257.42	Increased 474.17
	95th Percentile	10.09	5.57	Decreased 55.20
	sum	28767929.92	28768693.60	Increased 0.00265
Threads fairness	events (avg/stddev)	701497.7500/2374.5800	1635248.6875/3619.1000	Increased 42.90/Increased 52.41
	execution time (avg/stddev)	1797.9954/0.07	1798.0434/0.03	Increased 0.0027/Decreased 57.14

Impact

- **Files Operations:** Reads, Writes, and File Sync operations suffered a 57.10 percent due to encryption.
- **Throughput:** Lost a 57.10 percentage both at write and read operations while Encrypting.
- **General Statistics:** The number of events compared with the total time for them to occur shows that 233.11 percentage more events happened at the Not-Encrypted VM in the same amount of time.
- **Latency:** In the same period, the average for the Encrypted vs. Not-Encrypted suffered a 57.03 percentage more latency, while the Max increased by 4743.17 percent.
- **Threads Fairness:** The number of events per thread was 42.90 percent higher in a Not-Encrypted system than in an Encrypted one.

3.9 Virtual Drive over SDD Conclusions - Pros and Cons

- The SDD vs. the HDD shows a high impact over Encrypted vs. Not-Encrypted, being both around 60 percent.
- VM overlay shows that even though one VM was over HDD and the other over SDD, Files Operations shows similar RW rates.

3.10 Performance Test - Virtual Drive over SSD NVMe

3.10.1 CPU Benchmark

Test: ***sysbench –test=cpu –cpu-max-prime=20000 run***

Encrypted

```
sysbench 1.0.17 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   622.87
General statistics:
    total time:                          10.0009s
    total number of events:              6230
Latency (ms):
    min:                                    1.58
    avg:                                    1.61
    max:                                    1.92
    95th percentile:                        1.64
    sum:                                 9999.43
Threads fairness:
    events (avg/stddev):           6230.0000/0.00
    execution time (avg/stddev):   9.9994/0.00

```

Not-Encrypted

```
sysbench 1.0.17 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   622.71
General statistics:
    total time:                          10.0004s
    total number of events:              6228
Latency (ms):
         min:                                    1.58
         avg:                                    1.61
         max:                                    3.65
         95th percentile:                        1.64
         sum:                                 9998.94
Threads fairness:
    events (avg/stddev):           6228.0000/0.00
    execution time (avg/stddev):   9.9989/0.00

```

3.10.2 Disk Benchmark

Test: ***time sysbench –test=fileio –file-total-size=30G –file-num=24 prepare***

Encrypted

```
sysbench 1.0.17 (using system LuaJIT 2.0.4)

24 files, 1310720Kb each, 30720Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
Creating file test_file.20
Creating file test_file.21
Creating file test_file.22
Creating file test_file.23
32212254720 bytes written in 68.30 seconds (449.76 MiB/sec).

real    1m8.309s
user    0m0.061s
sys     0m10.984s
```

Not-Encrypted

```
sysbench 1.0.17 (using system LuaJIT 2.0.4)

24 files, 1310720Kb each, 30720Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
Creating file test_file.20
Creating file test_file.21
Creating file test_file.22
Creating file test_file.23
32212254720 bytes written in 58.35 seconds (526.51 MiB/sec).

real    0m58.350s
user    0m0.065s
sys     0m10.842s
```

Test: ***sysbench fileio –file-total-size=30G –file-num=24 –file-test-mode=rndrw –time=1800 –file-rw-ratio=1 –threads=16 –max-requests=0 run***

Encrypted

```
Number of threads: 16
Initializing random number generator from current time


Extra file open flags: (none)
24 files, 1.25GiB each
30GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.00
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      7753.55
    writes/s:                     7753.54
    fsyncs/s:                     3721.91

Throughput:
    read, MiB/s:                  121.15
    written, MiB/s:               121.15

General statistics:
    total time:                          1800.0066s
    total number of events:              34612150

Latency (ms):
    min:                                     0.00
    avg:                                     0.83
    max:                                   213.34
    95th percentile:                         2.86
    sum:                              28773912.07

Threads fairness:
  events (avg/stddev):          2163259.3750/2817.00
  execution time (avg/stddev):  1798.3695/0.03
```

Not-Encrypted

```
Number of threads: 16
Initializing random number generator from current time


Extra file open flags: (none)
24 files, 1.25GiB each
30GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.00
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      16655.75
    writes/s:                     16655.75
    fsyncs/s:                     7994.97

Throughput:
    read, MiB/s:                  260.25
    written, MiB/s:               260.25

General statistics:
    total time:                          1800.0064s
    total number of events:              74351698

Latency (ms):
    min:                                     0.00
    avg:                                     0.39
    max:                                   222.88
    95th percentile:                         1.55
    sum:                              28756585.90

Threads fairness:
  events (avg/stddev):          4646981.1250/35144.16
  execution time (avg/stddev):  1797.2866/0.05
```

Comparison Table

Section	Metric	Encrypted	Not-Encrypted	Percentage
File Operations	reads/s	7753.55	16655.75	Decreased 53.45
	writes/s	7753.54	16655.75	Decreased 53.45
	fsyncs/s	3721.91	7994.97	Decreased 53.45
Throughput	reads MiB/s	121.15	260.25	Decreased 53.45
	written MiB/s	121.15	260.25	Decreased 53.45
General Stadistics	total time	1800.0066	1800.0064	Decreased 0.000011
	total number of events	34612150	74351698	Increased 214.81
Latency	min	0	0	0.00
	avg	0.83	0.39	Increased 53.01
	max	213.34	222.88	Increased 4.47
	95th Percentile	2.86	1.55	Decreased 54.20
	sum	28773912.07	28756584.90	Increased 0.06025
Threads fairness	events (avg/stddev)	2163259.3750/2817.00	4646981.1250/35144.16	Increased 46.55/Increased 1147.57
	execution time (avg/stddev)	1798.3695/0.03	1797.2866/0.05	Decreased 0.0603/Increased 66.67

Impact

- **Files Operations:** Reads, Writes, and File Sync operations suffered a 54 percent due to encryption.
- **Throughput:** Lost a 54 percentage both at write and read operations while Encrypting.
- **General Statistics:** The number of events compared with the total time for them to occur shows that, in the same amount of time, 215 percentage more events happened at the Not-Encrypted VM.
- **Latency:** In the same period, the average for the Encrypted vs. Not-Encrypted suffered a 53 percentage more latency, while the Max increased by 4.5 percent.
- **Threads Fairness:** The number of events per thread, were 46.55 percent higher in a Not-Encrypted system than in an Encrypted one.

3.11 Virtual Drive SSD NVMe Conclusions - Pros and Cons

- The higher the speed rate from the disk, the less impact encryption produces over the drive.
- Average Latency increases over the Encrypted disk VM.
- The standard deviation over the events of the thread is abnormally high in the Not-Encrypted drive.

A Acronyms

Acronym	Description
CPU	Central Processing Unit
FDE	Full Disk Encryption
GB	Gigabyte
GPFS	General Parallel File System
HDD	Hard Drive Disk
HTTP	HyperText Transfer Protocol
IBM	International Business Machines
LUKS	Linux Unified Key Setup
LV	Logical Volume
NBDE	Network Bound Disk Encryption
NFS	Network File System
NVMe	Non Volatile Memory Express
OS	Operating System
PBD	Policy-Based Decryption
PMO	Project Management Office
PV	Physical Volume
PoC	Proof of Concept
SSD	Solid State Drive
TPM	Trusted Platform Module
USB	Universal Serial Bus
VDA	Virtual Drive A
VG	Volume Group
VM	Virtual Machine