

Integrating the MKV Block Cipher into the ZFS File System

DOI:<https://doi.org/10.54654/isj.v2i28.6405>

Pham Van Luc*, Hoang Thu Phuong, Tran Huy Phuong,
Hoang Xuan Thanh, Thai Doan Sang

Abstract— In the context of ever-growing digital data in both scale and value, the need to ensure data confidentiality and integrity has become essential for modern storage systems. Among advanced storage solutions, TrueNAS, built on ZFS, stands out for its data integrity verification and encryption capabilities. Based on an analysis of ZFS characteristics, this article evaluates the feasibility of integrating MKV, a Vietnamese post-quantum-resistant block cipher, to enhance data security. Experimental results show that the proposed MKV encryption mechanism achieves a write bandwidth of 390 MB/s, compared with 608 MB/s for AES encryption and 774 MB/s for the non-encrypted configuration, while maintaining low CPU utilization. Although the proposed mechanism introduces performance overhead, the results demonstrate its feasibility for applications requiring enhanced data confidentiality with an acceptable trade-off between security and storage performance.

Tóm tắt— Trong bối cảnh dữ liệu số ngày càng gia tăng về quy mô và giá trị, nhu cầu bảo vệ tính bí mật và toàn vẹn của dữ liệu trở thành yêu cầu thiết yếu đối với các hệ thống lưu trữ hiện đại. Trong số các giải pháp lưu trữ tiên tiến, TrueNAS với thành phần cốt lõi là ZFS nổi bật với khả năng kiểm tra toàn vẹn dữ liệu và hỗ trợ mã hóa. Trên cơ sở phân tích các đặc điểm của ZFS, bài báo này nghiên cứu về khả năng tích hợp MKV – thuật toán mã khối kháng thám mã lượng tử của Việt Nam vào hệ thống tệp ZFS nhằm nâng cao mức độ an toàn cho dữ liệu lưu trữ. Kết quả thực nghiệm cho thấy cơ chế mã hóa MKV được đề xuất đạt băng thông ghi 390 MB/s, so với 608 MB/s của mã hóa AES và 774 MB/s của cấu hình không mã hóa, đồng thời duy trì mức sử dụng CPU thấp. Mặc dù

cơ chế đề xuất tạo ra chi phí suy giảm hiệu năng, các kết quả cho thấy giải pháp này có tính khả thi đối với các ứng dụng yêu cầu tăng cường bảo mật dữ liệu, đồng thời duy trì sự cân bằng phù hợp giữa mức độ an toàn và hiệu năng lưu trữ.

Keywords— TrueNAS; ZFS; MKV block cipher; stored data; quantum cryptanalysis.

Từ khóa— TrueNAS; ZFS; mã khối MKV; dữ liệu lưu trữ; thám mã lượng tử.

I. INTRODUCTION

ZFS (Zettabyte File System) is an advanced file system developed by Sun Microsystems, now owned by Oracle, with many powerful features that help manage data more efficiently, securely, and flexibly in modern storage environments, such as:

Snapshots and Clones: ZFS allows users to create snapshots, which are unchanged copies of the file system at a specific point in time, making it easy to revert to a previous state in the event of an error. Editable clones created from snapshots allow access to data at a previous point in time, enabling editing, checking, or copying necessary data without restoring the entire system from the backup, resulting in faster and more flexible data recovery. This feature also helps protect data from ransomware.

RAID-Z: This is a method of allocating data across multiple locations, enhancing fault tolerance and improving performance. RAID-Z separates parity information across drives to ensure normal operation even if one drive fails. Additionally, in traditional RAID 5/6, when data is written, the system only updates a portion of the data and the corresponding parity. If there is a sudden power outage or hardware failure during writing, some disks may have new data written while others (containing

This manuscript was received on May 28, 2026. It was reviewed on July 28, 2026, revised August 10, 2026 and accepted August 20, 2026.

* Corresponding author

parity) retain old data. This results in a mismatch in parity, corrupting the array structure and making data recovery impossible. RAID-Z overcomes this problem, improving the reliability of data recovery.

RAID-Z2 and RAID-Z3: RAID-Z2 and RAID-Z3 provide enhanced fault tolerance for large-scale storage systems. RAID-Z2 is capable of continuing operation even when two drives fail simultaneously, while RAID-Z3 can tolerate up to three drive failure without data loss, thereby enhancing overall system security.

Automatic data compression: ZFS supports data compression during the write process, thus reducing the storage space requirements. Users can enable or disable compression, or select an appropriate compression algorithm based on the specific use case.

Data duplication prevention: By verifying the hash values of data blocks, ZFS can identify and eliminate duplicate data, which helps optimize storage capacity.

Sending and receiving: ZFS allows the transmission of file system snapshots over a network to another server, supporting remote backups, inter-server data synchronization, and data migration to cloud storage.

Data protection: ZFS integrates a fine-grained permission model and supports access control lists, enabling control over data modification rights. ZFS also provides a read-only mode to prevent data from being altered. Furthermore, ZFS on Solaris supports file encryption, which in turn increases the security of critical data.

Besides, ZFS is easy to use due to the integration of file system and disk management on a single platform, which simplifies configuration and operation while enabling flexible capacity expansion by adding drives without re-partitioning.

In the digital age, data is not only a strategic asset for individuals and organizations but also a primary target for numerous cybersecurity threats. As a result, the need to protect data confidentiality and integrity has become a

mandatory requirement in modern storage systems. To address this need, advanced storage solutions like TrueNAS have been developed, with ZFS playing a core role in data management and protection. Owing to its outstanding advantages, ZFS has become a file system worthy of further research and implementation.

In this article, the step-by-step integration of the MKV block cipher algorithm, which is designed to resist attacks from quantum computers, into the ZFS component of TrueNAS is presented, along with experiments comparing system performance when using this encryption algorithm and when no encryption is applied.

The rest of the article is organized as follows. Section II presents the background knowledge of the ZFS file system and the MKV algorithm. Section III reviews related work. Section IV analyzes data security in ZFS and describes the integration process of the MKV algorithm into the TrueNAS system. Section V presents the experimental results, and Section VI concludes the article.

II. BACKGROUND

A. ZFS file system

1. ZFS file system architecture

The groundbreaking change introduced by ZFS lies in the reorganization of the roles of components in the storage system software [1]. In a traditional file system, the architecture is divided into separate layers. The device driver provides a block device interface to the volume manager, which in turn supplies a block device interface to the file system. Finally, the file system exposes vnode operation functions to the system call layer.

Meanwhile, ZFS employs an integrated architecture in which key components such as the Storage Pool Allocator (SPA), Data Management Unit (DMU), and ZFS POSIX Layer (ZPL) operate within a unified system, replacing both the traditional storage pool manager and file system. Specifically, the device driver provides the block devices directly to the SPA, which manages all storage space and handles I/O operations. The SPA then supplies

virtual address blocks to the DMU, where they are converted into object-based transactional interfaces for the ZPL. Finally, the ZPL implements the POSIX file system on the DMU objects and exposes vnode operations to the system call layer.

The ZFS file system architecture reduces overlap between abstraction layers, thereby increasing interoperability between components while improving data integrity and management efficiency.

Figure 1 below illustrates the differences between a traditional file system architecture and the ZFS file system, where the dark blue boxes indicate the key components of ZFS.

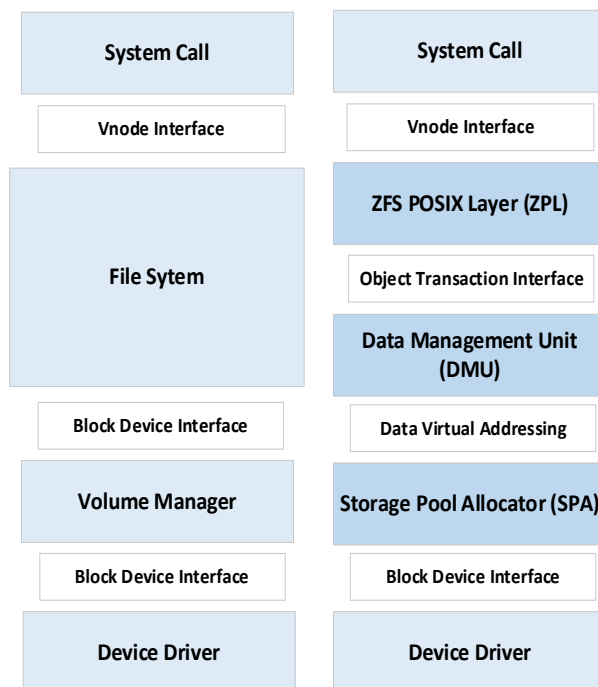


Figure 1. Traditional file system architecture (left) and ZFS file system architecture (right)

2. ZFS on-disk organization

The SPA component of ZFS manages the virtual devices (vdevs) that constitute ZFS storage pools. Each physical vdev is assigned a vdev label that describes the device and other related virtual devices [2, 3]. Inside each vdev label is an array of uberblocks that store block pointers indicating the locations of data blocks on disk [3]. The uberblock provides access to the

pool data and verifies its integrity. It is self-checksummed, and is updated atomically [2].

ZFS is an object-based transactional file system. The entire ZFS file system is represented as objects and object sets, which are formed by organizing the objects as arrays [4]. The DMU component, in turn, is responsible for grouping data blocks into these objects [3].

On disk, each object is represented by a `dnode_phys_t` structure, while each object set is represented by an `objset_phys_t` structure. The data in the object set pointed to by `objset_phys_t` consists of an array of `dnode_phys_t` structures. The `dnode_phys_t` structure contains a special buffer in the `dn_bonus` field, which is used to store metadata for each ZFS object [4].

These objects are stored on disk in blocks of data and linked together via block pointers. In ZFS, block pointers are represented by a special structure called `blkptr_t`. The `blkptr_t` structure contains important information about the referenced data block or sequence of blocks, such as the checksum algorithm, the compression algorithm applied to the next block, and the logical size before and after compression. It also indicates whether the block is a gang block. A gang block is a ZFS special structure used when a large data block cannot be allocated as a single contiguous region on disk due to fragmentation or insufficient free space. It is organized as a tree-like structure of block pointers, where the data is split into smaller blocks referenced by a hierarchy of pointers, allowing ZFS to store data that exceeds the limits of a conventional contiguous block chain [4].

Another important structure is the ZAP (ZFS Attribute Processor) object, which is used to store various attributes to handle arbitrary mappings between names and objects within an object set. Common attributes include the contents of a directory or pool-wide metadata [4]. The ZAP module manages these objects, while a ZAP object itself is implemented as a DMU object used for attribute storage [3].

The on-disk organization of ZFS is illustrated in Figure 2 below [3], where a dataset consists of multiple object sets together with free

space management and snapshot information. These datasets are further grouped by the dataset directory, which also provides attributes such as quotas and compression [4].

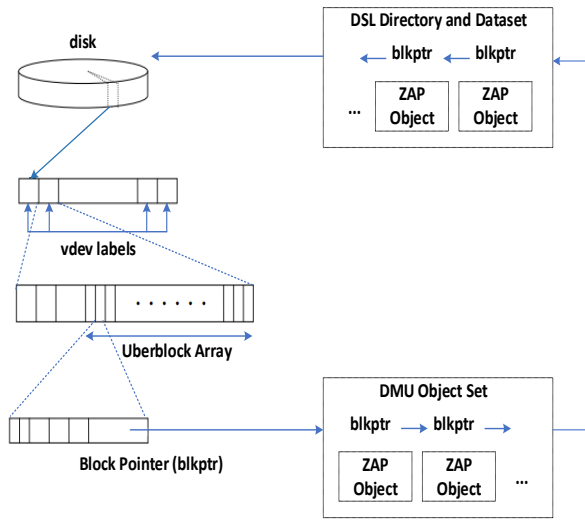


Figure 2. ZFS on-disk organization

B. The MKV Block Cipher

MKV (Mã khối Việt) developed by the Vietnam Government Information Security Commission (VGISC) is the first civilian block encryption algorithm officially standardized by the Ministry of Science and Technology of Vietnam according to the national standard TCVN 14263:2024 [5].

The MKV block cipher algorithm provides two block-size variants: a 256-bit version designed to achieve post-quantum security and a 128-bit version for the transition period. Each variant support three key length options, offering flexible security levels suitable for application developers. The MKV block cipher has achieved several notable design characteristics, as follows [6]:

- Structure: MKV employs an SPN-based architecture derived from the FLC scheme, achieving provable security under the Luby–Rackoff model. In addition, the cipher adopts the FLC-SDS structure, which provides practical resistance against differential and linear cryptanalysis.

- Cryptographic components: The block cipher employs 8-bit substitution boxes to provide strong confusion and MDS matrices to

achieve maximum diffusion. These cryptographic components possess strong security characteristics and are designed with consideration for implementation efficiency in both software and hardware.

- Key schedule: MKV employs an iterative key-generation mechanism that provides both theoretical structural security and practical resistance against related-key differential cryptanalysis.

- Resistance to practical cryptanalytic attacks: Besides its provable security against the two most common forms of cryptanalysis – differential and linear cryptanalysis – MKV is also designed to resist boomerang attacks, integral cryptanalysis, algebraic cryptanalysis, related-key differential cryptanalysis, impossible differential cryptanalysis, etc.

- Performance: The algorithm ensures efficient execution suitable for common information security applications in the civilian sector, both in hardware and software.

This modular design enables MKV to operate efficiently within Authenticated Encryption with Associated Data (AEAD) modes, making it compatible with block-level encryption pipelines in advanced file systems such as OpenZFS.

III. RELATED WORK

The integration of non-standard or national cryptographic algorithms into enterprise-grade file systems is a growing trend driven by data sovereignty and localized security requirements. Standard implementations of OpenZFS natively support only the Advanced Encryption Standard (AES) in GCM and CCM modes, which exposes the storage infrastructure to single-point-of-failure risks if vulnerabilities in AES emerge or under quantum computing threats. To address this, several researchers have successfully modified open-source file system pipelines to inject sovereign ciphers. A prominent example is the work by Bazanov et al. (2020) [7], who explored the possibility of using the Russian national cryptographic algorithm Kuznechik (GOST R 34.12-2015) within the ZFS architecture on Linux. Their

study demonstrated the feasibility of embedding a custom encryption module directly into the ZFS source code and evaluated its performance compared to native AES.

Although there have been individual advances in file system customization, a critical research gap remains unaddressed. First, existing academic efforts to integrate custom algorithms into ZFS (such as the Kuznechik integration) was limited to a basic Linux proof-of-concept and have not considered the enterprise storage architecture of dedicated NAS distributions like TrueNAS. Second, no prior study has attempted to integrate the newly standardized Vietnamese national cryptographic algorithm MKV with the sophisticated DMU layer of OpenZFS. This article addresses this problem by focusing on the architectural design required to seamlessly replace the native AES-GCM engine within the TrueNAS ZIO system with the MKV block cipher algorithm, thereby providing a complete, independent, and secure network-attached storage (NAS) solution.

IV. INTEGRATION OF THE MKV CIPHER ALGORITHM INTO THE TRUENAS SYSTEM

A. Data security in ZFS

ZFS provides data encryption features. Encrypted data includes file data and metadata, attributes, access permissions, file names, ACLs (names, permissions, attrs), directory listings, all Zvol data, FUID Mappings, master encryption keys, data stored in L2ARC and ZIL [7]. Unencrypted data includes dataset/snapshot names, dataset attributes, pool layout, ZFS structure, dedup tables and everything in RAM.

Depending on whether data duplication is checked before writing, ZFS encryption operates in two different modes: non-dedup (non-deduplication) and dedup (deduplication). In both cases, data blocks are encrypted with an encryption key derived from a master key. The master key is randomly generated using an initialization vector (IV), which acts as a type of salt to prevent encryption repetitions. The master key is never directly provided to the user. Instead, it is protected by a user-provided

wrapping key through a hash-based key derivation function. All wrapping keys are 256 bits in length, regardless of whether the selected key type is 128, 192, or 256 bit. Additionally, a MAC checksum is calculated for each block to ensure data integrity and protect against corruption [7].

- Non-dedup mode:

Because ZFS writes all data directly without checking for duplication, this mode offers advantages such as high performance, low RAM usage, excellent stability, and suitability for most use cases. However, its drawback is increased storage consumption when large amounts of duplicate data exist.

Figure 3 below illustrates the data encryption process in non-dedup mode.

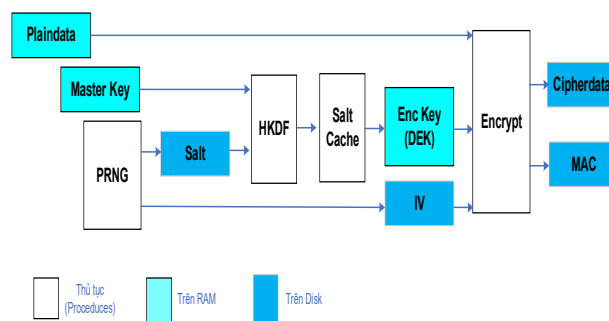


Figure 3. Data encryption in non-dedup mode

- Dedup mode:

In this mode, when writing data, ZFS calculates the hash of each data block and compares it with hashes of existing blocks. If any match is found, ZFS does not write new data but creates a reference to the corresponding block. The advantage of this mode is that it saves storage space in cases with considerable duplicate data, such as identical virtual machines, repeated backups, and duplicate operating system files. However, dedup consumes a large amount of RAM, typically around 1–5GB of RAM per TB of data processed. It also reduces performance due to the constant hash table lookups, and if RAM is insufficient, the system may become significantly slower.

Figure 4 below illustrates the data encryption process in dedup mode.

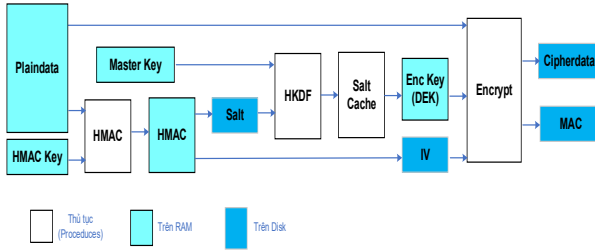


Figure 4. Data encryption in dedup mode

In deduplication mode, identical data blocks must produce the same ciphertext so that duplicates can be detected and removed. However, this conflicts with the encryption requirements, where the IV must be unique for each encryption to ensure security. To address this issue, ZFS generates the IV and salt deterministically by computing an HMAC of the plaintext using a secret key. From the resulting HMAC value, ZFS takes the first 64 bits as the salt and the next 96 bits as the IV. With this design, identical input data will produce the same IV and salt, resulting in identical ciphertext for effective deduplication, while security is maintained because the HMAC key remains secret.

B. Integrating the MKV block cipher algorithm

The MKV block cipher algorithm, defined in the TCVN 14263:2024 standard [5], is designed to resist quantum cryptanalysis [8, 9]. It supports two block sizes, 128-bit and 256-bit. For the 128-bit block size, three key lengths – 128-bit, 192-bit and 256-bit – are defined, with corresponding number of rounds of 7, 8, 9, respectively (denoted as MKV-128). For the 256-bit block size, three key lengths – 256-bit, 384-bit, 512-bit – are defined, with corresponding number of rounds of 7, 8, 9, respectively (denoted as MKV-256).

By default, ZFS uses the AES-256 block cipher in both dedup and non-dedup modes. To ensure long-term data security in the emergence of quantum computing, we propose replacing AES with the MKV-128 256-bit block cipher algorithm.

In ZFS, native encryption is handled within the DMU layer before the data blocks are passed to the SPA. In this section, we will present how the MKV cipher module replaces the default AES module within this layer.

The details of integrating the MKV-128 256-bit block cipher algorithm into the ZFS file system involve modifications to several modules in OpenZFS [10] as follows:

- In `include/sys/fs/zfs.h`:

```
/* supported encryption algorithms */
enum zio_encrypt {
    ZIO_CRYPT_INHERIT = 0,
    ZIO_CRYPT_ON,
    ZIO_CRYPT_OFF,
    ZIO_CRYPT_MKV_256_GCM,
    ZIO_CRYPT_MKV_256_CCM,
    ZIO_CRYPT_FUNCTIONS ;
};
```

- In `module/icp/algs/mkv128_impl.c`

```
void mkv_init_keysched(const uint8_t
*cipherKey, uint_t keyBits, void
*keysched)
int mkv_encrypt_block(const void *ks,
const uint8_t *pt, uint8_t *ct)
int mkv_decrypt_block(const void *ks,
const uint8_t *ct, uint8_t *pt)
void *mkv_alloc_keysched(size_t *size,
int kmflag)
```

- In `module/os/linux/zfs/zio_crypt.c`

```
const zio_crypt_info_t
zio_crypt_table[ZIO_CRYPT_FUNCTIONS] =
{
    {"", ZC_TYPE_NONE, 0,
    "inherit"},
    {"", ZC_TYPE_NONE, 0,
    "on"},
    {"", ZC_TYPE_NONE, 0,
    "off"},
    {SUN_CKM_MKV_GCM, ZC_TYPE_GCM,
    32, "mkv-256-gcm"},
    {SUN_CKM_MKV_CCM, ZC_TYPE_CCM,
    32, "mkv-256-ccm"}
};
```

- In `module/zcommon/zfs_prop.c`

```
static const zprop_index_t
crypto_table[] = {
    { "on", ZIO_CRYPT_ON
},
    { "off", ZIO_CRYPT_OFF
},
    { "mkv-256-gcm",
    ZIO_CRYPT_MKV_256_GCM },
    { "mkv-256-ccm",
    ZIO_CRYPT_MKV_256_CCM },
    { NULL }
};
```

After integrating the MKV algorithm into OpenZFS, the following command can be used to create encrypted datasets:

```
zpool create -o ashift=12 -o
encryption=mkv-256-gcm -o
keyformat=passphrase
-o keylocation=prompt -o xattr=sa
DATA raidz1 /dev/sdb /dev/sdc /dev/sdd
zfs create DATA/encdata1
zfs create DATA/encdata2
```

V. EXPERIMENTAL RESULTS

The ZFS encryption feature in TrueNAS [11] was evaluated in a test environment consisting of a TrueNAS server equipped with an NVMe hard drive. A storage pool containing three datasets was created to evaluate different storage configurations: (i) a standard dataset without encryption, (ii) an encrypted dataset using the AES-256-GCM algorithm provided by the native ZFS encryption mechanism, and (iii) an encrypted dataset using the proposed MKV-GCM-256 algorithm.

The fio benchmarking tool was used to evaluate sequential write performance on each dataset under identical test conditions. In addition to this, the encryption functionality was verified by attempting to access data stored in the encrypted datasets without the correct encryption key. The results confirmed that the encrypted data could not be accessed without the corresponding key, demonstrating the correct operation of both encryption mechanisms while enabling a performance comparison among the three storage configurations.

The commands used to measure write speed on the encrypted dataset with a 8K block size is shown below:

```
fio --name=seqwrite_8K --
filename=/mnt/test/atnv_ma/test --
ioengine=libaio --direct=1 --
rw=randwrite --bs=8K --iodepth=1 --
numjobs=16 --direct=1 --size=1G --
runtime=60 --time_based --
group_reporting
...
Run status group 0 (all jobs):
WRITE: bw=371MiB/s (390MB/s)
```

Table I summarizes the experimental write performance results of the three storage configurations in terms of Input/Output Operations Per Second (IOPS), bandwidth, and CPU utilization.

TABLE I. EXPERIMENTAL WRITE SPEEDS

Alg.	IOPS	Bandwidth		CPU	
	10 ³	MB/s	%	usr (%)	sys (%)
Non-encrypted	90.8	774	100	0.65	1.93
AES	74.3	608	78.55	0.62	1.73
MKV	49.9	390	50.39	0.44	1.32

The non-encrypted configuration achieved the highest write performance, reaching 90.8 KIOPS with a bandwidth of 774 MB/s. When AES encryption was applied, the performance decreased to 74.3 KIOPS and 608 MB/s, corresponding to reductions of approximately 18.2% in IOPS and 21.4% in bandwidth compared with the non-encrypted configuration. The MKV configuration showed a further reduction, achieving 49.9 KIOPS and 390 MB/s.

Regarding CPU utilization, all three configurations exhibited low resource consumption. The non-encrypted, AES, and MKV configurations consumed 0.65%, 0.62%, and 0.44% of user CPU time, respectively, while system CPU usage was 1.93%, 1.73%, and 1.32%. Among the three configurations, the MKV configuration recorded the lowest CPU utilization in both user and system CPU measurements.

The higher I/O bandwidth achieved by the AES configuration compared with the MKV configuration can be attributed to the hardware acceleration support available in modern CPUs [12]. Specifically, AES encryption benefits from the AES-NI (Advanced Encryption Standard New Instructions) instruction set, which provides hardware-assisted cryptographic operations. In contrast, the MKV encryption mechanism relies on software-based cryptographic operations, resulting in different processing characteristics compared with AES acceleration.

The availability of AES-NI support on a Linux system can be verified using the following command:

```
lsmod | grep aes
```

The output indicates the presence of the AES-NI kernel module:

```
aesni_intel 356352 0
crypto_simd 16384 1 aesni_intel
```

VI. CONCLUSION

Data encryption plays a crucial role in storage systems by protecting information from unauthorized access and ensuring data confidentiality and integrity. This article demonstrates the feasibility of integrating the block cipher into the ZFS component of the TrueNAS system, focusing on seamlessly replacing the native AES-GCM mechanism to build a complete, independent, and secure network-attached storage (NAS) system. Experimental results show that the proposed approach provides data protection while introducing additional performance overhead compared with conventional encryption. Although further optimization is required to improve storage performance, the results indicate that the integration of MKV is a promising approach for applications requiring enhanced data security with an acceptable performance trade-off.

REFERENCES

- [1] J. Bonwick, B. Moore, "The Zettabyte File System", *Sun Microsystems*, White Paper, 2007.
- [2] Y. Zhang, A. Rajimwale, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, "End-to-End Data Integrity for File Systems: A ZFS Case Study", *Proceedings of the 8th USENIX Conf. on File and Storage Technologies (FAST '10)*, pp. 29-42, 2010.
- [3] A. Kadav, A. Rajimwale, *Reliability Analysis of ZFS*, University of Wisconsin–Madison, 2007.
- [4] A. Li, "Zettabyte File System Autopsy: Digital Crime Scene Investigation for Zettabyte File System", 2009.
- [5] Bộ Khoa học và Công nghệ, *Tiêu chuẩn quốc gia TCVN 14263:2024 về Công nghệ thông tin - Kỹ thuật an toàn - Thuật toán mã khối MKV*, 2024.
- [6] Ban Cơ yếu Chính phủ, "Thuật toán mã khối MKV", Access time: 15/8/2025, <https://bcy.gov.vn/tieu-chuan-quy-chuan-ky-thuat-mat-ma/thuat-toan-ma-khoi-mkv-928832>.
- [7] V.V. Bazanov, N.S. Ivanov, D.A. Subbotin, "Research the Possibility of Using Russian

Cryptographic Algorithm in the ZFS File System", *2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, 2020.

- [8] B. C. Nguyen, V. L. Nguyen, T. A. Nguyen, S. N. Tran, D. L. Tran, D. L. Hoang, "MKV: a new block cipher for the post-quantum cryptography transition", *МАТЕМАТИЧЕСКИЕ ВОПРОСЫ КРИПТОГРАФИИ*, vol. 16, no. 2, pp. 113-138, 2025.
- [9] V. L. Nguyen, D. L. Hoang, Q. D. Le, "The quantum circuit construction for S-boxes without ancilla qubits: A more detail analysis", *Journal of Science and Technology on Information Security*, vol. 1, no. 21, pp. 75-88, 2024. DOI: 10.54654/isj.v1i21.1031.
- [10] GitHub, "OpenZFS Project", Access time: 15/8/2025, <https://github.com/openzfs/zfs>.
- [11] GitHub, "TrueNAS Project", Access time: 15/8/2025, <https://github.com/truenas>.
- [12] V. K. Tran, V. K. Phan, V. V. Vu, "Impact of Cipher-Core Selection on XTS-based Secure Network-Attached Storage Systems: A Comparison Study of MKV and AES", *Journal of Science and Technology on Information Security*, vol. 1, no. 27, pp. 20–34, 2026. DOI: 10.54654/isj.v1i27.4042.

ABOUT THE AUTHOR



Pham Van Luc

Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission

Email: pvluc@bcy.gov.vn

Education: He received the M.Sc. degree from Academy of

Cryptography Techniques in 2008, and the Ph.D. degree from Posts and Telecommunications Institute of Technology in 2022.

Recent research direction: His research interests include Analysis and Design of Security Protocols, and Applied Cryptography.

Tên tác giả: **Phạm Văn Lực**

Cơ quan công tác: Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ, Việt Nam

Email: pvluc@bcy.gov.vn

Quá trình đào tạo: Tốt nghiệp Thạc sĩ tại trường Học viện Kỹ thuật mật mã năm 2008 và Tiến sĩ tại trường Học viện Công nghệ Bưu chính Viễn thông năm 2022.

Hướng nghiên cứu hiện nay: Phân tích và thiết kế các giao thức bảo mật và mật mã ứng dụng.



Hoàng Thu Phương

Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission

Email: htphuong@bcy.gov.vn

Education: She received her B.S. degree from Shanghai University, China in 2012 and her M.S. degree from Academy of Cryptography Techniques, Vietnam in 2019.

Recent research direction: Her research interests include Applied Cryptography, and Software Security.

Tên tác giả: **Hoàng Thu Phương**

Cơ quan công tác: Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ, Việt Nam

Email: htphuong@bcy.gov.vn

Quá trình đào tạo: Tốt nghiệp Cử nhân tại trường Đại học Thương Hải, Trung Quốc năm 2012 và Thạc sĩ tại trường Học viện Kỹ thuật mật mã, Việt Nam năm 2019.

Hướng nghiên cứu hiện nay: Mật mã ứng dụng và bảo mật phần mềm.



Trần Huy Phương

Workplace: Cryptography Technical College, Vietnam

Email: phuonghuy6@gmail.com

Education: He graduated from Telecommunications University in 2011.

Recent research direction: His research interest is Cybersecurity.

Tên tác giả: **Trần Huy Phương**

Cơ quan công tác: Trường Cao đẳng Kỹ thuật Mật mã, Việt Nam

Email: phuonghuy6@gmail.com

Quá trình đào tạo: Tốt nghiệp trường Sĩ quan Thông tin năm 2011.

Hướng nghiên cứu hiện nay: An toàn thông tin trên mạng.

Hoàng Xuân Thành

Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission

Email: xuanthanhchv@gmail.com

Education: He graduated from A.F. Mozhaisky Military-Space

Academy in 2023.

Recent research direction: His research interests include Communication Security, Data Storage Security and Information Security.

Tên tác giả: **Hoàng Xuân Thành**

Cơ quan công tác: Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ, Việt Nam

Email: xuanthanhchv@gmail.com

Quá trình đào tạo: Tốt nghiệp Kỹ sư tại trường Học viện Quân sự Vũ trụ A.F. Mozhaisky năm 2023.

Hướng nghiên cứu hiện nay: Bảo mật truyền thông, bảo mật lưu trữ và an toàn thông tin



Thái Doãn Sang

Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission

Email: doansangk54@gmail.com

Education: He graduated with an Engineering degree from Military

Technical Academy in 2023.

Recent research direction: His research interests include Information Security, and Applied Cryptography.

Tên tác giả: **Thái Doãn Sang**

Cơ quan công tác: Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ, Việt Nam

Email: doansangk54@gmail.com

Quá trình đào tạo: Tốt nghiệp Kỹ sư tại Học viện Kỹ thuật quân sự năm 2023.

Hướng nghiên cứu hiện nay: Bảo mật và an toàn thông tin, mật mã ứng dụng.