RSSD: Defend against Ransomware with Hardware-Isolated Network-Storage Codesign and Post-Attack Analysis

Benjamin Reidys
University of Illinois,
Urbana-Champaign, USA
breidys2@illinois.edu

Peng Liu
Pennsylvania State University,
University Park, USA
pxl20@psu.edu

Jian Huang
University of Illinois,
Urbana-Champaign, USA
jianh@illinois.edu

ABSTRACT

Encryption ransomware has become a notorious malware. It encrypts user data on storage devices like solid-state drives (SSDs) and demands a ransom to restore data for users. To bypass existing defenses, ransomware would keep evolving and performing new attack models. For instance, we identify and validate three new attacks, including (1) garbage-collection (GC) attack that exploits storage capacity and keeps writing data to trigger GC and force SSDs to release the retained data; (2) timing attack that intentionally slows down the pace of encrypting data and hides its I/O patterns to escape existing defense; (3) trimming attack that utilizes the trim command available in SSDs to physically erase data.

To enhance the robustness of SSDs against these attacks, we propose RSSD, a ransomware-aware SSD. It redesigns the flash management of SSDs for enabling the hardware-assisted logging, which can conservatively retain older versions of user data and received storage operations in time order with low overhead. It also employs hardware-isolated NVMe over Ethernet to expand local storage capacity by transparently offloading the logs to remote cloud/servers in a secure manner. RSSD enables post-attack analysis by building a trusted evidence chain of storage operations to assist the investigation of ransomware attacks. We develop RSSD with a real-world SSD FPGA board. Our evaluation shows that RSSD can defend against new and future ransomware attacks, while introducing negligible performance overhead.

CCS CONCEPTS

• **Computer systems organization** → **Secondary storage organization**; • **Security and privacy** → **File system security**; • **Applied computing** → *Evidence collection, storage and analysis.*

KEYWORDS

Solid-State Drive, Ransomware Attacks and Defenses, Storage Forensics, NVMe over Fabrics

ACM Reference Format:

Benjamin Reidys, Peng Liu, and Jian Huang. 2022. RSSD: Defend against Ransomware with Hardware-Isolated Network-Storage Codesign and Post-Attack Analysis. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '22), February 28 – March 4, 2022, Lausanne, Switzerland.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3503222.3507773

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ASPLOS '22, February 28 – March 4, 2022, Lausanne, Switzerland
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9205-1/22/02…$15.00
https://doi.org/10.1145/3503222.3507773

1 INTRODUCTION

Although secure storage systems have been developed for decades [8, 13, 43, 65, 83], encryption ransomware imposes new challenges and has become one of the biggest cybersecurity threats [2, 82, 85, 91]. It stealthily encrypts user data and demands ransom from users to restore their data. Recent studies report that ransomware attack could happen every 11 seconds [61], the victims include governments, schools, hospitals, police departments, transportation, and personal computers [3]. Each attack requests an average of $8,100 and costs nearly $300,000 in server downtime [62]. These ongoing ransomware outbreaks and global damage reflect the fact that the current security design of storage systems falls short of defending against encryption ransomware.

To defend against ransomware attacks, software-based approaches, such as intrusion detection [37, 70, 87] and data backup [15, 19, 84, 90] have been proposed. These detection systems rely on file access patterns to identify attacks. Unfortunately, recent studies show that ransomware can obtain OS kernel privilege to terminate or destroy these software-based solutions [30, 79]. Moreover, even though the ransomware detection succeeds, some files have been encrypted and victims still have to pay to get their data back.

An alternative approach is to develop protection mechanisms in the storage controllers. This makes ransomware defense isolated from software systems. For instance, prior work developed in-storage ransomware detection [12, 56] and data recovery mechanisms [30, 56, 80], based on the fact that SSDs can intrinsically retain obsolete data in flash chips for a period of time, until their data is reclaimed by the garbage collection (GC). However, these defenses were developed based on the assumption that ransomware rarely considers SSD properties.

Moreover, our study of 144 recent ransomware attacks confirmed that modern storage systems usually lack fast data recovery capability (see §2.1). Also, a majority do not support trusted post-attack analysis, which impedes the progress of recovering from an attack. Our investigation of state-of-the-art approaches (see Table 1) indicates that both existing software and hardware-based defenses lack the support of efficient data recovery and post-attack analysis.

As SSDs have become prevalent in a vast majority of computing platforms because of their increased performance and decreased cost [23, 39], we believe new ransomware attack models with awareness of flash properties (Ransomware 2.0) will happen, creating a new generation of security threats. These new attacks will not only encrypt user data, but also defeat existing data protection schemes.

Table 1: Comparison with state-of-the-art approaches. The 6th column represents Data Recovery (❍: Unrecoverable, ◗: Partially Recoverable, ●: Recoverable). The 7th column represents Post-Attack Analysis or Storage Forensics.

	Related	Defend New Attacks			Recovery	Forensics
		GC	Timing	Trimming		
Software	Unveil [37]	✗	✗	✗	❍	✗
	CryptoDrop [70]	✗	✗	✗	❍	✗
	CloudBackup [15]	✗	✔	✗	◗	✗
	ShieldFS [19]	✗	✗	✗	◗	✗
	JFS [84]	✗	✗	✗	❍	✗
Hardware	FlashGuard [30]	✔	✗	✗	◗	✗
	TimeSSD [80]	✔	✗	✗	◗	✗
	SSDInsider [12]	✗	✗	✗	◗	✗
	RBlocker [56]	✗	✗	✗	◗	✗
RSSD		✔	✔	✔	●	✔

In this paper, we first present three new ransomware attacks that can circumvent existing SSD-based protections: (1) *GC attack*, in which ransomware exploits the limited storage capacity of SSDs and keeps writing data to occupy the available storage space and force SSDs to release retained data. (2) *Timing attack*, in which ransomware intentionally slows down the pace of encrypting data and hides its I/O patterns. (3) *Trimming attack*, in which ransomware utilizes the *trim* command in commodity SSDs to erase flash pages such that SSDs will remove the original copies of encrypted data.

It is not easy to defend against these new attacks, since each of them will generate new I/O patterns that can bypass existing detection and defense mechanisms. For instance, existing detection approaches [16, 37, 38, 48, 70, 72] worked by identifying the repeating I/O patterns of read-encrypt-overwrite or read-encrypt-delete operations. The timing attack can bypass it by intentionally lowering the attack frequency and imitating regular storage I/O patterns. The GC attack can invalidate existing data recovery schemes by forcing the SSD to conduct GC operations and erase the retained data. And the trimming attack enables ransomware to speed up the removal of the original data copies that have been encrypted.

To defend against the GC and timing attacks, we develop a ransomware-aware SSD named RSSD, which enables zero data loss recovery by conservatively retaining all the stale data. Thus, RSSD guarantees that all data that may be locked by ransomware is retained. However, this incurs significant performance overhead. To overcome this, RSSD proposes a hardware-isolated NVMe over Ethernet (NVMe-oE) to transfer the retained invalid pages in a compressed and encrypted format to remote cloud or storage servers in time-series order, while keeping the valid pages locally for performance. This enables RSSD to expand its local storage capacity in a secure and transparent manner.

To mitigate the trimming attack, we rethink the hardware support for the *trim* command in SSDs. The feature of using *trim* commands to directly notify SSDs to garbage collect flash pages, would be attractive to ransomware, since it can bypass existing defenses. Instead of disabling the *trim* command, RSSD enhances it. Specifically, upon receiving *trim* commands, RSSD will allocate new flash pages and remap the addresses touched by the *trim* command to these new pages. RSSD will retain the trimmed data, therefore, it can still restore the victim data upon trimming attack.

To enable efficient and trusted post-attack analysis, we extend RSSD and retain the log of storage operations in the SSD. Thus, RSSD has the capability of reproducing the storage operations in the original order that they were issued. As the logging approach is hardware isolated, RSSD can build a trusted evidence chain for post-attack or forensic analysis. Since most of the retained logs and obsolete data will be transferred remotely, RSSD enables the offloading of ransomware detection and analysis to remote servers. Therefore, we can detect ransomware in a more efficient and accurate way by utilizing the powerful computing resource and the flexibility of deploying various detection algorithms. RSSD also allows fast reconstruction of evidence chains by backtracking storage operations with our proposed hardware-assisted logging. We summarize the comparisons with state-of-the-art approaches in Table 1, and list the key contributions of this work as follows.

- We conduct an empirical study of more than a hundred ransomware cases and confirm that the lack of efficient data recovery and post-attack analysis is the major weakness of modern storage systems and ransomware defense solutions.
- We present a new understanding of Ransomware 2.0, discuss and validate three new ransomware attacks that include GC attack, timing attack, and trimming attack.
- We develop a new SSD that uses the hardware-isolated NVMe-oE to safely extend the retention time for the hardware-assisted logging, and enable the offloading of ransomware detection and analysis to the remote cloud/servers.
- We rethink the storage architecture support for the *trim* command in SSDs and enhance its security by enabling the retention of trimmed data with hardware-assisted logging.
- We present a hardware-isolated post-attack analysis approach that can efficiently build a trusted evidence chain of storage operations that lead up to an attack.

We implement RSSD with a Cosmos+ OpenSSD FPGA board, a cloud storage service Amazon S3, and local storage servers. We use various storage benchmarks and I/O traces, and ransomware samples collected from VirusTotal [79]. Our evaluation shows that RSSD can retain all obsolete data across the SSD and remote cloud/servers, with minimal storage cost, and negligible impact on the local storage performance and device lifetime. We replay multiple ransomware attacks on RSSD, and show that RSSD can restore the data encrypted by ransomware, and correctly reconstruct the original sequence of I/O events that lead to the attacks in a short time.

2 BACKGROUND AND MOTIVATION

2.1 State-of-the-Art Ransomware Attacks

Encryption ransomware reads the target files and encrypts them. After that, they can either use the encrypted data to overwrite the original data, or write the encrypted data to new files while deleting the original copies. To understand recent ransomware attacks, we conduct an empirical study of major attacks between Jan. 2020 and Mar. 2021. We manually studied 144 cases listed in [3], with a focus on those having detailed attack descriptions. We categorize these cases into 8 sectors and show the study result in Table 2.

Table 2: A study of 144 recent major ransomware attacks.

Sector	Count	Downtime (weeks)	Software Backups	Forensics	
				Conducted	Time (weeks)
Government	31	4.39	61.29%	16.12%	7.40
Healthcare	31	2.88	38.71%	41.94%	7.31
School	21	2.32	52.38%	14.29%	5.00
Technology	17	1.80	29.41%	23.53%	4.25
Transport	11	5.65	45.45%	27.27%	6.14
Financial	6	4.33	33.33%	0.00 %	-
Police	4	0.57	75.00%	0.00 %	-
Other	23	2.36	34.78%	26.09%	2.00
Total	**144**	**3.12**	**45.13%**	**23.61%**	**5.72**

Previous studies have reported that 32% of the victims paid the ransom, but only got 65% of their data back on average [64]. Additionally, 22% of them never got any data back [27]. Beyond these, we observe two new facts:

New Observations: (1) the victims suffered from service downtime of 3.12 weeks on average, due to the tedious data recovery procedure; (2) only 23.61% of the victims have post-attack analysis for their storage systems, and the analysis usually takes 3 days to 5 months, and 5.72 weeks on average; (3) 45.13% of the cases on average have software-based data backups, however, many of them still suffer from service downtime and even data loss. In RSSD, we aim to achieve full data recovery with minimal service downtime and trusted post-attack analysis by default.

2.2 State-of-the-Art Ransomware Defenses

As reported in our study presented above, many victims rely on software-based data backups to recover victim data from attacks. Unfortunately, software-based solutions suffer from four major limitations. First, since software-based solutions are not hardware isolated from malicious processes, they can be compromised by ransomware. Particularly, attackers could obtain OS kernel privilege and terminate software-based backup systems. Second, attackers can hide in the system long before deploying their ransomware. For example, prior work has confirmed that hackers can hide in the victim's network for an average of 11 days before deploying their ransomware [89]. Third, ransomware can overwrite data backups with encrypted versions [30, 37]. Finally, software-based solutions usually lack the capability of trusted post-attack analysis.

To defend against ransomware attacks, the most recent work [12, 30, 56] exploited the intrinsic flash properties to detect ransomware attacks and restore the victim data. However, these solutions have three major limitations.

First, they were mainly developed to defend against existing encryption ransomware which assumed the underlying storage devices perform like conventional HDDs [37, 70]. As SSDs have been widely used, ransomware will evolve and update their attack models (see §4). Therefore, we must anticipate and proactively prevent new and emerging ransomware attacks.

Second, due to the limited storage capacity, we can only retain the stale data for a certain period of time. This will significantly affect storage performance, especially for data-intensive workloads. Even worse, ransomware could take advantage of limited storage capacity to initiate new attacks.

Third, most defenses do not support post-attack or forensic analysis, which will miss the opportunity to learn new attack models. This slows down the post-attack investigation procedure, and limits their ability to adapt to evolving malware.

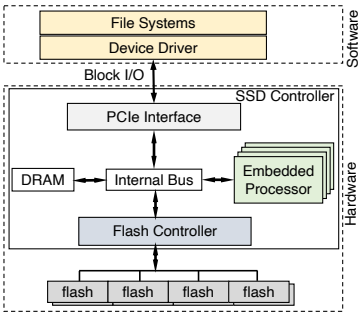


Figure 1: System architecture of flash-based SSDs.

2.3 Flash-Based Solid-State Drive

To facilitate our discussion, we introduce the technical background of flash-based SSDs in this section. We present the system architecture of an SSD in Figure 1.

An SSD has three major components: a set of flash memory packages, an SSD controller that has embedded processors with DRAM, and flash controllers. Commodity flash-based SSDs employ a block interface to encapsulate the idiosyncrasies of flash chips. As such, it gives upper-level file systems an impression that both flash-based storage devices and HDDs perform storage operations in the same manner.

When a free flash page is written, that page cannot be updated until it is erased. However, the erase can only be performed at a block granularity. Thus, writes are issued to free pages that have been erased (i.e., out-of-place writes). GC is performed later to clean the obsolete data. As each flash block has limited endurance, it is important for the blocks to age uniformly (i.e., wear leveling). SSDs employ the flash translation layer (FTL) in their controllers to handle out-of-place writes, GC, wear leveling, and to maintain the logical-to-physical address mapping. To exploit the massive parallelism of flash chips, SSD controllers use general-purpose processors and DRAM. The processors will issue I/O requests, translate logical addresses to physical addresses, and run GC. DRAM is used to store the address mapping table and cache the I/O data.

3 THREAT MODEL

As discussed in §1, malicious users could elevate their privilege to run as administrators and disable/destroy the software-based data backup solutions. We do not assume the OS is trusted, instead, we trust the SSD firmware. We believe this is a realistic threat model for two reasons. First, the SSD firmware is located within the storage controller, underneath the generic block interface. It is hardware-isolated from higher-level malicious processes. Second, SSD firmware has a much smaller trusted computing base (TCB) than the OS kernel, making it typically less vulnerable to malware attacks. Once the firmware is flushed into the SSD controller, commodity SSDs will not allow firmware modifications without authentication, which guarantees the implementation integrity.

Once malicious events are recognized (e.g., malicious attacks are detected or ransomware notification is received) or users want to recover data, they can use RSSD and remote server/cloud to conduct the data recovery procedure. Malicious programs may exploit the data recovery procedure to conduct new attacks. For example,

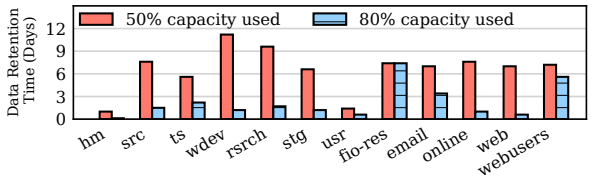


Figure 2: Data retention time for different applications.

adversaries could recover the data in the SSD by rolling back the storage states to a previous version, and then try every possible means to erase the retained data. RSSD can defend against such attacks because it will roll back data by writing them to the local SSD like new updates and will retain the old versions. Moreover, the data recovery can be conducted in a secure and isolated environment. For instance, the device owner can deploy the SSD in another isolated computer to conduct the data recovery. We assume only the device owner or trusted user has the privilege to access the data stored in the remote cloud or storage servers.

Compared to the local storage, remote cloud or storage servers offer guaranteed data reliability [10, 20] and the flexibility to expand storage capacity with low cost (see §8.1). RSSD is compatible with existing cloud storage platforms. It always encrypts the transferred data to enhance its data security on the remote cloud or servers.

RSSD makes the network module hardware isolated from malware attacks. We view the data encryption and hardware isolation as mitigating factors in network related attacks. A potential approach to attacking RSSD would be board-level physical attacks that exploit side-channel attacks or bus-snooping attacks to reveal transferred data. However, these attacks are extremely cumbersome [33]. The majority of ransomware attacks are conducted remotely via phishing emails, exploit kits, or network misconfigurations [4]. Therefore, we do not consider these physical attacks in this work.

4 NEW RANSOMWARE ATTACKS

As SSDs use the same block interface as HDDs, existing ransomware attackers did not consider the intrinsic properties of SSDs in their attacks [37, 70]. Prior researchers [12, 30] leveraged this property to prevent attacks. However, ransomware will evolve and circumvent these solutions. It has two ultimate goals: ① prevent victims who do not pay the ransom from obtaining copies of their data; and ② bypass ransomware detection schemes. We identify three new ransomware attacks that include the *GC attack*, *timing attack*, and *trimming attack*. We will undertake an analysis for each of them and demonstrate how they can bypass existing SSD-based ransomware defenses as well as ensure the victim data is deleted.

4.1 GC Attack

SSDs usually over-provision a certain ratio of flash blocks for GC (i.e., over-provisioning). Once the number of free blocks in the SSD is below a threshold (10–40% of all the flash blocks), GC will be triggered to free space. An attacker can exploit the storage capacity and write/overwrite data to the SSD to occupy its available space, forcing it to release its retained data via GC. Thus, the device-level data retention will fail. The GC attack can be fulfilled shortly. Given a 1TB SSD with 2.2GB/s write bandwidth [68], the SSD will be full in 7.8 minutes. The GC attack will help attackers achieve ①. Even if it can be detected, part of the user data could be lost and victims still need to pay the ransom to get their data back.

Furthermore, ransomware attackers can hide their behavior by following the patterns of regular storage workloads, which enables attackers to achieve ②. Once the victim data is dropped by the GC, the ransomware attack will succeed. For this attack, there is a fundamental tradeoff between storage performance and security. The lifespan of retaining stale data could be short for performance, however, the users would suffer from the threat of ransomware attacks or data loss.

4.2 Timing Attack

Ransomware can intentionally slow down the pace of encrypting data and notifying victims. Although this will increase the risk of being caught, and thwart ransomware authors from gaining rewards quickly, the I/O pattern generated by this attack will be hard to identify. This will help attackers fulfill both ① and ②.

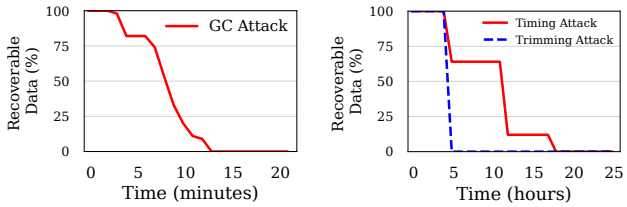
Due to the limited storage capacity, it is challenging to retain stale data for a long time in SSDs. To increase the difficulty of being detected by analyzing the I/O patterns [37, 70], attackers could mimic regular storage operations from common applications. By then, the victim data would have been garbage collected, making the SSD protection in vain.

To further understand this attack, we use traces collected from enterprise servers and universities, and replay them in a 1TB SSD with different used capacity (see the experimental setting in §7.1). As shown in Figure 2, the stale data can be retained for 0.9-11.2 days when 50% of the storage capacity has been used. When less storage capacity is available (80% capacity used), the SSD can retain the stale data for only 0.12-7.4 days. As regular users would also generate I/O traffic, the retention time could be even shorter.

4.3 Trimming Attack

Ransomware authors can also exploit the *trim* command available in modern SSDs [52] to initiate new attacks. The *trim* command was introduced in commodity SSDs to allow the OS to inform the SSD which data blocks are no longer in use and can be erased. This can reduce the GC overhead as the SSD can safely erase the block without first moving pages. It is also an attractive feature for ransomware attackers. Encryption ransomware can leverage this command to speed up the reclamation of the original data blocks after encryption, helping attackers achieve ①. Specifically, these blocks will be erased at the next GC. With write-intensive workloads, this can happen within a few seconds [45], leaving no copies of the data encrypted by ransomware stored in the SSD.

The trimming attack also enables attackers to achieve ②, because no existing detection approaches consider *trim* operations in their learned I/O patterns. Existing defenses could evolve and learn new I/O patterns like read-encrypt-trim, however, part of the user data could have been erased as discussed above. The fact that attackers can utilize the *trim* command to generate various new attack patterns reiterates the need of a holistic approach to defend against future trim-based attacks.



(a) GC Attack. (b) Timing and Trimming Attack.

Figure 3: Effectiveness of the proposed new attacks.

4.4 Proof-of-Concept Attack Prototypes

We develop the proof-of-concept prototypes with the ransomware samples listed in [63]. We only need to instrument a few lines of codes (LoCs) to initiate these attacks. The GC attack needs less than 10 LoCs to write garbage data to the SSD, the trimming attack requires only a few LoCs to call *trim* in the samples, and the timing attack only needs a timer to control the window notifications.

To verify the effectiveness of these new attacks, we use a 1TB SSD (see the detailed setting in §6). We first occupy half of its capacity with valid files, and run WannaCry ransomware to encrypt some of the files (128MB). After that, we initiate these new attacks respectively and measure how much modified data can be recovered over time. We present the results in Figure 3. Since GC attack can quickly generate garbage data, the GC is triggered shortly and all the victim data is garbage collected in a few minutes. After we initiate timing and trimming attack, we run the storage traces of database workload TPCC (see the detailed description in §7.1) to emulate regular storage operations. After about 4 hours, GC is triggered, and the victim data will be reclaimed by the SSD. As shown in Figure 3b, the trimming attack accelerates the loss of victim data, as it notifies the SSD which flash blocks can be reclaimed.

We believe these attacks would appear in the near future for two reasons. First, our proof-of-concept attacks demonstrate that, with only a few lines of code, these attacks can be implemented. With such a low barrier, ransomware authors can easily adopt them. Second, attackers can only make a profit if they can prevent victims from recovering their data. Thus, there is a high possibility that attackers will build SSD-aware exploits to remain profitable, as SSDs are becoming the dominant storage devices today.

5 RSSD DESIGN

5.1 Key Ideas of RSSD

A fundamental approach to defending against ransomware is to enable zero data loss recovery and trusted post-attack analysis. Due to the limited storage capacity in the local SSD, we have to rely on the remote cloud or storage servers to retain data. If we can ensure the remote data backups are securely isolated, we can restore the victim data without paying the ransom. This will eventually stop ransomware, as attackers cannot profit anymore.

RSSD proposes a hardware-isolated network/storage co-design to ensure security isolation for remote cloud/servers. Therefore, we can recover victim data quickly and shorten the system downtime. RSSD also enables trusted post-attack analysis with our proposed hardware-assisted logging. Thus, we can reconstruct the trusted evidence chain, which will help victims and government agencies to investigate the attack and defend future attacks. We develop RSSD with the following key ideas:

- **Extend SSD with Secure NVMe over Ethernet (§5.2, §5.3, and §5.4):** We develop RSSD with secure NVMe over Ethernet to extend the local SSD. Unlike conventional ways of communicating with remote cloud or servers via host network cards, we integrate Ethernet network into the SSD controller, making the network hardware isolated from host.
- **Enhance the Security Support for *trim* Command (§5.4):** We redesign the hardware support for the *trim* command in the SSD controller, with the insight that the *trim* command is attractive for ransomware attackers. RSSD will track *trim* history and retain the potential victim data with hardware-assisted logging.
- **Data Recovery and Post-Attack Analysis (§5.5 and §5.6):** As RSSD will transparently retain stale data in both local SSD and remote cloud/server in a time order, it enables data recovery by retrieving old versions of updated data. It can build trusted evidence chains for attack investigations.

5.2 Design Overview of RSSD

Our investigation of Ransomware 2.0 attacks (see §4) shows that stronger data retention is critical to fundamentally preventing future ransomware attacks. However, since the SSD firmware has no semantic knowledge of upper-level software, we have to conservatively retain all invalid flash pages to ensure zero data loss recovery. This strategy guarantees that all the data that may be locked by ransomware is retained. Unfortunately, this will add additional performance overhead to the SSD. Thus, we have to either sacrifice the storage performance or shorten the data retention time.

One solution to address this challenge is to expand local storage capacity by transferring the retained data to remote cloud or storage servers. However, with the conventional system architecture that decouples storage/network, we have to read data from the SSD to the host and transfer the data out through the network card. However, adversaries with kernel privileges could easily terminate this process or destroy user data (see §1). Thus, we seek a hardware-isolated solution with low cost and strong security guarantees.

To this end, we explore NVMe over Fabrics (NVMoF) [55] and integrate the network component into the SSD controller to enable secure data transfer to the remote cloud/servers. Thus, we can transfer data without the host involvement.

This is a practical solution for three reasons. First, the NVMoF has become a standard protocol for NVMe devices [54]. This has driven the recent development of real products [7, 35, 36]. Second, following the industry specifications, there is no technical barrier to hinder SSD vendors from implementing NVMoF in real products. And it does not introduce much hardware cost to existing SSD manufacturing (i.e., less than 5% area overhead for supporting the Ethernet interface [77]). And the integrated Ethernet network controller will add only 5.7%–19.2% more power consumption for a commodity SSD [69, 76]. Third, considering ransomware asks for an average of $8,100 and causes long server downtime [62], it is worthwhile for our community to develop a thorough solution.

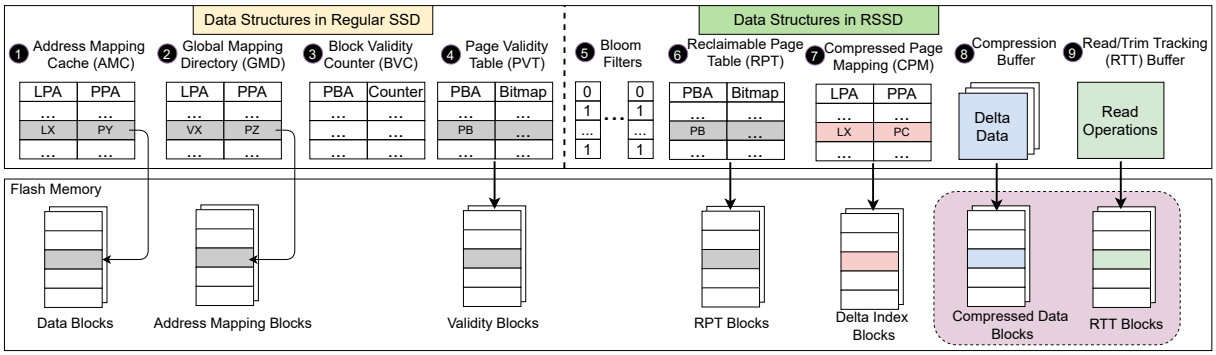


Figure 4: Key structures used in the SSD controller of RSSD.

5.3 Design Details of RSSD

Using NVMe-oE to defend against ransomware is not easy. We have to answer three questions: (1) what data should be transferred to the remote cloud or servers? (2) how should we manage the data to facilitate the data recovery procedure and post-attack analysis? (3) how should we ensure the overall efficiency? To address these challenges, we build RSSD with the following techniques.

5.3.1 Data Structures in the Regular SSD Firmware. To facilitate our discussion, we first introduce the data structures used in traditional SSD controller in Figure 4. To support out-of-place updates, the SSD maintains an address mapping table for translating logical page addresses (LPAs) of storage requests to physical page addresses (PPAs). The mapping table is cached in the DRAM of the SSD controller (see ❶ AMC in Figure 4). The entire mapping table is stored in the flash chips as a set of translation pages, and their locations are tracked in ❷ global mapping directory (GMD) [25, 30].

In a regular SSD, the GC will first select the candidate flash blocks (e.g., those flash blocks who have the least valid pages), migrate the valid pages of the selected blocks to a free block, erase the selected flash blocks at block granularity, and mark them as free blocks. To assist the GC, the SSD controller usually has a ❸ block validity counter (BVC) table that tracks the number of valid pages for each flash block. It also maintains a ❹ page validity table (PVT) that uses bitmaps to track which pages are valid in a flash block. During the GC, the address mapping table will be updated accordingly when a valid page is copied from one flash block to another one.

5.3.2 Retaining Stale Data in Time Order. RSSD conservatively retains stale data in a time order. It requires minimal firmware changes with five new data structures (see Figure 4).

Instead of tracking the timestamp of each flash page when it becomes invalid, RSSD uses ❺ bloom filters to index invalid pages. It organizes multiple bloom filters in time order, and each bloom filter represents one time window or epoch, such that RSSD can maintain the ordering of invalid data versions. Specifically, once a flash page is invalidated, its PPA is inserted into the most recently created bloom filter. Once the number of PPAs in the bloom filter reaches a threshold, the bloom filter become inactive and a new bloom filter is created. RSSD will reuse the created bloom filters in the order of their creation. When RSSD resets the oldest bloom filter and reuses it as the latest one, it will implicitly remove the indexed PPAs in the bloom filter.

When GC reclaims an invalid page, it will check the bloom filters. If the PPA of an invalid page is found in a bloom filter, the page will be retained and compressed (see §5.3.3). Each bloom filter is associated with dedicated data blocks for retaining the compressed flash pages that become invalid in that epoch. Although bloom filters have false positives, they do not cause incorrect behaviors. This is because even though an invalid page could be reclaimed, retaining it in the SSD conservatively will not generate much negative impact. Bloom filters do not have false negatives, invalid pages that should be retained will not be reclaimed by mistake.

5.3.3 Packing Retained Stale Data with GC. Retaining all stale data will consume significant storage space, RSSD uses delta compression for retained stale data, as only a small portion is usually changed in a page update [86]. RSSD has ❽ delta buffers to group deltas (update differences) at page granularity. Once a delta buffer is full, RSSD writes it back to flash blocks, and updates the mapping table for those compressed pages (❼ CPM) for future retrieval. For invalid pages that have been compressed, RSSD will update the ❻ reclaimable page table (RPT) to indicate that they can be reclaimed.

RSSD modifies the GC procedure of SSDs to compact the invalid pages and clean them for free space. RSSD reclaims invalid pages after they have been compressed. The compressed stale data will be reclaimed after they have been transferred to the remote cloud/server (see § 5.3.5). RSSD uses delta compression during GC. It computes the difference between two versions of the page mapped to the same LPA, and uses the difference as the delta to represent the invalid page. RSSD uses the latest data version mapped to the LPA as the reference. Because RSSD retains the stale data in time order, a reference version will never be reclaimed before its deltas.

RSSD uses a ❽ compression buffer to coalesce deltas in a page. We organize the delta page as shown in Figure 5. The header of each delta page includes the number of deltas and the byte offset of each delta. Each delta item has the metadata and the delta value. The delta metadata includes (1) the LPA mapped to this compressed page; (2) the PPA that points to the previous data version mapped to the same LPA (i.e., back pointer); (3) the write timestamp of this data version; and (4) the PPA of the reference flash page.

Once an invalid page is compressed, RSSD will set the ❻ RPT to indicate that it can be reclaimed. Utilizing the embedded processors

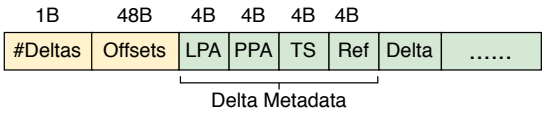


Figure 5: The Structure of a delta page.

Algorithm 1: GC Procedure of RSSD

```
Check ❸ block validity counter
Select a candidate block with least number of valid pages
Identify valid/invalid pages by checking ❹ PVT
for each valid page do
    Migrate this page to a free page
    Update the ❶ address mapping table
for each invalid page do
    Check the ❻ reclaimable page table
    if this page is reclaimable then
        Discard this page (compressed or expired)
    else
        Check if this page is in the ❺ bloom filters
        if this page misses all the bloom filters then
            Discard this page as it has been expired
        else
            Read this page and its OOB metadata
            Read all the older and unexpired data versions
            Read the latest version mapped to this LPA
            Compress the old versions with the ref. version
            Write deltas to delta blocks with metadata
            Update the head of delta page chain in ❼ CPM
            Set compressed pages as reclaimable in ❻ RPT
Erase the selected flash block
```

available in the SSD, RSSD executes the lightweight delta compression with the GC. When the compression buffer is full or its space cannot host a newly compressed page, RSSD will write it to the reserved flash blocks. We show the GC procedure in Algorithm 1.

RSSD uses an address mapping table for those compressed invalid pages (❼ CPM). With the back pointer in the delta metadata, RSSD maintains the chain of all invalid pages mapped to the same LPA. As each physical flash page has a reserved out-of-band (OOB) metadata (16-64 bytes) [25], we use it to store (1) the LPA mapped to this physical flash page; (2) the previous PPA mapped to the same LPA; (3) the timestamp when the flash page is written.

5.3.4 Tracking Read and Trim Operations in SSD. Beyond retaining the invalid pages, RSSD logs read and *trim* operations in a log-structured manner. Each log item has 8 bytes (4 bytes for each timestamp, 4 bytes for each LPA). The resulting storage cost is small (800MB for tracking 100 million storage operations, which takes 0.076% storage capacity of a 1TB SSD). RSSD tracks the read and *trim* operations in a separate buffer (1MB by default). Once the buffer is full, it will be written into corresponding flash blocks. Note that RSSD does not explicitly track write operations, because it tracks write timestamps of each flash page in the OOB (see §5.3.3).

This detailed logging enables trusted storage forensics analysis. First, it enables the reconstruction of the event chain leading up to an incident. We use the previous physical page address stored in the OOB metadata of each page to reverse an invalid page to its previous versions. In this fashion, we can build the evidence chain of the storage operations (see §5.5). Second, since the RSSD never misses an event, those events representing the occurrence of anti-forensics

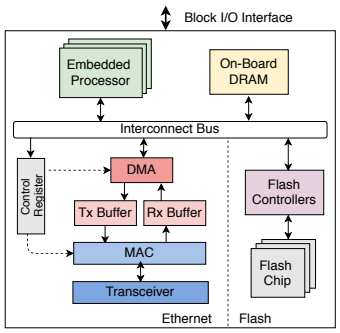


Figure 6: Secure SSD with NVMe over Ethernet.

will also be recorded. Third, with the trusted evidence chain, we only need to analyze operations involved in attacks, which reduces the cost of conducting the post-attack analysis.

5.3.5 Transferring Retained Data via NVMe-oE. Ideally, we wish to expand the local storage with unlimited storage space in a secure and cost-effective way. We present the hardware architecture of RSSD in Figure 6. As the network module is integrated into the SSD controller, it is not exposed to the host. It utilizes DMA to transfer flash blocks from the DRAM of the SSD controller to the transmit (Tx) and receive (Rx) buffer queues. The network module has an unique MAC address and can be configured with an IP address to communicate with other networked devices. As NVMe-oE supports TCP/IP protocol, it does not have specific requirements for low-level network devices like wireless network.

Although NVMe-oE is hardware isolated in RSSD, its IP address and connected IPs can be configured by issuing NVMe commands enabled in modern NVMe protocols [41]. For example, end users can configure the IP addresses of remote servers or the access URLs of remote cloud services. Note that the initial configuration procedure should be conducted in a secure environment. Since we do not configure frequently, it is disabled at runtime to avoid attacks.

RSSD transfers the compressed stale data and tracked read/trim operations to the remote cloud or servers at flash-block granularity, after which these flash blocks will be erased with GC by setting their corresponding ❸ BVC to zeros, as well as ❻ RPT to indicate these pages can be reclaimed.

RSSD conducts the data transfer at idle I/O cycles. It predicts the next idle time based on the last interval between I/O requests [11, 49]. Specifically, it predicts the next idle time ($t_i^{predict}$) based on the last interval of time between I/O requests (t_{i-1}^{real}) with $t_i^{predict} = \alpha * t_{i-1}^{real} + (1-\alpha) * t_{i-1}^{predict}$, where $t_{i-1}^{predict}$ refers to the idle time of the last prediction, and α refers to the exponential smoothing parameter ($\alpha = 0.5$ in RSSD). Once $t_i^{predict}$ is larger than a defined threshold (50 millisecs by default), RSSD will first send the oldest compressed blocks and RTT blocks to the remote cloud/servers.

To ensure secure data transfer, RSSD has data encryption of each flash block being transferred. It uses the 128-bit AES scheme, which has been deployed in modern SSDs [21, 69]. The encryption key is stored in the SSD firmware.

RSSD will not affect the functionality of regular storage operations. It only moves compressed invalid pages and RTT blocks to

the remote cloud/server as they will not be accessed by regular applications. To alleviate the negative impact on storage performance, RSSD will suspend the transfer upon I/O requests. When a flash block is being transferred, RSSD will finish in the background. Since RSSD transparently moves stale data to the remote cloud/servers, it will only refuse writes when full with valid data.

5.3.6 Managing Data in Remote Cloud/Server. We manage the compressed data blocks and RTT blocks in a log-structured manner in the remote cloud/servers. We use objects to store each flash block in the cloud storage services like Amazon S3, and use the arrival timestamp to name the object. As for the in-house storage servers, we use log-structured file systems such as F2FS [44] to store received flash blocks.

Storing retained stale data and operation logs in the remote cloud/server has four benefits. First, the cloud/server has much larger storage capacity, it can extend the local storage capacity. In this manner we can retain victim data as long as possible, while having minimal impact on the local storage performance. Second, cloud storage offers nearly 100% data reliability with replicas [10, 20], which further enhances the capability of defending against ransomware attacks. Third, the cloud/server has much more powerful computing resource, which facilitates the data recovery and post-attack analysis. For instance, users can apply advanced machine learning models in the cloud/server to learn and detect new ransomware attacks, which is impossible in the local storage. Finally, the cloud provides the flexibility of increasing the storage capacity with a much lower cost following the pay-as-you-go model [5, 9]. According to our cost-effectiveness analysis (see the detailed discussion in §8), the hardware-isolated cloud backup can achieve 85.6× cost reduction, in comparison with the alternative approaches that extend the storage capacity locally.

It is worth noting that, to defend against ransomware, we need to have frequent data backups anyway today. RSSD does not introduce much additional storage cost. Moreover, RSSD has no intention of replacing existing cloud storage. Instead, it provides a hardware-isolated network/storage co-design solution to make the connections to the data backups secure and transparent. Therefore, attackers cannot stop the remote data backups, even if they obtain the OS kernel privilege. RSSD allows end users to register remote cloud connections in the SSD (see §5.3.5) or to rely on the SSD vendors to use pre-registered accounts in their own cloud.

5.4 Defend Against New Ransomware Attacks

Defend against GC attack. As attackers initiate a GC attack by writing data to the SSD (see §4.1), the victim data that has been encrypted could be erased in regular SSDs. However, with RSSD, the original copies of the victim data will be transferred to the remote cloud/server via hardware-isolated NVMe-oE. Since we have much larger storage capacity in the remote cloud/server, RSSD can retain the victim data for a long time with lower cost (see §8.1).

Without RSSD, an SSD can refuse writes rather than erasing retained data, however, this will significantly decrease the performance of regular storage operations. Since an SSD does not know whether the stale data is generated by regular applications or ransomware, RSSD achieves the best trade-off between performance and security, it can defend against potential GC attacks while ensuring high performance for regular storage workloads.

Defend against timing attack. Although ransomware attackers can intentionally slow down their attack (see §4.2), the victim data will not be erased until it has been transferred to the remote cloud/server. As the cloud storage services offer extremely low storage cost, it is reasonable to retain victim data until users confirm that no ransomware attack happened or the data integrity is ensured. As the NVMe-oE is hardware isolated in RSSD, it is hard for adversaries to exploit system vulnerabilities to terminate or hijack the network connection with the remote cloud/server.

In the worst case, the network connection with the remote cloud/server is broken or the remote servers/cloud are not available, RSSD will best utilize the local storage capacity with data compression to retain the stale data locally until the SSD is full. After that, it will stop issuing I/O requests, resulting in the failure of filesystem operations. Victim users can easily notice the abnormal events.

Defend against trimming attack. RSSD keeps the *trim* command in use. However, we modify its operations in the SSD controller. RSSD tracks the *trim* commands and records the corresponding flash block addresses. Instead of reclaiming these flash blocks, RSSD retains them. RSSD allocates new flash blocks and remaps the addresses touched by the *trim* command to these newly allocated blocks by setting the ❶ address mapping table. Therefore, it gives attackers an illusion that the *trim* command is successfully executed. RSSD will mark the trimmed data as invalid and insert their PPAs into a ❺ bloom filter. Such that they will be compressed, encrypted, and transferred to the remote cloud or storage servers. By transferring the trimmed data to the remote cloud/servers, RSSD minimizes its impact on the local storage performance.

Defend against future attacks. We believe that RSSD is a holistic solution that can defend against future ransomware attacks for four reasons. First, with the zero data loss recovery capability, RSSD can ensure victims recover their data against attempts to remove it. Second, in designing RSSD, we considered the potential interactions the upper-level software can have with the SSD controller. Since RSSD is a firmware-level solution, it is hardware isolated from software, with a much smaller TCB. Third, RSSD provides hardware-assisted post-attack analysis, which allows it to track the behaviors of ransomware, although ransomware will evolve and new attack patterns will happen. The analysis results will facilitate developers to deploy new defense solutions. Finally, RSSD enables advanced ransomware detections by utilizing the powerful compute resources in the remote cloud/servers.

5.5 Data Recovery

RSSD utilizes three sources of retained stale data for its data recovery: (1) the invalid pages that have not been compressed yet; (2) the compressed invalid pages; (3) the compressed stale data that has been transferred to the remote cloud/server. Upon data recovery, RSSD can roll back the storage states to a previous version for a specific LPA or a set of LPAs. Given an LPA, RSSD first retrieves its previous version by checking the retained stale data, and then writes back the retrieved version to the SSD. It will then invalidate the current version and update the corresponding address mapping.

Specifically, to retrieve a previous version of an LPA at specific timestamp, RSSD first checks its latest version with the ❶ address mapping table, and uses the OOB metadata to pinpoint previous versions one by one. If the previous version at a specific timestamp cannot be found, RSSD will check the ❼ CPM to pinpoint the compressed versions. Following the chain built with the delta metadata, RSSD traverses back to previous versions and check its timestamp until it finds the demanded version. If RSSD cannot find the version in the local SSD, RSSD checks the flash blocks stored in the remote cloud/server. Since each compressed flash page has its metadata (see Figure 5), and the flash blocks in the remote cloud/server are transferred and stored in time order, we can scan them to identify the previous versions of an LPA.

5.6 Post-attack Analysis

To facilitate the post-attack analysis, we need to build trusted evidence chains, which requires complete records of storage operations. As discussed in §5.1, the key to defending against ransomware is to enable zero data loss recovery and trusted post-attack analysis. Therefore, a fine-grained hardware-isolated logging mechanism is desirable. RSSD enables trusted users to reconstruct the original sequence of events that lead to the incident (i.e., storage forensics). Unlike existing forensics techniques that execute under OS control, RSSD collects storage operations in the SSD firmware. Since RSSD has tracked the read/trim operations with ❾ RTT blocks, it can reproduce the storage operations in the order they were issued.

As a large portion of the operation log is available in the remote cloud or servers, it enables users to utilize their powerful computing resources to conduct advanced analysis, such as machine learning based ones, for ransomware detection, post-attack investigation, and identification of new attack patterns. New detection and analysis algorithms are out of the scope of this paper, we wish to investigate them as future work.

RSSD tolerates power outages and system crashes. This is because most of the data structures for RSSD are used to cache index/metadata information for fast access, they can be reconstructed by scanning the OOB metadata of flash blocks. And many SSDs today have deployed battery-backed DRAM, capacitors, or Power Loss Protection [57] mechanisms to ensure the durability of the buffered data in the SSD controller.

5.7 Put It All Together

We now summarize how RSSD serves storage operations. As RSSD is a firmware-based solution, it does not require modifications to the host systems software such as operating systems and file systems, as well as upper-level applications.

Read Operation. Upon receiving a read request, RSSD first checks ❶ AMC for address translation. If the mapping entry is in AMC, RSSD gets the PPA and serves the flash page. If not, RSSD looks up ❷ GMD to locate the mapping entry in the address-mapping page, and place the entry in ❶ AMC. This read operation will be tracked in ❾ RTT buffer.

Write Operation. Upon a write, RSSD will conduct the same address translation procedure as for reads. For a cache hit in ❶ AMC, RSSD will write data to a new flash page and update the corresponding mapping entry in ❶ AMC with the new PPA. For a miss, it will create a new mapping entry in ❶ AMC.

Trim Operation. Upon receiving a *trim* command, RSSD will conduct the address translation to locate the corresponding block *X*. It will also allocate a free block *Y*. RSSD will read the OOB of each page in the block *X* to find its LPA, and map each LPA to pages in *Y* one by one by updating the mapping entries in ❶ AMC. This gives the host program an impression that the *trim* operation is fulfilled. Internally, RSSD retains the block *X* by inserting its PPAs into the ❺ bloom filter. Thus, RSSD can recover the victim data caused by the trimming attack. Since these pages will soon be compressed, we largely retain the performance benefits of *trim*.

Table 3: Workloads used in evaluating RSSD.

Name	Description
MSR [51]	Storage traces from enterprise servers.
FIU [22]	Storage traces from computers at FIU.
OLTP [73]	An open-source database engine Shore-MT.
IOZone [34]	A benchmark of various file operations.
PostMark [53]	A benchmark that emulates mail servers.

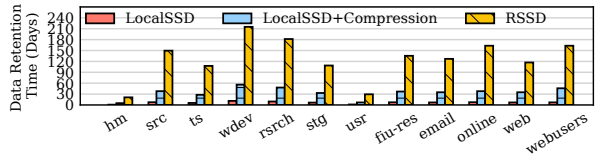
6 RSSD IMPLEMENTATION

We implement RSSD with a Cosmos+ OpenSSD FPGA development board [77] which supports the NVM Express (NVMe) protocol and NVMe-oE. This board includes a ARM Cortex-A9 Dual-core, 1GB DRAM, and 1TB flash memory. We reserve 15% of the capacity as over-provisioning space by default, which is also the trigger condition for the GC of the SSD. In the SSD, each flash page is 4KB with 12 bytes of OOB metadata, and each flash block has 256 pages.

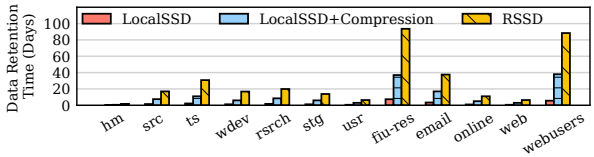
Besides supporting the basic I/O requests such as read, write, and *trim* commands, we define new NVMe commands to enable users to configure the network connections. To support data recovery in RSSD, we slightly modify the NVMe command interpreter and add a state query engine into the SSD firmware for locating retained stale data. We reserve 64MB DRAM in the SSD controller for bloom filters and data compression buffer. We also use 4MB DRAM in the SSD controller for logging read and *trim* operations, respectively. We use the page-level address translation [25] mapping in RSSD. We implemented delta compression with the LZF algorithm [46] for its high performance. As we develop RSSD as a firmware solution, we can integrate new defense solutions at the manufacturing stage. Once the firmware is flushed into the SSD controller, the firmware can also be updated with authorization. We use an in-house server to work as the remote server, and get it connected with the OpenSSD board via the NVMe-oE (1GigE). The server has a 16-core Skylake based Intel CPU running at 3.6GHz with 64GB DRAM and 10TB HDD. We deploy the log-structured file system F2FS on the remote server to manage the transferred flash blocks.

7 EVALUATION

Our evaluation shows that (1) RSSD can retain the stale data for a much longer time than state-of-the-art approaches (§7.2); (2) It has minimal negative impact on storage performance and device lifetime (§7.3, §7.4, and §7.5); (3) It performs fast data recovery after attacks (§7.6); (4) It enables efficient post-attack analysis by building a trusted chain of I/O operations (§7.7).



(a) 50% capacity used in the SSD.



(b) 80% capacity used in the SSD.

Figure 7: Data retention time when running apps on RSSD.

7.1 Experimental Setup

To evaluate RSSD, we use various real-world storage traces, file system benchmarks, and data-intensive workloads. We list them in Table 3. They include (1) a set of storage traces collected on storage servers for different applications at Microsoft [51]; (2) a set of storage traces collected from computers at FIU [22]; (3) an open-source database engine, Shore-MT, running with benchmarks TPCC, TPCB, and TATP; (4) IOZone benchmark consisted of a variety of file operations [34]; (5) PostMark benchmark that emulates the I/O operations generated by a mail server. Before each experiment, we warm up the SSD by randomly running these workloads.

We compare RSSD with three baseline SSDs: (1) an SSD that retains stale data in time order (LocalSSD); (2) LocalSSD with delta compression (LocalSSD+Compression); (3) a regular SSD that does not intentionally retain data (Regular SSD).

7.2 Impact on Data Retention Time

We first evaluate the impact of RSSD on data retention time, with MSR and FIU storage traces. To evaluate the capability of retaining data before filling up the SSD, we prolong each trace to ten months by duplicating it a hundred times. Since MSR and FIU traces do not contain real data content, we use five as the default compression ratio, following the real workload characterization study in [88].

As expected, the data retention time is determined by both storage utilization and workload patterns. We show the data retention time of running different storage workloads under various capacity utilization (50% and 80%) in Figure 7. LocalSSD can retain stale data for 0.9-11.2 days and 0.12-7.4 days for 50% and 80% capacity utilization, respectively. With delta compression enabled, LocalSSD extends the data retention time by up to 6.4×. RSSD further extends the data retention time by 2.1–4.3×, compared to LocalSSD+Compression. As we decrease the compression ratio, we see the similar trend on the improvement of data retention time (not shown due to space). As shown in Figure 7, RSSD can retain data for up to 215 days by gradually transferring stale data to the remote server, such that the local SSD has more free space. Note that RSSD transfers only stale data to remote cloud/server and will be filled with valid data eventually.

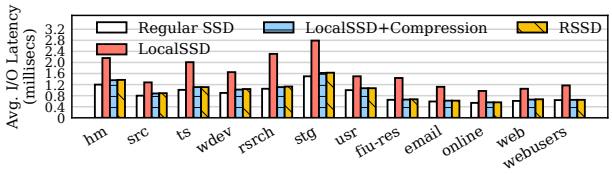


Figure 8: Storage performance of RSSD.

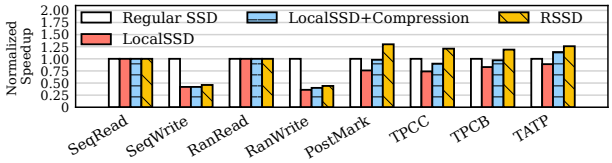


Figure 9: Impact on the local storage performance.

7.3 Impact on Storage Performance

We now evaluate the impact of RSSD on the storage performance. We set the SSD utilization as 80%. We show the results in Figure 8. Compared to a regular SSD that does not intentionally retain data, LocalSSD increases the I/O response time by 1.9×, due to the intensive GC operations. This overhead is reduced by LocalSSD+Compression, as data compression is used to compact the retained invalid pages and free more flash blocks. In comparison to LocalSSD+Compression, RSSD does not introduce much performance overhead (0.77% on average), when it gradually transfers packed stale data to the remote server. This indicates the NVMe-oE of RSSD does not affect regular storage operations. It helps free storage space for the local SSD and alleviate the GC overhead.

We also evaluate RSSD with IOZone, PostMark, and OLTP workloads that generate real data. To show the effectiveness of data compression, we use the regular SSD as the baseline. We use IOZone to generate sequential/random reads and writes. As shown in Figure 9, LocalSSD decreases the storage performance by 24.8%, compared to Regular SSD. For the workloads of sequential/random writes, LocalSSD performs even worse due to the significant GC overhead. LocalSSD+Compression outperforms LocalSSD by 1.13× on average, as the delta compression reduces the storage space occupied by the stale data. RSSD outperforms LocalSSD by 1.31× on average, as it further free storage space.

For sequential read/write and random read workloads, RSSD performs similarly to others, as there are no invalid pages produced. For PostMark, RSSD achieves 1.71× speedup, compared to LocalSSD. For OLTP workloads, RSSD offers 1.63× (8.5K TPS), 1.43× (38.0K TPS), 1.42× (135.7K TPS) more throughput than LocalSSD for TPCC, TPCB, and TATP, respectively. For these workloads, RSSD performs even better than the Regular SSD, this is because RSSD gradually utilizes the idle I/O cycles to transfer stale data to the remote storage, which frees more storage space.

7.4 Performance Impact of the Network

To evaluate the performance impact of the NVMe-oE network connected to the remote server, we test another two configurations. We replace the local server with (i) a similar server located in another university (Remote Server), and (ii) the Amazon S3 cloud storage (Remote Cloud), respectively. As shown in Figure 10, the

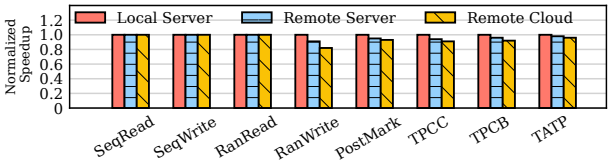


Figure 10: Impact of connected remote servers.

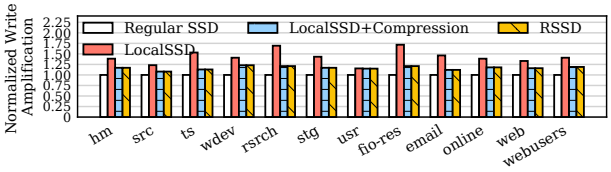


Figure 11: Impact on the SSD lifetime.

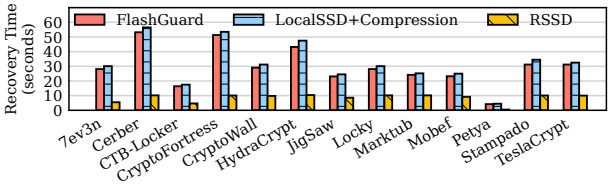


Figure 12: Data recovery time after ransomware attacks.

performance of RSSD slightly drops by 3.2% and 6.7% when connected with Remote Server and Remote Cloud, respectively. As the transfer speed becomes slower, the local SSD will retain more stale data temporarily, which will affect the local storage performance. If the SSD is disconnected to the remote cloud/server, RSSD will still compress and retain stale data until the device becomes full.

7.5 Impact on the SSD Lifetime

We use the ratio of flash write traffic to user-space write traffic as the write amplification metric for the SSD lifetime. Higher write amplification means a shorter lifetime. As shown in Figure 11, LocalSSD and LocalSSD+Compression increase the write amplification by 42.9% and 16.7% on average respectively, under 80% capacity utilization. This is mainly caused by the page migration of retained stale data at GC. Compared to LocalSSD+Compression, RSSD introduces little new write traffic when it transfers data to remote server. As we transfer retained stale data to remote cloud/servers more frequently, the write amplification will be reduced because this will reduce the GC frequency in the local SSD.

7.6 Recovery from Ransomware Attacks

To evaluate the data recovery, we compare it with the existing work FlashGuard [30], which retains only the invalid pages potentially encrypted by ransomware. We use 13 ransomware samples from VirusTotal and run them in a virtual machine with the local SSD mounted. As shown in Figure 12, LocalSSD+Compression needs 12.4% more time to recover the data encrypted by ransomware, due to the data decompression in the SSD. RSSD significantly reduces the recovery time (4.2× on average), when the packed stale data is available on the remote server, as it can take advantage of the powerful computing resource to speed up the data decryption and decompression, and parallelize the stale data lookups.

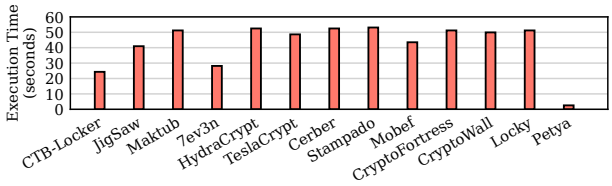


Figure 13: Post-attack analysis time of ransomware attacks.

Note that our proposed new attacks can easily bypass state-of-the-art approaches, such as FlashGuard [30], SSD-Insider [12], RBlocker [56], and others [6, 19, 40, 78, 80]. They invalidate most of these defense mechanisms developed based on learning existing ransomware behaviors [12, 56]. Existing data recovery approaches [12, 30, 80] cannot defend against these new attacks, as most of them can only retain the victim data for a limited time. None of them can retain the victim data caused by the *trim* command.

7.7 Effectiveness of Post-attack Analysis

RSSD enables post-attack analysis by logging all the storage operations. To evaluate the effectiveness of this function, we conduct the post-attack analysis after running 13 ransomware samples as used in §7.6. We present the performance of the post-attack analysis of each ransomware sample in Figure 13. We start the post-attack analysis when the packed stale data is available on the remote server. During the post-attack analysis, we build the evidence chain by listing all the storage operations in the victim SSD in time order, it includes the timestamp of issuing the storage operation, the operation type (read/write/trim), the logical page address, and the physical page address. With this evidence chain, we can replay the attack procedure. RSSD can finish the analysis in 2.6–52.5 seconds.

8 DISCUSSION

8.1 Cost-Effectiveness Analysis

To evaluate the cost effectiveness of RSSD, we compare it to the approach of increasing the local storage capacity by (1) packing more flash chips in the SSD or (2) using hybrid SSD/HDD. To ensure a fair comparison, we calculate the cost of cloud storage when the local storage is filled with valid data.

To implement (1), we plug more flash modules into the OpenSSD to increase its capacity to 2TB (Expanded SSD in Table 4). We assume flash memory is $0.2/GB, and remote cloud storage is $0.00081/GB per month [5, 9]. We use the network cost models of cloud storage to calculate the data transfer cost, which is free in both Amazon S3 [1] and Azure [50]. We rerun the storage traces (see Table 3) until the SSD is full. We compare RSSD to LocalSSD+Compression that retains compressed stale data in the local SSD until it fills up the SSD. RSSD reduces the storage cost by 85.6× on average (see Table 4), while providing the flexibility to extend storage capacity. As we conduct a conservative calculation with $0.0228 per 10K writes and $0.00182 per 10K reads, and retaining the data as long as the LocalSSD+Compression, RSSD still achieves 74.5× cost reduction on average. Note that RSSD introduces trivial overhead to the local storage performance (see §7.3).

To implement (2), we expand the local storage capacity by using hybrid SSD (1TB) and HDD (10TB) locally. For this approach, we

Table 4: Storage cost reduction of RSSD, in comparison with the approaches of expanding the storage capacity locally.

	Storage in Enterprise							Storage in University				
Scheme	hm	src	ts	wdev	rsrch	stg	usr	fiu-res	email	online	web	webusers
Expanded SSD	236.9×	49.5×	68.8×	34.4×	40.8×	68.1×	301.9×	54.7×	58.3×	45.4×	63.3×	41.4×
SSD+HDD	24.7×	4.9×	6.9×	3.4×	4.1×	6.8×	25.2×	5.5×	5.8×	4.5×	6.3×	4.5×

assume the HDD price is $0.02/GB [26]. RSSD reduces the storage cost by 8.6× on average (see SSD+HDD in Table 4).

RSSD suggests remote cloud/servers, as there is a fundamental drawback of adding SSDs or HDDs locally – it lacks the flexibility of expanding the storage capacity. In contrast, RSSD offers enhanced flexibility and high data reliability enabled in modern cloud platforms to expand the local storage with lower cost.

8.2 Design Choices and Future Work

RSSD Deployment. According to our study in §2.1, we observed that any industry can be targeted by ransomware attacks (see Table 2). Furthermore, the threat model does not change across different industries or deployments (e.g., servers or desktops). Therefore, developing RSSD underneath the standard block I/O interface makes it general and useful for any deployment scenarios.

Zero Data Loss Recovery. There are two common approaches today to fight ransomware: intrusion detection [37, 70, 87] and data backups [15, 19, 84, 90]. However, we realize that few of these solutions can fundamentally protect against ransomware. Although intrusion detection can successfully detect ransomware attacks, part of the user data could have been encrypted. And existing data backup solutions cannot ensure all the recent data updates have been securely retained, as discussed in §2.2. Therefore, we argue that zero data loss recovery could be the most promising approach to stop ransomware. No matter how ransomware evolves, if we can restore the victim data without paying the ransom, ransomware attacks would decrease and even stop. We develop RSSD to demonstrate the feasibility of achieving this goal with low hardware cost.

Post-Attack Analysis. As we develop secure storage systems to defend against ransomware, we realize few prior studies worked on post-attack analysis or storage forensics. We believe this will inevitably limit our understanding of new and emerging ransomware attacks. Therefore, we define the trusted post-attack analysis as an important feature of secure storage systems. However, developing this feature is not easy, as we have to guarantee the constructed evidence can be trusted. In RSSD, we make the post-attack analysis hardware isolated by coupling it with our design for the zero data loss recovery. Therefore, we enable this feature in RSSD with minimal additional cost (see §5.3.4).

In-Storage Detection. To further reduce the storage cost of RSSD, a potential approach is to detect the ransomware attacks as early as possible. Therefore, we would not need to conservatively retain all invalid flash pages for a long time. This will also help us achieve both improved storage efficiency and the capability of zero data loss recovery. Instead of relying on the software-based detection solutions that could be compromised, we can develop in-storage ransomware detection by running detection algorithms in the SSD. To enable SSDs to detect new and emerging ransomware attack patterns, we can enable learning-based algorithms by upgrading embedded processors to in-storage hardware accelerators [47]. We wish to extend RSSD in this direction as future work.

9 RELATED WORK

Ransomware Detection and Defense. Researchers have been investigating ransomware detection mechanisms [37, 38, 70]. For instance, they proposed to leverage machine learning to perform ransomware classification [16, 48, 72]. However, they cannot recover the damage that has been caused. Thus, ransomware still locks up some of data, forcing end users to pay a ransom. Data backup and recovery enables users to restore their data to their previous copies [18, 24, 60, 67]. They include log-structured file systems [66], journaling file systems [58], and cloud-based storage systems [24]. However, since ransomware can run with kernel privilege, those backup systems can be disabled or compromised. The versioning SSDs [30, 31, 80] can retain storage states. However, they suffer from limited storage capacity and cannot retain data removed by the *trim* command. RSSD proposes a new approach to extend the storage capacity in a secure and transparent manner.

Secure Storage Systems. System and architecture community have been working on flash-based storage [17, 28, 29, 59, 75], However, most focused on performance rather than security. Similarly, hardware researchers have focused on increasing storage capacity and performance [25, 29, 42], few of them considered security in their design. Although we have deployed flash-based storage devices on various computing platforms [32, 71], none of the released products claimed they can defend against ransomware attacks. We develop RSSD with a real SSD.

Post-Attack Analysis. To further strengthen storage security, it is essential to conduct vulnerability analysis after identifying malware attacks. However, adversaries can destroy the attack evidence and perform malicious modifications to the logs to hide their behaviors [14, 74, 81]. RSSD retains all the storage operations in the SSD, which can reconstruct the entire evidence chain, including anti-forensics operations. Most importantly, this evidence chain is trusted, as the logging procedure is conducted in a hardware-isolated manner, which cannot be manipulated by malware.

10 CONCLUSION

We present a secure SSD to defend against new ransomware attacks and enable post-attack analysis. RSSD utilizes NVMe over Ethernet to expand the storage capacity in a transparent and secure manner. It also enhances the security support for the trim command. Experimental results show that RSSD incurs negligible overhead.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers and our shepherd Haibo Chen for their helpful comments and feedback. We thank Yuqi Xue for his help with the study of recent ransomware attacks. We also thank the group members in the Systems Platform Research Group at UIUC for their insightful discussions on this work. This work was partially supported by the NSF grant CCF-1919044 and ARO Small Business Technology Transfer Program W911NF-20-C-0010.

REFERENCES

[1] 2021. Amazon S3 Pricing. https://aws.amazon.com/s3/pricing/.

[2] 20 Cybersecurity Statistics That Matter in 2019. 2019. https://www.appknox.com/blog/cybersecurity-statistics-2019.

[3] 2021 Ransomware Attack List and Alerts. 2021. https://cloudian.com/ransomware-attack-list-and-alerts/.

[4] 3 Common Ways Ransomware Attacks Happen and How to Prevent Them. 2020. https://www.provendatarecovery.com/blog/common-ransomware-attacks/.

[5] Amazon S3 Pricing. 2020. https://aws.amazon.com/s3/pricing/.

[6] Nicolo Andronio, Stefano Zanero, and Federico Maggi. 2015. HelDroid: Dissecting and Detecting Mobile Ransomware. In *Proceedings of the International Symposium on Research in Attacks, Intrusion and Detection (RAID'15)*. Kyoto, Japan.

[7] Andy Patrizio. 2020. Western Digital Rolls Out NVMe-over-Fabric SSDs for Legacy Storage Migration. https://www.networkworld.com/article/3564607/western-digital-rolls-out-nvme-over-fabric-ssds-for-legacy-storage\-migration.html.

[8] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. 2007. Provable Data Possession at Untrusted Stores. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS'07)*. Alexandria, VA.

[9] Azure Storage Overview Pricing. 2020. https://azure.microsoft.com/en-us/pricing/details/storage/.

[10] Azure Storage Redundancy. 2021. https://docs.microsoft.com/en-us/azure/storage/common/storage-redundancy.

[11] Eitan Bachmat and Jiri Schindler. 2002. Analysis of Methods for Scheduling Low Priority Disk Drive Tasks. In *Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'02)* (Marina Del Rey, California).

[12] SungHa Baek, Youngdon Jung, Aziz Mohaisen, Sungjin Lee, and DaeHun Nyang. 2018. SSD-Insider: Internal Defense of Solid-State Drive against Ransomware with Perfect Data Recovery. In *Proceedings of 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS'18)*. Vienna, Austria.

[13] Kevin R. B. Butler, Stephen McLaughlin, and Patrick D. McDaniel. 2008. Rootkit-Resistant Disks. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*. Alexandria, VA.

[14] Scott A. Carr and Mathias Payer. 2017. DataShield: Configurable Data Confidentiality and Integrity. In *ACM Asia Conference on Computer and Communications Security (ASIACCS'17)*. Abu Dhabi, UAE.

[15] Rich Castagna. 2014. What is cloud backup and how does it work? https://searchdatabackup.techtarget.com/definition/cloud-backup.

[16] Li Chen, Chih-Yuan Yang, Anindya Paul, and Ravi Sahita. 2019. Towards Resilient Machine Learning for Ransomware Detection. In *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19)*. Anchorage, Alaska.

[17] Joel Coburn, Trevor Bunker, Meir Schwarz, Rajesh Gupta, and Steven Swanson. 2013. From ARIES to MARS: Transaction Support for Next-generation, Solid-state Drives. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP'13)*.

[18] IBM Comparing. 2002. Tivoli Storage Manager and VERITAS NetBackup in Real-World Environments. *A summary by IBM of the whitepaper and benchmark written by Progressive Strategies* (2002).

[19] Andrea Continella, Alessandro Guagneli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. 2016. ShieldFS: A Self-healing, Ransomware-aware Filesystem. In *Proc. the 32nd Annual Conference on Computer Security Applications (ACSAC'16)*. Los Angeles, CA.

[20] Data Protection in Amazon S3. 2021. https://docs.aws.amazon.com/AmazonS3/latest/userguide/DataDurability.html.

[21] Encryption and Security Development in Solid State Storage Devices. 2020. https://www.delkin.com/blog/encryption-and-security-development-in-solid-state-storage-devices-ssd/.

[22] FIU Traces. 2010. http://iotta.snia.org/traces/390.

[23] Gartner Sees Enterprise SSD-HDD Revenue Crossover in 2017. 2017. https://www.theregister.co.uk/2016/01/07/gartner_enterprise_ssd_hdd_revenue_crossover_in_2017/.

[24] James Gross. 2013. Cloud based storage: A brief look at dropbox. *Chronicles* 30, 4 (2013).

[25] Aayush Gupta, Youngjae Kim, and Bhuvan Urgaonkar. 2009. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating System (ASPLOS'09)*.

[26] Here is the cheapest hard drive per TB right now. 2020. https://www.techradar.com/news/heres-the-cheapest-hard-drive-per-tb-right-now.

[27] How Bad is the Problems? Cyberattacks rank 5th in the Top 10 list of most-likely global risks. 2021. https://cloudian.com/lp/lock-ransomware-out-keep-data-safe-ent/.

[28] Jian Huang, Anirudh Badam, Laura Caulfield, Suman Nath, Sudipta Sengupta, Bikash Sharma, and Moinuddin K. Qureshi. 2017. FlashBlox: Achieving Performance Isolation and Uniform Lifetime for Virtualized SSDs. In *Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST'17)*. Santa Clara, CA.

[29] Jian Huang, Anirudh Badam, Moinuddin K. Qureshi, and Karsten Schwan. 2015. Unified Address Translation for Memory-Mapped SSDs with FlashMap. In *Proceedings of the 42nd International Symposium on Computer Architecture (ISCA'15)*. Portland, OR.

[30] Jian Huang, Jun Xu, Xinyu Xing, Peng Liu, and Moinuddin K. Qureshi. 2017. FlashGuard: Leveraging Intrinsic Flash Properties to Defend Against Encryption Ransomware. In *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS'17)*. Dallas, TX.

[31] Ping Huang, Ke Zhou, Hua Wang, and Chun Hua Li. 2012. BVSSD: Build Built-in Versioning Flash-Based Solid State Drives. In *Proceedings of 5th Annual International Systems and Storage Conference (SYSTOR'12)*. Haifa, Israel.

[32] Intel Corporation. 2018. Intel® Optane™ SSD DC P4801X Series. *Technical Report* (2018).

[33] Intel SGX Explained. 2016. https://eprint.iacr.org/2016/086.pdf.

[34] IOzone Lab. 2016. http://www.iozone.org/.

[35] Jeffrey Burt. 2019. Bringing NVMe Over Fabrics Into the Storage Big Tent, https://www.nextplatform.com/2019/02/26/bringing-nvme-over-fabrics-into-the-storage-big-tent/.

[36] John Edwards. 2019. NVMe over Fabrics Creates Data-Center Storage Disruption, https://www.networkworld.com/article/3394296/nvme-over-fabrics-creates-data-center-storage-disruption.html.

[37] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. 2016. UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 757–772.

[38] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. 2015. Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks. In *Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*. Milan, IT.

[39] Scott Kipp. 2015. Will SSD replace HDD. *Technical Report* (2015).

[40] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele. 2017. PayBreak: Defense Against Cryptographic Ransomware. In *Proc. the 2017 ACM on Asia Conference on Computer and Communications Security*. Abu Dhabi, United Arab Emirates.

[41] Gunjae Koo, Kiran Kumar Matam, Te I, Hema Venkata Krishna Giri Nara, Jing Li, Hung-Wei Tseng, Steven Swanson, and Murali Annavaram. 2017. Summarizer: Trading Bandwidth with Computing Near Storage. In *50th Annual IEEE/ACM International Symposium on Microarchitecture*. Boston, MA.

[42] Gunjae Koo, Kiran Kumar Matam, HV Narra, Jing Li, Hung-Wei Tseng, Steven Swanson, Murali Annavaram, et al. 2017. Summarizer: Trading Communication with Computing Near Storage. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'17)*. Boston, MA.

[43] Swetha Krishnan, Giridhar Ravipati, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Barton P. Miller. 2007. The Effects of Metadata Corruption on NFS. In *Proceedings of the 2007 ACM Workshop on Storage Security and Survivability (StorageSS'07)*. Alexandria, VA.

[44] Changman Lee, Dongbo Sim, Joo-Young Hwang, and Sangyeun Cho. 2015. F2FS: A New File System for Flash Storage. In *Proc. FAST'15*. Santa Clara, CA.

[45] J. Lee, Y. Kim, G. M. Shipman, S. Oral, and J. Kim. 2013. Preemptible I/O Scheduling of Garbage Collection for Solid State Drives. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2013).

[46] LibLZF. [n. d.]. http://oldhome.schmorp.de/marc/liblzf.html.

[47] Vikram Sharma Mailthoday, Zaid Qureshi, Weixin Liang, Ziyan Feng, Simon Garcia de Gonzalo, Youjie Li, Hubertus Franke, Jinjun Xiong, Jian Huang, and Wen mei Hwu. 2019. DeepStore: In-Storage Acceleration for Intelligent Queries. In *Proceedings of the 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO'19)*. Columbus, OH.

[48] Sumith Maniath, Aravind Ashok, Prabaharan Poornachandran, Sujadevi VG, Prem Sankar A U, and Srinath Jan. 2017. Deep Learning LSTM based Ransomware Detection. In *Proceedings of the 2017 Recent Development in Control, Automation and Power Engineering (RDCAPE'17)*. Noida, India.

[49] Ningfang Mi, Alma Riska, Qi Zhang, Evgenia Smirni, and Erik Riedel. 2009. Efficient Management of Idleness in Storage Systems. *ACM Trans. Storage* 5, 2 (2009).

[50] Microsoft Azure Bandwidth Pricing Details. 2021. https://azure.microsoft.com/en-us/pricing/details/bandwidth/.

[51] MSR Cambridge Traces. 2010. http://iotta.snia.org/traces/388.

[52] Codrut Neagu. 2018. What is SSD TRIM, why is it useful, and how to check whether it is turned on. https://www.digitalcitizen.life/simple-questions-what-trim-ssds-why-it-useful.

[53] NetApp. 1997. http://www.shub-internet.org/brad/FreeBSD/postmark.html.

[54] NVM Express. [n. d.]. NVMe over Fabrics. https://nvmexpress.org/wp-content/uploads/NVMe_Over_Fabrics.pdf

[55] NVMe over Fabrics. 2016. https://nvmexpress.org/wp-content/uploads/NVMe_Over_Fabrics.pdf.

[56] Jisung Park, Youngdon Jung, Jonghoon Won, Minji Kang, Sungjin Lee, and Jihong Kim. 2019. RansomBlocker: a Low-Overhead Ransomware-Proof SSD. In *Proceedings of the 57th Design Automation Conference (DAC'19)*. Las Vegas, USA.

[57] Power Loss Protection. 2020. https://www.atpinc.com/blog/how-industrial-SSDs-handles-power-loss).

[58] Vijayan Prabhakaran, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. 2005. Analysis and Evolution of Journaling File Systems.. In *USENIX Annual Technical Conference (USENIX ATC'05)*. 105–120.

[59] Vijayan Prabhakaran, Thomas L. Rodeheffer, and Lidong Zhou. 2008. Transactional Flash. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation (OSDI'08)*.

[60] Curtis Preston. 2007. *Backup & recovery: inexpensive backup solutions for open systems.* O'Reilly Media, Inc.

[61] Ransomware Attack Every 14 Seconds. 2019. https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-20-billion-usd-by-2021/.

[62] Ransomware Costs. 2020. https://purplesec.us/resources/cyber-security-statistics/ransomware/#:~:text=The%20Cost%20Of%20Ransomware%20Attacks,2019%20%E2%80%93%20%2411.5%20billion.

[63] Ransomware Samples. 2021. https://github.com/x7z9h3/x7z9h3.

[64] Ransomware Statistics, Trends and Facts for 2021 and Beyond. 2021. https://www.cloudwards.net/ransomware-statistics/.

[65] Joel Reardon, David Basin, and Srdjan Capkun. 2013. SoK: Secure Data Deletion. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy (Oakland'13)*.

[66] Mendel Rosenblum and John K Ousterhout. 1992. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems (TOCS)* 10, 1 (1992), 26–52.

[67] Mark E Russinovich, David A Solomon, and Jim Allchin. 2005. *Microsoft Windows Internals: Microsoft Windows Server 2003, Windows XP, and Windows 2000*. Vol. 4. Microsoft Press Redmond.

[68] Samsung. 2020. Samsung SSD, https://www.samsung.com/semiconductor/minisite/ssd/product/data-center/983dct/.

[69] Samsung. 2021. Samsung NVMe SSD 983 DCT: Enhanced for Speed and Reliability. *White Paper* (2021).

[70] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin RB Butler. 2016. Cryptolock (and drop it): stopping ransomware attacks on user data. In *Proceedings of the 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS'16)*.

[71] Samsung Semiconductors. 2018. Ultra-Low Latency with Samsung Z-NAND SSD. *Technical Report* (2018).

[72] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu. 2016. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection. *ArXiv e-prints* (Sept. 2016). arXiv:1609.03020 [cs.CR]

[73] Shore-MT. 2014. https://sites.google.com/view/shore-mt/.

[74] Gopalan Sivathanu, Charles P. Wright, and Erez Zadok. 2004. Enhancing File System Integrity Through Checksums. *Technical Report* (2004).

[75] Sriram Subramanian, Swaminathan Sundararaman, Nisha Talagala, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2014. Snapshots in a Flash with ioSnap. In *Proceedings of the 9th ACM European Conference on Computer Systems (EuroSys'14)*.

[76] Texas Instruments. 2015. DP83867E/IS/CS/IR/CR RGZ Power Consumption Data. *White Paper* (2015).

[77] The OpenSSD Project. 2019. http://www.openssd-project.org/wiki/The_OpenSSD_Project.

[78] Aragorn Tseng, YunChun Chen, YiHsiang Kao, and TsungNan Lin. 2019. Deep Learning for Ransomware Detection. *Technical Report* (2019).

[79] VirusTotal - Free Online Virus, Malware and URL Scanner. 2016. https://www.virustotal.com/.

[80] Xiaohao Wang, You Zhou, Chance C. Coats, and Jian Huang. 2019. Project Almanac: A Time-Traveling Solid-State Drive. In *Proceedings of the 14th European Conference on Computer Systems (EuroSys'19)*. Dresden, Germany.

[81] Yongzhi Wang and Yulong Shen. 2016. RIA-An Audition-based Method to Protect the Runtime Integrity of MapReduce Applications. In *23rd ACM Conference on Computer and Communications Security*. Vienna, Austria.

[82] WannaCry Ransomware Attack. 2017. https://en.wikipedia.org/wiki/WannaCry_ransomware_attack.

[83] Michael Yung Chung Wei, Laura M Grupp, Frederick E Spada, and Steven Swanson. 2011. Reliably Erasing Data from Flash-Based Solid State Drives.. In *Proceedings of 9th USENIX Conference on File and Storage Technologies (FAST'11)*.

[84] Wikipedia. 2014. Journaling File System. https://en.wikipedia.org/wiki/Journaling_file_system.

[85] Wikipedia Ransomware. 2019. https://en.wikipedia.org/wiki/https://www.appknox.com/blog/cybersecurity-statistics-2019Ransomware.

[86] Guanying Wu and Xubin He. 2012. Delta-FTL: Improving SSD Lifetime via Exploiting Content Locality. In *Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys'12)*.

[87] Dongpeng Xu, Jiang Ming, and Dinghao Wu. 2017. Cryptographic Function Detection in Obfuscated Binaries via Bit-precise Symbolic Loop Mapping. In *Proceedings of the 38th IEEE Symposium on Security and Privacy (Oakland'17)*. San Jose, CA.

[88] Qing Yang and Jin Ren. 2011. I-CASH: Intelligently Coupled Array of SSD and HDD. In *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture (HPCA'11)*.

[89] ZDNet. 2021. This is how long hackers will hide in your network before deploying ransomware or being spotted, https://www.zdnet.com/article/this-is-how-long-hackers-will-spend-in-your-network-before-deploying-ransomware-or-being-spotted/.

[90] Lingchen Zhang, Sachin Shetty, Peng Liu, and Jiwu Jing. 2014. RootkitDet: Practical End-to-End Defense Against Kernel Rootkits in a Cloud Environment. In *Proceedings of the 19th European Symposium on research in Computer Security (ESORICS'14)*. Wroclaw, Poland.

[91] Leah Zhang-Kennedy, Hala Assal, Jessica Rocheleau, Reham Mohamed, Khadija Baig, and Sonia Chiasson. 2018. The aftermath of a crypto-ransomware attack at a large academic institution. In *27th USENIX Security Symposium (Security'18)*. Baltimore, MD.