IRIS AperTO

UNIVERSITÀ DEGLI STUDI DI TORINO

AperTO - Archivio Istituzionale Open Access dell'Università di Torino

**End-To-End Confidentiality with Sev-Snp Leveraging in-Memory Storage**

**This is the author's manuscript**

*Original Citation:*

*Availability:*

This version is available http://hdl.handle.net/2318/2093251 since 2025-09-05T12:30:51Z

*Publisher:*

IEEE

*Published version:*

DOI:10.1109/eurospw67616.2025.00054

*Terms of use:*

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)

05 September 2026

# End-to-End Confidentiality with SEV-SNP Leveraging In-Memory Storage

Lorenzo Brescia*†, Iacopo Colonnelli*, Valerio Schiavoni†, Pascal Felber†, Marco Aldinucci*

*University of Turin, Turin, Italy
{lorenzo.brescia, iacopo.colonnelli, marco.aldinucci}@unito.it

†University of Neuchâtel, Neuchâtel, Switzerland
{valerio.schiavoni, pascal.felber}@unine.ch

***Abstract*—Confidential computing ensures data in-use protection in untrusted cloud environments, yet securing data at-rest typically relies on Full Disk Encryption (FDE), which imposes significant performance overhead. This work proposes an alternative in-memory storage approach that eliminates FDE by leveraging SEV-SNP confidential virtual machines (CVMs). Our framework extends SNPGuard, an open-source platform for booting and attesting SEV-SNP VMs, to manage workload execution using temporary file systems (tmpfs), inherently secured by CVM memory encryption. By enabling seamless deployment of Docker based applications, our approach improves runtime and throughput by 20% on average, with peak gains of 45% in read-only database workloads. These findings establish in-memory storage as a secure and performant alternative to FDE for handling temporary intermediate data in storage intensive workflows, laying the foundation for future research in this direction.**

## 1. Introduction

Cloud computing has become a fundamental solution for outsourcing computational workloads, allowing users to access scalable infrastructure on demand. Leading providers such as Amazon, Microsoft, Google, and IBM offer flexible resource allocation, enabling cost efficiency, seamless scalability, and enhanced reliability. However, mitigating privacy concerns remains a significant challenge, particularly when handling sensitive user data [1]–[3]. These concerns are further amplified by stringent regulatory frameworks like the General Data Protection Regulation (GDPR) [4], the Health Insurance Portability and Accountability Act (HIPAA) [5], and the California Consumer Privacy Act (CCPA) [6], which impose strict requirements on data protection and compliance.

Confidential computing [7], [8] addresses these privacy concerns by leveraging specialized hardware to enforce security guarantees for data in-use. However, ensuring confidentiality during computation alone is insufficient; data must also be protected while in-transit across networks and when stored persistently at-rest. Achieving end-to-end data protection requires integrating complementary security mechanisms. Well-established techniques such as Transport Layer Security (TLS) [9] and Secure Shell (SSH) [10] secure data in-transit, while full-disk encryption (FDE) [11], [12] safeguards data at-rest. Technologies like Linux Unified Key Setup (LUKS) [13], [14] provide robust encryption mechanisms at the storage level. However, our microbenchmark results in §5.1 demonstrate that FDE can introduce substantial performance overhead, making it a less efficient solution for storage intensive workloads.

This work explores an alternative approach that delegates data at-rest protection to the same mechanisms used for safeguarding data in-use. Specifically, we present a framework based on prior research on AMD confidential computing technologies, called SNPGuard [15]. Our framework [16] enables the seamless execution of end-to-end confidential workloads within a Docker [17] container, eliminating the need for traditional disk encryption. Instead, it leverages temporary file systems (tmpfs) [18], which inherently benefit from AMD's memory encryption mechanisms. By relying on encrypted memory rather than persistent storage, this approach mitigates the performance penalties associated with FDE while maintaining strong security guarantees. Our results demonstrate that our framework improves the runtime and throughput of storage intensive workloads by 20% on average, with peak gains of 45% in read-intensive database workloads. Even though FDE is a straightforward solution for long-term storage of workload results, we demonstrate that leveraging in-memory file systems for temporary intermediate data is an effective technique to accelerate storage intensive workloads.

The remainder of this paper is organized as follows. §2 provides an overview of existing confidential computing technologies and a brief description of AMD Secure Encrypted Virtualization (SEV), including its architecture, attestation flow, and how to obtain end-to-end workload protection. §3 reviews prior studies on I/O protection in confidential computing and describes the SNPGuard setup for attested confidential VMs. §4 details our framework, which extends SNPGuard to enable seamless execution of end-to-end protected Docker workloads. §5 presents microbenchmark results assessing the overhead of FDE, along with two macrobenchmarks, evaluating compression/decompression performance and on database workloads, respectively. §6 discusses the limitations of our approach and proposes potential future research directions. Finally, §7 concludes by summarizing our key findings.

## 2. Background

The primary objective of confidential computing is to protect data in-use by establishing Trusted Execution Environments (TEEs). A TEE provides code integrity, data confidentiality, and remote attestation procedures to verify its authenticity [19]. TEEs can be categorized into two primary isolation models. The first is process-based isolation, where security is enforced at the process level. Notable implementations include Intel Software Guard Extensions (SGX) [20] and Arm TrustZone [21], [22]. The second model is VM-based isolation, which extends protection to the entire virtual machine. Prominent examples of this approach include Intel Trust Domain Extensions (TDX) [23], [24], AMD SEV [25]–[27], Arm Confidential Compute Architecture (CCA) [28], RISC-V Confidential Virtual Environments (CoVE) [29], IBM OpenPOWER ISA [30], IBM Secure Execution for Linux [31], and NVIDIA's confidential VMs for its Hopper GPUs [32].

This work focuses on AMD SEV. The following sections provide a technical overview of its architecture (Section 2.1) and describe how to achieve end-to-end confidentiality through attestation mechanisms (Section 2.2).

### 2.1. AMD SEV

Secure Memory Encryption (SME) [25] is an AMD security feature designed to enhance memory confidentiality by encrypting memory pages using a hardware-integrated AES encryption engine and a dedicated C-bit marker. The AES encryption key is randomly generated and securely managed by the AMD Secure Processor (ASP), ensuring that encryption is applied transparently to memory management operations without necessitating modifications to the operating system or application software.

Building upon SME, Secure Encrypted Virtualization (SEV) [25] integrates memory encryption into AMD Virtualization (AMD-V) [33] to establish isolated execution environments for guest virtual machines. SEV assigns a unique encryption key to each VM using Address Space Identifier (ASID) tags, ensuring that encrypted data remains inaccessible to unauthorized entities, including the hypervisor and host operating system.

SEV memory pages are categorized as either private or shared, utilizing the SME C-bit mechanism to differentiate between them. Private pages are encrypted with VM-specific keys, ensuring strict isolation and preventing unauthorized access. Instead, shared pages remain unprotected, requiring additional security mechanisms to safeguard sensitive data. To mitigate potential vulnerabilities, SEV enforces the confidentiality of critical memory regions, such as instruction pages and page tables, by designating them as private. Conversely, Direct Memory Access (DMA) operations are confined to shared memory, preventing unauthorized entities from accessing confidential data. SEV provides transparent protection for data in-use, but it does not inherently secure data during transmission or storage. To achieve comprehensive, end-to-end confidentiality, complementary security solutions must be implemented, ensuring that data remain protected throughout their entire lifecycle.

SEV Encrypted State (SEV-ES) [26] enhances SEV by encrypting VM register states, preventing unauthorized access to execution data. When a VM is suspended, its registers — typically stored in hypervisor accessible memory — are encrypted using the VM's memory encryption key. To ensure integrity, the hardware generates a verification value during save and restore operations, which is stored in a protected memory region and validated upon resumption, preventing tampering.

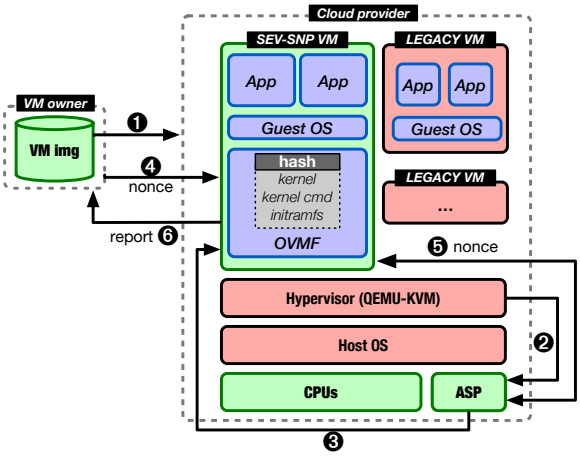


Figure 1. SEV-SNP attestation flow: trusted components are highlighted in green, while untrusted components are shown in red.

SEV Secure Nested Paging (SEV-SNP) [27] is the latest advancement in SEV technology, enhancing memory integrity protections by ensuring that private memory always reflects the VM's most recent writes. It achieves this goal through security mechanisms such as the Reverse Map Table (RMP) for tracking memory mappings, page validation for enforcing integrity constraints, and page state tracking for monitoring access permissions.

### 2.2. End-to-end data protection with SEV

SEV-SNP guarantees data confidentiality and integrity during VM execution. However, the VM launch process takes place in an untrusted environment, such as a cloud provider's infrastructure. Remote attestation [27] provides cryptographic proof of both the authenticity of the underlying hardware and the integrity of the software running within the confidential VM. Nevertheless, while SEV-SNP enforces memory protection, its security guarantees do not inherently extend to end-to-end data protection, delegating the protection of data in-transit and at-rest to complementary mechanisms.

**Attestation flow**. The attestation procedure involves a relying party, typically the VM owner, who wants to verify the authenticity of a VM launched within an untrusted cloud infrastructure as shown in Fig. 1. The VM owner provides the cloud with a VM image for deployment (Fig. 1-❶). During the launch process, the hypervisor requests the ASP (Fig. 1-❷) to initialize the guest VM's memory pages and generate a cryptographic measurement of this initial state (Fig. 1-❸). By default, this measurement primarily reflects the virtual firmware and a few initial CPU register values. However, the process can be enhanced by leveraging modified versions of QEMU (as the hypervisor) [34] and OVMF (as the virtual firmware) [35]. These extensions enable additional

space within the OVMF executable to store hashes of critical components, including the kernel, initramfs, and kernel command line. During VM initialization, QEMU computes and injects these hashes into the OVMF, which then verifies them at runtime. Consequently, attestation extends beyond the OVMF firmware itself, covering the loaded kernel, boot parameters and initramfs. In order to obtain the attestation report, the VM owner sends a nonce to the guest VM (Fig. 1-❹), which in turn requests it from the ASP through a secure communication channel (Fig. 1-❺). The generated report includes a measurement of the VM's initial state, platform details, and Trusted Computing Base (TCB) status. Additionally, it may contain arbitrary user-defined data, which can be used for key exchange algorithms or nonces to prevent replay attacks. Once the guest VM receives the report, it forwards it to the VM owner (Fig. 1-❻), who verifies its authenticity by validating it against AMD's certificate chain. Upon successful attestation, the VM owner can, if necessary, securely provision sensitive data, such as an encryption key needed to unlock the virtual disk. Previous research has extensively examined the security guarantees of SEV attestation [36] and proposed methodologies to streamline attestation procedures [15], [37], [38].

**Protection in-transit and at-rest**. Since shared memory pages of a confidential VM fall outside SEV-SNP's security guarantees, achieving end-to-end confidentiality requires additional safeguards. Protecting data in-transit between the VM owner and the VM itself necessitates secure communication channels, such as TLS or SSH. Furthermore, securing data at-rest — specifically during read and write operations on virtual storage — remains a critical challenge. A common approach is FDE using LUKS, but this introduces significant performance overhead. In contrast, we propose an alternative strategy that mitigates this overhead by leveraging tmpfs-mounted directories within the VM. Since tmpfs operates entirely in volatile memory, which is inherently protected by SEV-SNP, data never resides on a persistent storage device, significantly reducing I/O overhead. However, this approach also introduces challenges related to data persistence, as all stored data is lost upon VM shutdown. We use a straightforward orchestration method for managing confidential VMs under volatile constraints, leaving the development of more sophisticated orchestration strategies for future research.

## 3. Related work

M. Yan and K. Gopalan [39] highlight that disk I/O suffers from significant performance degradation when using SEV, with penalties reaching up to 56%. Similarly, L. Qiu et al. [40] present microbenchmark results indicating that a non-confidential VM can perform disk operations at twice the speed of a SEV VM. Furthermore, M. Misono et al. [41] observe that when CPU utilization is high, the I/O overhead stems from the internal implementation of default I/O software stacks, which rely on bounce buffers. These buffers are essential for SEV to handle I/O operations, as they require an additional copy of the data from private VM pages to unprotected shared pages. Given these challenges, achieving efficient I/O performance in confidential computing environments is crucial.

However, as H. Lefeuvr et al. [42] noted, the security and efficiency of confidential I/O remain largely unexplored. The authors emphasize that designing secure and efficient I/O interfaces is a complex task. One approach to addressing I/O bottlenecks is optimizing the software stack that manages I/O operations. Another promising strategy is extending the trust boundary to include devices capable of operating directly within the confidential VM's private memory. AMD SEV-TIO [43] and Intel TDX Connect [44] aim to improve I/O performance by enabling direct interaction between hardware components and confidential memory pages.

To evaluate our methodology, which relies on volatile computation within tmpfs mounted directories inside an SEV-SNP VM, we utilize SNPGuard [15], an effective open-source solution for deploying guest VMs with or without LUKS FDE. It allows direct Linux booting with a specified SEV-SNP-enabled kernel and an arbitrary initramfs. With SNPGuard two kind of workflows are available. In the confidentiality and integrity workflow, the virtual disk is ciphered and disk encryption keys are provisioned during the initramfs phase via remote attestation using a Diffie-Hellman key exchange [45]. In the integrity-only workflow, the VM operates with a non-confidential, read-only disk, where its integrity is verified during the initramfs phase at launch. Since directories such as `/home`, `/etc`, and `/var` require write access, they are mounted as tmpfs, ensuring their protection through SEV-SNP's memory encryption. In this case, secrets can be securely provisioned after attestation, but workload setup and execution remain the responsibility of the VM owner. Our framework extends the integrity-only workflow by enabling seamless deployment of arbitrary Docker containers through a simple JSON configuration. This approach eliminates the performance overhead of FDE, replacing it with a more efficient in-memory storage solution, significantly improving execution speed while maintaining strong security guarantees.

## 4. Methodology

**Management of volatile computations**. After booting with SNPGuard's integrity-only workflow, the VM remains unattested. However, during the initramfs phase, the VM stores the attestation report in `/etc`, which is secured by being mounted as tmpfs. The first necessary step is to retrieve and validate this report to ensure that the VM has booted correctly. Once attestation is verified, the VM owner can begin using the confidential VM. In this setup, the responsibility for configuring and executing workloads falls entirely on the VM owner, who must also address challenges related to operating in a volatile memory environment. Since only `/home`, `/etc`, `/var` and `/tmp` are mounted as tmpfs, the rest of the disk remains non-confidential and read-only. Consequently, workloads requiring write access outside these directories cannot run. Another limitation arises from the static allocation of tmpfs sizes in the integrity workflow. These sizes are hardcoded during the initramfs compilation, requiring recompilation for any modifications, which significantly reduces flexibility for the VM owner.

Our framework addresses these limitations by allowing tmpfs sizes to be specified at launch without requiring

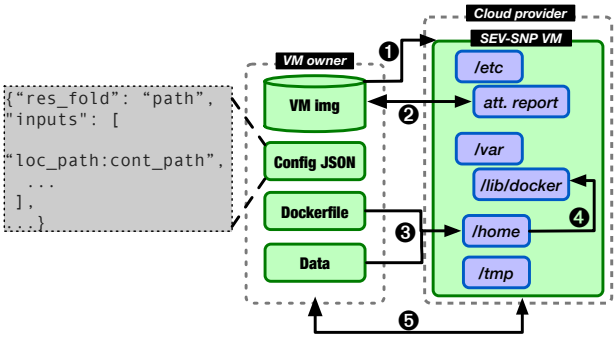


Figure 2. Execution flow for achieving end-to-end confidentiality in a Docker workload. Blue indicates folders mounted as tmpfs, while all other directories are mounted in a standard read-only configuration.

initramfs recompilation and by leveraging Docker for workload portability. Docker operates, using an overlay filesystem [46], primarily within specific system directories such as `/var/lib/docker`. A Docker image is built from an arbitrary Dockerfile, configuring the necessary dependencies within this location. When executed, the container utilizes the same overlay filesystem, ensuring that all runtime writes occur within predefined directories mounted as tmpfs. This guarantees that all modifications remain within SEV-SNP-protected memory, enabling the execution of applications in a volatile memory environment while maintaining portability.

In cloud environments, workloads often process sensitive data that require end-to-end protection throughout their lifecycle. Ensuring confidentiality demands secure data transmission between the VM owner and the cloud provider after attestation. To achieve this, our framework employs SSH for securing data in-transit and relies on tmpfs to protect data at-rest. Once the Docker image is built within the VM and the necessary data is securely transferred, the container must be executed. A key challenge lies in correctly mapping data volumes [47] between the VM and the container, ensuring proper access to required files. This is managed by provisioning a JSON configuration file, which specifies essential options, including the correct directory mappings and the VM owner's designated folder for securely collecting results before the VM is shut down.

**End-to-end confidentiality execution flow**. The VM owner possesses VM image, confidential data, a Dockerfile defining the image that will process the data, and a configuration file specifying critical settings such as input mappings between local data and the container, as well as the location designated to collect the workload results. Figure 2 illustrates the process for achieving end-to-end confidential workload execution without relying on costly FDE techniques. Our framework automates the deployment of the VM using SNPGuard's integrity workflow (Fig. 2-❶). Once the VM is booted, the attestation report is retrieved and validated (Fig. 2-❷). If verification is successful, the next step is securely transferring the Dockerfile and input data specified in the JSON configuration file into the VM (Fig. 2-❸). The Docker image is then built inside the VM (Fig. 2-❹), followed by the execution of the container, ensuring that all necessary volumes are correctly mounted according to the predefined directories within the VM. Upon completion of the computation, if the workload produces output data, the VM owner initiates a secure retrieval process to ensure that all specified results are safely transferred back (Fig. 2-❺). Once the data transfer is complete, the VM can be shut down without any risk of data loss.

## 5. Results

**Testbed**. All experiments were conducted on a machine equipped with an AMD EPYC 9124 (4th Gen. Genoa) processor, featuring 16 physical cores and simultaneous multithreading (SMT) for a total of 32 threads. The system includes 66 GiB of RAM and 512 GB of SSD storage. The host operating system is Ubuntu 22.04.5, running kernel version 6.9.0-rc7-snp-host-05b10142ac6a. We deployed three different VMs, each with identical virtual hardware specifications, including 32 vCPUs, 32 GiB of RAM, and a 70 GB scsi-hd virtual disk. All VMs run Ubuntu 22.04.5 with kernel 6.9.0-snp-guest-a38297e3fb01 and identical software components, such as Docker. The only difference among them is the filesystem configuration (with ext4 as the baseline), allowing us to compare different data at-rest protection techniques:

| | |
|---|---|
| std | A standard SEV-SNP VM. |
| FDE | A SEV-SNP VM utilizing LUKS FDE. |
| tmpfs | A SEV-SNP VM that leverages tmpfs mounted directories to secure data at-rest. |

To evaluate LUKS FDE performance, in Section 5.1 we perform various Flexible I/O (`fio`) [48] microbenchmarks on the three SEV-SNP VMs. Additionally, we assess the efficiency of our volatile end-to-end confidentiality framework using two storage intensive macrobenchmarks: compression and decompression with `tar` and `gzip` (Section 5.2) and multiple YCSB [49] database workloads on RocksDB [50] (Section 5.3). To ensure a fair comparison between VM configurations, the macrobenchmark workloads are executed within the same Docker container, built from an identical Docker image.

### 5.1. `fio`

Figure 3 illustrates the `fio` microbenchmark results using four job threads, a 4 kB block size, a 2 GB test file, and direct I/O with the `libaio` engine to bypass the OS page cache for accurate raw storage performance measurement. In the read benchmark (Figure 3a), the tmpfs solution achieves a significant performance boost, with bandwidth, throughput, and latency improvements of approximately 1200% compared to FDE and 540% compared to a standard virtual disk. While this drastic performance difference is expected, given that tmpfs operates entirely in memory — making it inherently faster than traditional storage — it is also noteworthy that FDE introduces a substantial overhead compared to an unencrypted virtual disk. This suggests that for real-world storage intensive workloads, the performance penalty of FDE could be pronounced. Consequently, leveraging tmpfs for volatile computation within a confidential computing environment, where memory is already protected, becomes an even more compelling approach. The write benchmark

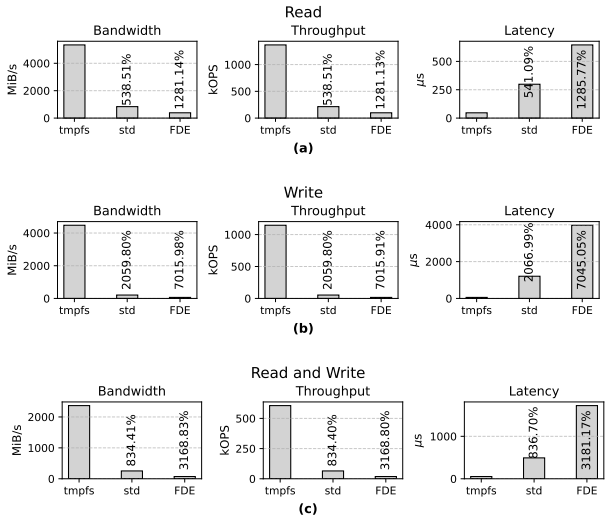


Figure 3. `fio` benchmarks, 4 threads, 4k blocksize, 2G test file size, 16 IOdepths, `libaio` engine and direct storage access. The bars represent bandwidth, throughput and latencies values for different SEV-SNP VM configurations: std without disk encryption, FDE using LUKS disk encryption and tmpfs mounting solution. The text within bars indicates the performance gaining obtained using tmpfs. **(a)** Read, **(b)** Write and **(c)** Read and Write.

(Figure 3b) shows an even greater disparity, with tmpfs achieving performance gains of approximately 7000% over FDE and 2000% over an unencrypted virtual disk. Mixed read/write workloads (Figure 3c) exhibit performance improvements in between these extremes, with tmpfs outperforming FDE by around 3100% and a standard disk configuration by about 830%.

We tested numerous `fio` parameter combinations across the three VM configurations. While we do not report all results here due to their consistency with the discussed trends, additional observations emerged. Increasing the number of jobs (tested with 1, 4, and 8 jobs) further amplifies the performance gains of the tmpfs solution. Conversely, increasing the block size (tested with 4 kB, 16 kB, 64 kB, and 1 MB) results in a decrease in tmpfs performance gains relative to the other configurations.

## 5.2. Compression and decompression

Compression and decompression tasks are inherently storage intensive when processing large files. We selected approximately 10 GB from an open dataset containing video clips of human actions [51]. We then executed the `tar` command to archive and compress/decompress the selected data. Figure 4 presents the average runtime results of three executions for compression and decompression across all SEV-SNP VM configurations. The deviation from the mean is not reported, as it is not significant. The compression task shows minimal benefit from our volatile computation approach, yielding only a 6.39% performance improvement compared to FDE. This limited gain is likely because compression relies heavily on CPU-bound operations rather than disk I/O, reducing the impact of storage performance on overall execution time.

In contrast, decompression exhibits a more significant performance improvement, with the tmpfs based approach achieving approximately 20% faster execution compared

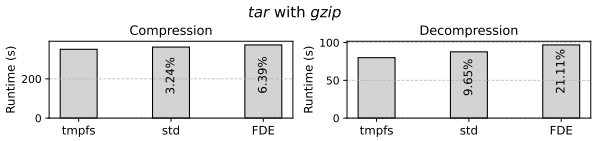


Figure 4. `tar` with `gzip` workloads. The bars represent the average runtime of three execution across different configurations: std without disk encryption, FDE using LUKS disk encryption and tmpfs mounting solution with our execution framework. The text within the bars indicates the performance gain achieved with tmpfs.

to FDE. This difference can be attributed to the higher dependency of decompression on disk read performance. Since tmpfs operates in memory, it eliminates the overhead of reading from an encrypted disk, leading to a noticeable reduction in runtime.

## 5.3. RocksDB

Database management workloads can significantly benefit from volatile execution. To investigate this, we ran various YCSB workloads on a RocksDB database across three SEV-SNP VM configurations. Figure 5 shows the average runtime, throughput, and operation latencies of three execution for six different workloads under each configuration. As for compression/decompression tasks, the deviation is not reported, as it was not significant.

In the read-update workload (Figure 5a), where read and update operations are evenly mixed, our tmpfs based solution improves runtime and throughput by approximately 20% compared to FDE. A similar improvement is observed in the workload where an update immediately follows a read (Figure 5d). In this case, the runtime benefit is around 18%. In contrast, in a workload with 95% reads and 5% updates (Figure 5b), the tmpfs solution achieves only about a 12% gain in runtime and throughput. Also the workload with 95% reads and 5% inserts (Figure 5c) shows a similar, modest improvement of roughly 10%. The most significant benefits are observed in the mixed update-insert workload (Figure 5e), which yields a runtime improvement of about 25%, and in the read-only workload (Figure 5f), where tmpfs delivers a throughput gain of up to 45%. Overall, the tmpfs solution reduces read latencies by between 10% and 213% relative to FDE. Update latencies improve by 13–30%, and insert latencies consistently show a 30% gain across all workloads.

These results can be explained by considering the nature of disk encryption overhead. In read-only workloads, FDE imposes significant decryption overhead for each access, as demonstrated by the highest latencies, making tmpfs — which operates entirely in memory — far superior. However, introducing even a small fraction of update operations brings additional factors into play, such as cache pressure and memory consistency mechanisms. While modern CPUs employ store buffers and write-back caching to optimize writes, updates can still lead to increased cache contention, eviction, or coherence traffic. This process reduces the relative advantage of tmpfs over FDE, as both systems must frequently access memory, partially mitigating FDE's decryption overhead. Additionally, the FDE system can optimize low-frequency writes through caching and batching, further narrowing

the performance gap. Thus, while read-only workloads showcase the maximum benefit of tmpfs, mixed workloads reveal a more nuanced performance gain for volatile tmpfs based computation.

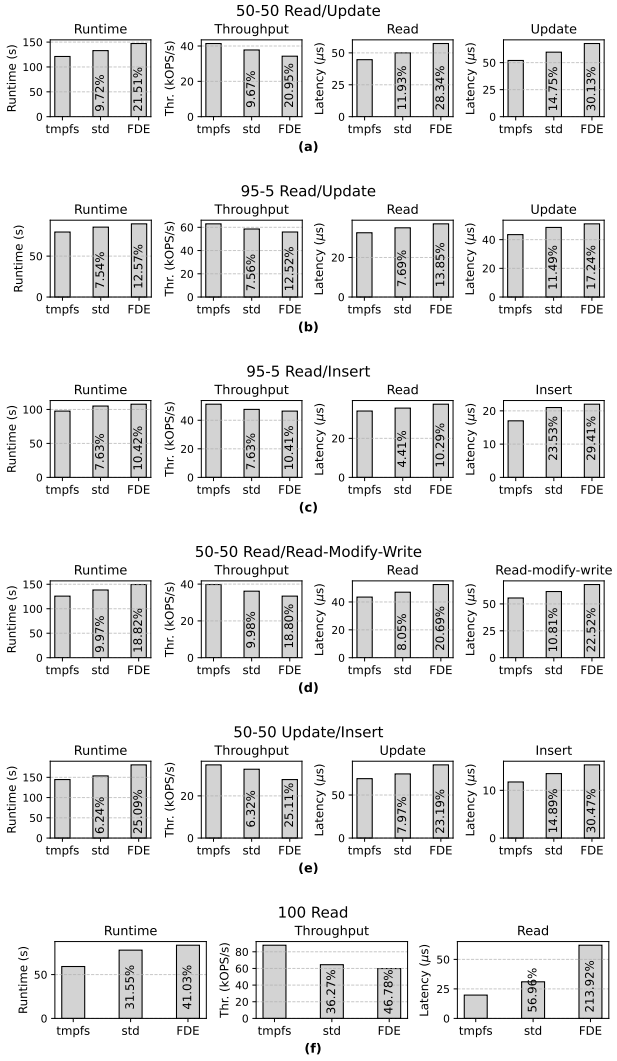


Figure 5. RocksDB YCSB workloads. The bars represent the average runtime, throughput and latencies of three execution across different configurations: std without disk encryption, FDE using LUKS disk encryption and tmpfs mounting solution with our execution framework. The text within the bars indicates the performance gain achieved with tmpfs. **(a)** 50% Read, 50% Update, **(b)** 95% Read, 5% Update, **(c)** 95% Read, 5% Insert, **(d)** 50% Read, 50% Read-Modify-Write, **(e)** 50% Update, 50% Insert, and **(f)** 100% Read.

## 6. Limitations and future work

**Limitations.** The proposed approach, while offering performance advantages, has limitations due to the nature of volatile execution. Memory rental costs are higher than disk storage, posing a cost challenge for large workloads. Additionally, the volatility of tmpfs impacts reliability, as data loss occurs when the VM shuts down. In case of failure, the lack of persistent checkpoints means execution restarts from scratch. Although result retrieval is supported, fault tolerance is not inherently provided. Estimating memory requirements is another challenge. While tmpfs size can be set at launch, determining the optimal configuration remains the VM owner's responsibility. Input data size and Docker image size can be estimated beforehand using the `docker images` command, but output size is unpredictable. However, since tmpfs allocates space as needed, resource estimation is feasible. A critical limitation arises when RAM is exhausted, as storage and VM execution compete for memory.

**Future research directions.** Our findings establish a foundation for future research by presenting a lightweight framework for managing Docker based computations, demonstrating the performance benefits of volatile execution over FDE while maintaining comparable security. Tested on AMD SEV-SNP machines, the model is adaptable to other architectures, such as Intel TDX, as it employs general CVM principles. Beyond hardware, the methodology has broader applications in distributed confidential computing, such as database sharding across locations with mixed confidentiality requirements. Applying volatile execution to high-performance, read-heavy workloads while ensuring robust security demonstrates its potential for cloud computing and secure data processing. Moreover, replacing naive tmpfs mounts with ad-hoc user-space file systems [52], [53] and I/O libraries [54] could reduce I/O overhead in confidential scientific workflows. Future research will also focus on a more precise comparative analysis of overhead between volatile execution and FDE, considering CPU utilization, Docker build time, CVM boot time, and data retrieval costs to assess performance and scalability.

## 7. Conclusion

Our findings from the `fio` microbenchmarks indicate that LUKS FDE imposes significant latency on storage access, which can negatively impact storage intensive applications deployed in the cloud. This suggests that when data protection is required throughout its entire lifecycle, FDE introduces substantial performance overhead. Our macrobenchmark results confirm this concern, demonstrating that our framework for volatile Docker execution with end-to-end data protection achieves an average performance improvement of 20% compared to FDE solution, with a peak gain of 45% for read-only database workloads. Read-only computations are particularly well-suited for volatile execution, as they can be performed without requiring result retrieval before the VM is terminated. Our framework provides end-to-end security by protecting data in-use and at-rest with SEV-SNP technology while securing data in-transit through SSH channels. The confidential VM operates with a read-only virtual disk that is integrity verified, while all sensitive data is securely transmitted via SSH to a tmpfs-mounted directory inside the VM, inherently protected by SEV-SNP.

## Acknowledgements

This work was supported by the Spoke 1 "FutureHPC & BigData" of ICSC - Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing, funded by the European Union - NextGenerationEU. This research was also supported by the Swiss-European Mobility Programme (SEMP).

# References


[1] F. K. Parast, C. Sindhav, S. Nikam, H. I. Yekta, K. B. Kent, and S. Hakak, "Cloud computing security: A survey of service-based models," *Comput. Secur.*, vol. 114, p. 102580, 2022. [Online]. Available: https://doi.org/10.1016/j.cose.2021.102580

[2] K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernández, "An analysis of security issues for cloud computing," *J. Internet Serv. Appl.*, vol. 4, no. 1, pp. 5:1–5:13, 2013. [Online]. Available: https://doi.org/10.1186/1869-0238-4-5

[3] L. Coppolino, S. D'Antonio, G. Mazzeo, and L. Romano, "Cloud security: Emerging threats and current solutions," *Comput. Electr. Eng.*, vol. 59, pp. 126–140, 2017. [Online]. Available: https://doi.org/10.1016/j.compeleceng.2016.03.004

[4] E. Parliament and C. of the European Union, "General data protection regulation (gdpr)," 2016, Last accessed: 2025-02. [Online]. Available: https://gdpr-info.eu

[5] U.S. Congress, "Health insurance portability and accountability act (hipaa)," 1996, Last accessed: 2025-02. [Online]. Available: https://www.govinfo.gov/content/pkg/PLAW-104publ191/pdf/PLAW-104publ191.pdf

[6] C. S. Legislature, "California consumer privacy act of 2018," 2018, Last accessed: 2025-02. [Online]. Available: https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5

[7] C. C. Consortium, "A technical analysis of confidential computing," 2022, Last accessed: 2025-02. [Online]. Available: https://confidentialcomputing.io/wp-content/uploads/sites/10/2023/03/CCC-A-Technical-Analysis-of-Confidential-Computing-v1.3_unlocked.pdf

[8] M. U. Sardar and C. Fetzer, "Confidential computing and related technologies: a critical review," *Cybersecur.*, vol. 6, no. 1, p. 10, 2023. [Online]. Available: https://doi.org/10.1186/s42400-023-00144-1

[9] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," *RFC*, vol. 8446, pp. 1–160, 2018. [Online]. Available: https://doi.org/10.17487/RFC8446

[10] T. Ylönen and C. Lonvick, "The secure shell (SSH) transport layer protocol," *RFC*, vol. 4253, pp. 1–32, 2006. [Online]. Available: https://doi.org/10.17487/RFC4253

[11] L. Khati, N. Mouha, and D. Vergnaud, "Full disk encryption: Bridging theory and practice," in *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, ser. Lecture Notes in Computer Science, H. Handschuh, Ed., vol. 10159. Springer, 2017, pp. 241–257. [Online]. Available: https://doi.org/10.1007/978-3-319-52153-4_14

[12] M. Broz, M. Patocka, and V. Matyás, "Practical cryptographic data integrity protection with full disk encryption," in *ICT Systems Security and Privacy Protection - 33rd IFIP TC 11 International Conference, SEC 2018, Held at the 24th IFIP World Computer Congress, WCC 2018, Poznan, Poland, September 18-20, 2018, Proceedings*, ser. IFIP Advances in Information and Communication Technology, L. J. Janczewski and M. Kutylowski, Eds., vol. 529. Springer, 2018, pp. 79–93. [Online]. Available: https://doi.org/10.1007/978-3-319-99828-2_6

[13] Linux Kernel Organization, "The Linux Kernel Archives," 2025. [Online]. Available: https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/dm-crypt.html

[14] C. Fruhwirth, "LUKS1 On-Disk Format Specification Version 1.2.3," 2018, Last accessed: 2025-02. [Online]. Available: https://www.kernel.org/pub/linux/utils/cryptsetup/LUKS_docs/on-disk-format.pdf

[15] L. Wilke and G. Scopelliti, " SNPGuard: Remote Attestation of SEV-SNP VMs Using Open Source Tools ," in *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE Computer Society, 2024, pp. 193–198.

[16] Anonymous, "End-to-End Confidentiality with SEV-SNP Leveraging In-Memory Storage," 2025, Last accessed: 2025-02. [Online]. Available: https://zenodo.org/records/14899836

[17] Docker Inc., "Docker," 2013, Last accessed: 2025-02. [Online]. Available: https://www.docker.com

[18] Linux Kernel Organization, "Tmpfs," Last accessed: 2025-02. [Online]. Available: https://www.kernel.org/doc/html/latest/filesystems/tmpfs.html

[19] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20-22, 2015, Volume 1*. IEEE, 2015, pp. 57–64. [Online]. Available: https://doi.org/10.1109/Trustcom.2015.357

[20] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptol. ePrint Arch.*, p. 86, 2016. [Online]. Available: http://eprint.iacr.org/2016/086

[21] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," *ACM Comput. Surv.*, vol. 51, no. 6, 2019.

[22] T. Alves and D. Felton, "Trustzone: Integrated hardware and software security," 01 2004.

[23] Intel, "Intel® Trust Domain Extensions (Intel® TDX)," 2023, Last accessed: 2025-01. [Online]. Available: https://cdrdv2.intel.com/v1/dl/getContent/690419

[24] ——, "Intel CPU Architectural Extensions Specification," 2021, Last accessed: 2025-01. [Online]. Available: https://cdrdv2.intel.com/v1/dl/getContent/733582

[25] AMD, "AMD Memory Encription," 2016, Last accessed: 2025-01. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/memory-encryption-white-paper.pdf

[26] ——, "Protecting VM Register State with SEV-ES," 2017, Last accessed: 2025-01. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/Protecting-VM-Register-State-with-SEV-ES.pdf

[27] ——, "AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More," 2020, Last accessed: 2025-01. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf

[28] ARM, "Arm confidential compute architecture," 2023, Last accessed: 2025-02. [Online]. Available: https://developer.arm.com/documentation/den0125/0300

[29] R. Sahita, V. Shanbhogue, A. Bresticker, A. Khare, A. Patra, S. Ortiz, D. Reid, and R. Kanwal, "Cove: Towards confidential computing on risc-v platforms," in *Proceedings of the 20th ACM International Conference on Computing Frontiers*. ACM, 2023.

[30] G. D. H. Hunt, R. Pai, M. V. Le, H. Jamjoom, S. Bhattiprolu, R. Boivie, L. Dufour, B. Frey, M. Kapur, K. A. Goldman, R. Grimm, J. Janakirman, J. M. Ludden, P. Mackerras, C. May, E. R. Palmer, B. B. Rao, L. Roy, W. A. Starke, J. Stuecheli, E. Valdez, and W. Voigt, "Confidential computing for openpower," in *Proceedings of the Sixteenth European Conference on Computer Systems*. ACM, 2021.

[31] C. Bornträger, J. D. Bradbury, R. Bündgen, F. Busaba, L. C. Heller, and V. Mihajlovski, "Secure your cloud workloads with ibm secure execution for linux on ibm z15 and linuxone iii," *IBM Journal of Research and Development*, vol. 64, no. 5/6, pp. 2:1–2:11, 2020.

[32] NVIDIA, "NVIDIA Confidential Computing," 2021, Last accessed: 2025-02. [Online]. Available: https://www.nvidia.com/en-us/data-center/solutions/confidential-computing/

[33] AMD, "Amd64 virtualization technology," 2005, Last accessed: 2025-02. [Online]. Available: https://www.0x04.net/doc/amd/33047.pdf

[34] AMDESE, "QEMU fork for AMD SEV-SNP," Last accessed: 2025-02. [Online]. Available: https://github.com/AMDESE/qemu

[35] ——, "OVMF fork for AMD SEV-SNP," Last accessed: 2025-02. [Online]. Available: https://github.com/AMDESE/ovmf

[36] R. Buhren, C. Werling, and J.-P. Seifert, "Insecure until proven updated: Analyzing amd sev's remote attestation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. ACM, 2019, p. 1087–1099.

[37] A. Galanou, K. Bindlish, L. Preibsch, Y. Pignolet, C. Fetzer, and R. Kapitza, "Trustworthy confidential virtual machines for the masses," in *Proceedings of the 24th International Middleware Conference, Middleware 2023, Bologna, Italy, December 11-15, 2023*. ACM, 2023, pp. 316–328. [Online]. Available: https://doi.org/10.1145/3590140.3629124

[38] D. Pontes, F. Silva, E. D. L. Falcão, and A. Brito, "Attesting AMD SEV-SNP virtual machines with SPIRE," in *12th Latin-American Symposium on Dependable and Secure Computing, LADC 2023, La Paz, Bolivia, October 16-18, 2023*. ACM, 2023, pp. 1–10.

[45] E. Rescorla, "Diffie-hellman key agreement method," *RFC*, vol. 2631, pp. 1–13, 1999. [Online]. Available: https://doi.org/10.17487/RFC2631

[46] Linux Kernel Organization, "Overlay filesystems," Last accessed: 2025-02. [Online]. Available: https://docs.kernel.org/filesystems/overlayfs.html

[47] Docker Inc., "Docker volumes," Last accessed: 2025-02. [Online]. Available: https://docs.docker.com/engine/storage/volumes

[48] J. Axboe, "fio - Flexible I/O tester," Last accessed: 2025-02. [Online]. Available: https://fio.readthedocs.io/en/latest/fio_doc.html

[49] Yahoo!, "Yahoo! Cloud Serving Benchmark," Last accessed: 2025-02. [Online]. Available: https://github.com/brianfrankcooper/YCSB/

[50] Meta, "RocksDB," 2012, Last accessed: 2025-02. [Online]. Available: https://github.com/facebook/rocksdb

[51] I. Laptev, "Hollywood2 human actions and scenes dataset," Last accessed: 2025-02. [Online]. Available: https://www.di.ens.fr/~laptev/actions/hollywood2/

[52] M. Vef, N. Moti, T. Süß, M. Tacke, T. Tocci, R. Nou, A. Miranda, T. Cortes, and A. Brinkmann, "GekkoFS - A temporary burst buffer file system for HPC applications," *J. Comput. Sci. Technol.*, vol. 35, no. 1, pp. 72–91, 2020.

[53] D. Zhou, V. Aschenbrenner, T. Lyu, J. Zhang, S. Kannan, and S. Kashyap, "Enabling high-performance and secure userspace NVM file systems with the Trio architecture," in *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023*. ACM, 2023, pp. 150–165. [Online]. Available: https://doi.org/10.1145/3615366.3615419

[39] M. Yan and K. Gopalan, "Performance overheads of confidential virtual machines," in *31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2023, Stony Brook, NY, USA, October 16-18, 2023*. IEEE, 2023, pp. 1–8. [Online]. Available: https://doi.org/10.1109/MASCOTS59514.2023.10387607

[40] L. Qiu, R. Taft, A. Shraer, and G. Kollios, "The price of privacy: A performance study of confidential virtual machines for database systems," in *Proceedings of the 20th International Workshop on Data Management on New Hardware, DaMoN 2024, Santiago, Chile, 10 June 2024*, C. Binnig and N. Tatbul, Eds. ACM, 2024, pp. 2:1–2:8. [Online]. Available: https://doi.org/10.1145/3662010.3663440

[41] M. Misono, D. Stavrakakis, N. Santos, and P. Bhatotia, "Confidential vms explained: An empirical analysis of AMD SEV-SNP and intel TDX," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 8, no. 3, pp. 36:1–36:42, 2024. [Online]. Available: https://doi.org/10.1145/3700418

[42] H. Lefeuvre, D. Chisnall, M. Kogias, and P. Olivier, "Towards (really) safe and fast confidential I/O," in *Proceedings of the 19th Workshop on Hot Topics in Operating Systems, HOTOS 2023, Providence, RI, USA, June 22-24, 2023*, M. Schwarzkopf, A. Baumann, and N. Crooks, Eds. ACM, 2023, pp. 214–222. [Online]. Available: https://doi.org/10.1145/3593856.3595913

[43] AMD, "AMD SEV-TIO: trusted I/O for secure encrypted virtualization," 2023, Last accessed: 2025-02. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/developer/sev-tio-whitepaper.pdf

[44] Intel, "TDX connect architecture specification," 2023, Last accessed: 2025-02. [Online]. Available: https://cdrdv2.intel.com/v1/dl/getContent/773614

[54] A. R. Martinelli, M. Torquati, M. Aldinucci, I. Colonnelli, and B. Cantalupo, "CAPIO: a middleware for transparent I/O streaming in data-intensive workflows," in *2023 IEEE 30th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 2023.