IPOD2: An Irrecoverable and Verifiable Deletion Scheme for Outsourced Data

Xiaolei Zhang[1,3,4], Zhaoyu Chen[2,3,4], Xin Zhang[2,3,4], Qingni Shen[2,3,4,*] and Zhonghai Wu[1,2,3,4,*]

[1]*School of Computer Science, Peking University, Beijing 100871, China*
[2]*School of Software and Microelectronics, Peking University, Beijing 102600, China*
[3]*National Engineering Research Center for Software Engineering, Peking University, Beijing 100871, China*
[4]*PKU-OCTA Laboratory for Blockchain and Privacy Computing, Peking University, Beijing 100871, China*
[*]*Corresponding author: qingnishen@ss.pku.edu.cn; wuzh@pku.edu.cn*

To alleviate the burden of data storage and management, there is a growing trend of outsourcing data to the cloud that enables users to remotely manage their data flexibly. However, this shift also raises concerns regarding outsourced data deletion, as users lose physical control over their outsourced data and are unable to verify its proper eradication. To address this issue, cloud service providers are required to provide a scheme that guarantees the effective deletion of outsourced data. Existing schemes, including key management-based and overwriting-based schemes, fail to ensure both the irrecoverability of deleted data and the verifiability of the deletion process. In this paper, we propose IPOD2, an irrecoverable and verifiable deletion scheme for outsourced data. Specifically, IPOD2 utilizes the overwriting-based deletion method to implement outsourced data deletion and extends the Integrity Measurement Architecture (IMA) to measure the operations in the deletion process. The measurement results are protected by the Trusted Platform Module (TPM) and verifiable for users. To demonstrate the viability of IPOD2, we implement a prototype of IPOD2 on the Linux kernel 5.4.120. Experimental results show that, compared with the three existing schemes, IPOD2 has the minimum overhead in both deletion and verification processes.

Keywords: Outsourced Data Deletion; Trusted Computing; Cloud Storage; Integrity Measurement Architecture

1. INTRODUCTION

To alleviate the burden of data storage and management, there is a growing trend of outsourcing data to the cloud [1–3] that enables users to remotely manage their data flexibly [4, 5]. Despite the advantages offered by cloud storage, outsourced data encounters significant data security challenges, leading to frequent instances of data leakage [6–8]. In particular, due to the multi-tenancy feature of cloud storage, outsourced data that is not promptly deleted according to users' requirements can easily be inadvertently exposed to unauthorized users. This results in the leakage of user privacy [9, 10]. Recognizing these issues, both the General Data Protection Regulation (GDPR) [11] and the China Personal Information Protection Law (PIPL) [12] necessitate data controllers (e.g., cloud service providers) to provide effective deletion methods for outsourced data.

The root cause preventing users from confirming the deletion of outsourced data is the lack of transparency in the deletion process. Once data is outsourced, users lose physical control over the data because they cannot access the infrastructure of cloud storage [9, 13–15]. This results in the deletion of outsourced data becoming a black-box operation, making it impossible for users to confirm whether the data has been deleted as required [14–16]. To mitigate these concerns, the cloud service provider (CSP) should provide an effective outsourced data deletion scheme that fulfills two essential properties:

- **Irrecoverability:** The deleted outsourced data should become irrecoverable immediately, and no one can obtain any information about the deleted data.
- **Verifiability:** CSP should provide detailed and trustworthy deletion proof of the deletion process,

TABLE 1. The comparison of outsourced data deletion schemes. IPOD2 achieves irrecoverability and verifiability well at the same time.

Scheme	**Deletion Method**	**Security Guarantee**	**Irrecoverability**	**Verifiability**
[17]	delete key	CKM	◐	○
[10]	delete key	CKM	◐	○
[18]	delete key	CKM	◐	○
[9]	delete key	DKM	◐	○
[13]	delete key	DKM	◐	○
[19]	delete key	CSP	◐	○
[14]	delete key	CSP	◐	◐
[15]	re-encrypt data	CSP	◐	◐
[20]	overwrite data	CSP	●	◐
[16]	overwrite data	CSP	●	◐
IPOD2	**overwrite data**	**TH**	●	●

CKM: Centralized Key Manager. DKM: Decentralized Key Manager. CSP: Cloud Service Provider. TH: Trusted Hardware.
○ Not satisfy the property. ◐ Partially satisfy the property. ● Fully satisfy the property.

showcasing the successful deletion of the data as per the users' request.

Previous approaches for deleting outsourced data can be divided into two classes based on deletion methods: key management-based schemes and overwriting-based schemes. Specifically, key management-based schemes [9, 10, 13–15, 17–19] encrypt the data before outsourcing and render the encrypted outsourced data irretrievable by deleting corresponding decryption keys. Although these schemes have demonstrated high efficiency in deleting data, they face the following three security concerns. First, as the ciphertext of the deleted data remains in the cloud, adversaries who obtain or recover the decryption key (e.g., forensic forensics or offline dictionary attacks) could recover the deleted data [9, 16, 20]. Second, as we summarized in Table 1, all these key management-based schemes entail the need for a trusted entity (e.g., a trusted third party or the cloud service provider itself) to manage cryptographic keys, thereby increasing the management overhead. Last, most key management-based schemes do not consider the verification of outsourced data deletion. Although Mo et al. [14] and Xue et al. [15] provide simple proofs that include deletion results, their simple proofs do not include detailed information on the deletion process, and the credibility of proofs lacks assurance from trusted hardware (e.g., Trusted Platform Module). As a result, CSP cannot furnish users with adequate details related to the deletion process and may face doubts from users regarding potential tampering or forgery of the proofs.

Overwriting-based deletion schemes [16, 20] aim to ensure data irrecoverability by overwriting the content of original data with a meaningless data pattern. The data pattern used for overwriting is typically a user-defined character sequence. After deletion, the user can verify the deletion result by checking whether the content of overwritten data is consistent with his predefined character sequence. These overwriting-based schemes leave minimal or even no information about the deleted data [21]. However, these deletion proofs also only contain the deletion result without the details of the deletion process and lack credibility assurance. Besides, due to the need to transfer the content of the overwritten data for each verification, the cost of verification increases with the data size.

In this paper, we propose IPOD2[1], a new data deletion scheme that provides detailed and trustworthy proof of the overwriting-based deletion process for verification. Specifically, to ensure the irrecoverability of deleted data, IPOD2 utilizes the overwriting-based deletion method to implement outsourced data deletion. To provide detailed and trustworthy proof of the deletion process, IPOD2 extends the Integrity Measurement Architecture (IMA) to measure the operations in the deletion process and protects the measurement results with the Trusted Platform Module (TPM). The measurement results are used to prove the deletion process of outsourced data. This way, IPOD2 can achieve irrecoverable deletion of outsourced data while providing detailed and trustworthy deletion proofs.

To construct IPOD2, there are four challenges to overcome.

First, since IMA can only identify system calls in the kernel space rather than operations in the user space, to enable IMA to identify and measure operations during the deletion process, it is necessary to map the operations to corresponding system calls. To achieve this, we conduct a comprehensive analysis of three overwriting-based deletion programs [22–24] that are commonly used in Linux to extract the system call sequence of the overwriting-based deletion process and establish a mapping from operations to system calls (see Section 5.2).

Second, IMA is event-driven, which means that IMA responds to specific events (measurement events), e.g., file access, and performs measurements for target files (measurement objects) associated with these events. So we need to specify the measurement events and measurement objects in IPOD2. Based on the mapping between operations and system calls, we design deletion hooks within related system calls to monitor executions of system calls. Additionally, we construct a Deletion Finite State Machine (D-FSM) to identify operations according to executions of system calls. The operations identified by D-FSM are the measurement objects in IPOD2. D-FSM generates corresponding measurement events for these operations to trigger IMA to measure

[1]IPOD2 refers to **I**MA **P**roof-based **O**utsourced **D**ata **D**eletion.

them. To further reduce redundant measurements, we set a specific attribute for outsourced data and IPOD2 only measures the operations on the data with this attribute (see Section 5.3).

Third, IMA does not measure the same object repeatedly and stores the measurement results of all objects together. However, the deletion process in IPOD2 includes repeated operations (e.g., multiple overwriting), and the measurement results generated by IMA are used as deletion proofs for users to verify. Directly applying IMA to measure operations in the deletion process poses two issues: 1) the inability to repeatedly measure the same object leads to the loss of measurement results for repeated operations in the deletion process, and 2) storing all measurement results together introduces significant overhead and potential privacy concerns during verification. To address these issues, we modify the measurement strategy of IMA to accommodate the requirements of IPOD2. For the measurement objects (i.e., operations in the deletion process) in IPOD2, we reset the flag used by IMA to mark the repeated measurement objects, enabling IMA to measure repeated operations in the deletion process. Based on the identifier of files within operations, we introduce a Splitter to split the measurement results of operations associated with different data. Besides, we designed a file-based Platform Configuration Register (fPCR) module to provide independent and TPM-based protection for measurement results (see Section 5.4).

Last, after deletion, the user should be able to verify the deletion of outsourced data based on the measurement results. To achieve this, in IPOD2, the user verifies measurement results based on the remote attestation technology. During the verification, if measurement results have not been tampered with or forged by anyone, we consider the measurement results to be trustworthy. The user initially validates whether the received measurement results are trustworthy. If the validation result is true, the user proceeds to verify whether the outsourced data has been deleted as required according to the content of the trustworthy measurement results (see Section 5.5).

We implement a prototype of IPOD2 on the Linux kernel 5.4.120, requiring minimal modifications with only 1528 lines of additional code. To demonstrate the viability and practicality of IPOD2, we perform a comprehensive evaluation from three aspects. First, to evaluate the impact of IPOD2 on the kernel, we separately run LMbench3 [25] and invoke each of the system calls implanted with deletion hooks 10 million times. Experimental results show that the overall performance of the kernel with IPOD2 is close to that of the original kernel and the additional overhead of these system calls is less than 15%. Second, we evaluate the deletion overhead and verification overhead of IPOD2. Specifically, we calculate the measurement overhead by comparing the time cost of the deletion process in the kernel with IPOD2 to the deletion process in the original kernel. According to experimental results, with the increase in file size, the measurement overhead slowly increases, but its influence on the overhead of the overall deletion process diminishes. To evaluate the verification overhead in IPOD2, we test the time cost for proof generation, communication, and proof verification during the verification process. The experimental results show that the entire overhead of the verification process does not exceed 0.5 seconds, and the overhead of the user to verify deletion proof is less than 14 milliseconds. Last, we compare the deletion and verification overheads of IPOD2 with three existing schemes, the experimental results show that IPOD2 has the minimum overhead in both the deletion and verification processes (see Section 6).

Summary of Contributions: The main contributions of this paper are as follows:

- We propose an irrecoverable and verifiable deletion scheme for outsourced data, called IPOD2, which provides detailed and trustworthy proof of the overwriting-based deletion process for verification. To the best of our knowledge, this is the first deletion scheme for outsourced data that provides detailed and trustworthy proof of the deletion process.

- To construct IPOD2, first, we map operations in the deletion process to system calls. Second, based on the mapping between the operations and system calls, we introduce deletion hooks and D-FSM to identify operations in the deletion process and trigger IMA to measure them. Third, we modify the measurement strategy of IMA to satisfy the requirements of measuring the deletion process. We further design fPCR module to provide independent and TPM-based protection for the measurement rustles of different data. Last, the verification process in IPOD2 is derived from the remote attestation, allowing the user to verify whether the data has been deleted as required based on the measurement results.

- To demonstrate the viability and practicality of IPOD2, we implement a prototype of IPOD2 on Linux 5.4.120 and conduct a comprehensive evaluation of it. Experimental results show that IPOD2 only adds 1528 lines of code to the kernel and introduces minimal additional overhead. Compared with the three existing schemes, IPOD2 has the minimum overhead in both deletion and verification processes.

The rest of this paper is organized as follows. In Section 2, we survey and discuss related works. In Section 3, we overview the background information. Next, in Section 4, we formalize the system model, threat model, and design goals of IPOD2. Following that, Section 5 presents an overview of IPOD2, followed by its design details. The implementation and evaluation are provided in Section 6. In Section 7, we discuss some limitations of IPOD2. Last, we conclude this paper in Section 8.

2. RELATED WORK

According to deletion methods, existing schemes can be classified into two classes as follows.

Key Management-based Schemes: In general, key management-based schemes work by encrypting data before outsourcing, and later deleting the data by destroying the corresponding key. Existing key management-based schemes can be divided into two classes based on key management methods, i.e., three-party-based schemes and two-party-based schemes.

In three-party-based schemes, researchers aim to design a system that works atop existing cloud storage services. Therefore, these schemes typically introduce a trusted third party to manage keys, while CSP is only responsible for storing data (ciphertext). Perlman et al. [17] first proposed a trusted centralized key manager (called Ephemerizer) to maintain the keys. However, the scheme does not consider fine-grained management. Tang et al. [10] then proposed a policy-based outsourced data deletion scheme, which associates data with policy to achieve fine-grained management, and multiple key managers based on Shamir's(M, N) threshold secret sharing [26] are used to avoid single point of failure. Tang et al. [18] constructed a Timed-Ephemerizer based on Ephemerizer [17] with complete security proofs, and it utilizes the timed-release encryption to achieve the data only be released after a predefined disclosure time. Considering the trust and security concerns of a centralized key manager, Geambasu et al. [9] proposed a distributed key manager based on a Distributed Hash Table (DHT), but attacks in DHT may affect the scheme's security. Castelluccia et al. [13] proposed the EphPub that builds on the Domain Name System (DNS) and its caching mechanism, but this scheme does not easy to achieve flexible control of the data's whole life cycle.

In two-party-based schemes, researchers consider the increased attack surface introduced by the third party and the users' suspicion regarding the trustworthiness of the third party. They proposed schemes involving only users and CSP. Mo et al. [14,19] proposed a fine-grained outsourced data deletion scheme without third-party key managers, which use the key modulation function and modulator adjustment algorithms to manage keys. Xue et al. [15] utilized the attribute revocation in KP-ABE [27] to achieve outsourced data deletion, user re-encrypt the outsourced data to make it unrecoverable and verify the re-encrypt result generated by CSP to verify the deletion result.

Key management-based schemes keep the ciphertext of the data in the cloud after deleting keys. Therefore, adversaries who obtain the keys (e.g., forensic forensics or offline dictionary attacks) could recover the deleted data. IPOD2 utilizes an overwriting-based method to delete data, which fundamentally addresses this issue. Besides, since data deletion in IPOD2 does not require key management, this effectively reduces the system's management overhead. Last, most existing key management-based schemes do not consider deletion verification. On the contrary, IPOD2 provides details of data deletion processes for users to verify.

Overwriting-based Schemes: Overwriting-based schemes [16, 20] aim to ensure data irrecoverability by overwriting the content of outsourced data with a user-defined, meaningless data pattern. PoSE-s proposed in [20] deletes outsourced data by using a special pattern generated by the users to overwrite the physical medium, and users verify the deletion result by comparing the overwritten result with the pattern. However, this scheme is designed for embedded devices with limited storage, which is not suitable for cloud storage. Zhang et al. [16] leveraged similar methods and proposed PTAD for cloud storage. PTAD references the idea of Proof of Data Possession(PDP) [28], which verifies the deletion result by checking the correctness and integrity of the overwritten data.

However, the deletion results provided in [16, 20] do not include details of deletion processes and lack credibility assurance. To address these issues, IPOD2 provides users with the deletion proof that includes detailed information about operations during the deletion process, further signed and protected by TPM. Besides, in [16], the necessity to transmit the content of overwritten data for verification results in increased verification overhead as the data size grows. In contrast, the deletion proof in IPOD2 records information about operations during the deletion process, independent of data size. Therefore, the verification overhead in IPOD2 is fixed and small irrespective of data size. We thoroughly evaluate the verification overhead of IPOD2 in Section 6.2.

Most existing schemes [9, 10, 14–16] focus on the irreversibility of deleted data, overlooking the verifiability of the deletion process. These schemes aim to resist an adversary (data thief) who may attempt to recover deleted data after deletion. Some schemes [14–16, 19, 20] also consider the verifiability of the deletion process. However, the provided deletion proof lacks both details of the deletion process and credibility assurance. Such deletion proof not only fails to convince users that outsourced data has been deleted as required but also prompts malicious users [16] to doubt the credibility of the proof. These malicious users allege that CSP has failed to delete data as required and seek compensation.

In IPOD2, we simultaneously consider the irrecoverability of deleted data and the verifiability of deletion processes under the aforementioned two types of attackers. The overwriting-based deletion method prevents

TABLE 2. Overwriting-based data deletion methods.

Type	Name	Overwrite Rounds	Pattern used to overwrite
Research	Gutmann method [30]	1 - 35	*random character*
	Pfitzner method [31]	7 or 33	*random character*
	Schneier method [32]	7	1. *number* 0 2. *number* 1 3-7. *random character*
Standard	NAVSO P-5239-26 [33]	3	1. *character* 2. *character's compliment* 3. *random character*
	DoD 5220.22-M [34] CSEC ITSG-06 [35] AFSSI-5020 [36] HMG IS5(Enhanced) [37]	3	1. *number* 0 2. *number* 1 3. *random character*
	HMG IS5(Baseline) [37]	2	1. *number* 0 2. *random character*
	NIST SP 800-88 [21]	1	1. *fixed pattern*(*e.g.*, 0)
	NZSIT 402 [38]	1	1. *random letter*

the data thief from recovering deleted data, while the detailed and trustworthy deletion proof enables users to verify the deletion process and leaves no room for malicious users to exploit.

3. BACKGROUND

3.1. Overwriting

Overwriting refers to writing new data to the same location on a storage device where previous data resides, the data used to write typically is random or meaningless content. Unlike unlinking, which only removes the pointer of data from the file system's directories, overwriting erases the original data, making the deletion more effective. Meanwhile, compared to physical destruction that completely destroys the physical medium containing the data, overwritten medium can still be reused, making the cost of deletion lower. Therefore, in file systems, overwriting is often used as a means of effectively deleting sensitive or confidential data. The deletion process usually involves multiple overwriting to ensure the irreversibility of the deleted data.

Reardon et al. [29] survey and discuss the overwriting-based deletion approaches based on the layers and interfaces involved in accessing a physical medium, where the Gutmann method [30], Pfitzner method [31], and Schneier method [32] are widely employed to eliminate data on the physical medium. In industry, overwriting-based deletion methods are also highly favored. Table 2 illustrates the details of some representative overwriting-based deletion methods, where the *Rounds* refers to the number of overwriting included in the deletion process, and the *Pattern* refers to the data patterns used to overwrite. As stated in NIST Special Publication 800-88 [21], for storage devices containing magnetic media, a single overwrite pass with a fixed pattern such as binary zero typically hinders recovery of data even if state-of-the-art laboratory techniques are applied to attempt to retrieve the data. In IPOD2, we utilize the overwriting-based method to delete data.

3.2. Integrity Measurement Architecture

We now provide the necessary background on the Integrity Measurement Architecture (IMA) and its dependence (i.e., TPM and remote attestation) on which IPOD2 builds.

TPM: From the perspective of Trusted Computing [39], trust can be transferred step by step. The Chain of Trust is constructed from a root of trust and extended with the entire boot time, including BIOS, GRUB, and finally the OS kernel. TPM is a hardware-based root of trust for a platform. Any component in the chain of trust is measured into TPM before loading, and finally, a list of measurement results is generated for this chain of trust.

TPM provides cryptographic algorithms and security storage. There are 24 PCRs in TPM, which are utilized to record the status information of the platform. PCR_{0-7} are used to record the measurement results of hardware, BIOS, and bootloader stages. Other PCRs record measurement results of runtime components based on the requirements of the system and platform. There are only two operations have been defined for PCRs by Trusted Computing Group's specifications, where `PCR_Reset` is used to reset it when the platform powers on, and the `PCR_Extend` is used to extend the new measurement result into it. After reboot, all PCRs are initialized. Once the platform starts up, all the values measured into a PCR cannot be reversed [40].

Remote Attestation: The remote attestation [41] is used to enable a verifier to validate the integrity of the target platform with the help of TPM. Specifically, when receiving an attestation request, the target platform collects integrity proof, including the measurement result, PCRs' values, and the signature signed by TPM. This signature is signed by the private key (i.e., Attestation Identity Key, AIK) of the specific TPM attached to the target platform, including the PCRs' values, to represent the identity of the target platform and the trustworthiness of the transferred PCRs' values. After receiving the integrity proof, the verifier first checks the validity of the integrity proof, i.e., checking whether the integrity proof has been tampered with or forged. After the validation result is positive, the verifier compares the content of measurement results with his expected value to judge the integrity of the target platform.

IMA: Sailer et al. proposed IMA [42] as a component of the kernel's integrity subsystem for expanding the scope of the chain of trust to the application layer. IMA provides a set of hooks to inspect and measure the file-loading events (e.g., executing a binary program, installing a kernel module, or reading a configuration file) for protecting the integrity of loaded files and executable programs within the platform. The measurement list of IMA records measurements of all events in order. An aggregate measurement value over these events is bound to a physical PCR in the TPM (PCR_{10} by default). Hence, on a trusted boot system, users can attest to the system's runtime integrity by checking the measurement results of IMA. IMA has been extensively studied by researchers, including optimizing IMA's measurement overhead [43], reducing IMA's measurement results size [44–46], and compensating the deficiency of IMA's remote attestation [47]. Besides, IMA also has been applied in various domains, such as resisting runtime attacks [48], protecting the integrity of Docker containers [49, 50], and protecting the integrity of IoT devices [51]. In IPOD2, we extend IMA to measure and record the operations during the outsourced data deletion process. By adopting remote attestation, IPOD2 enables users to verify the deletion process of outsourced data.

4. PROBLEM STATEMENT

4.1. System Model

In IPOD2, there are two entities involved: CSP and Users. CSP offers its resources for users to store and manage data. Users outsource their data to CSP to reduce overhead and access data flexibly. Figure 1 illustrates an example workflow depicting the process of data storage and management between CSP and users. This workflow consists of nine operations, which can be categorized into four stages: upload, download, delete, and verify. In the following, we provide a detailed description of the workflow based on the example depicted in Figure 1.

- **Upload:** ① The user uploads data to CSP. ② During the data storage process in CSP, a unique identifier is assigned to each outsourced data, and a special attribute is set for it. ③ After the data storage is complete, CSP returns the data identifier (i.e., F_C) to the user.

- **Download:** ④ The user downloads/accesses the outsourced data with the corresponding identifier from CSP.

- **Delete:** ⑤ The user deletes the outsourced data with the corresponding identifier from CSP. ⑥ CSP deletes the data identified by the identifier. During the deletion process, the data is deleted by an overwriting-based deletion method, while IMA measures the operations during the deletion process and TPM protects the measurement results.

- **Verify:** ⑦ After deletion, the user requests the deletion proof of the data with the corresponding identifier. ⑧ CSP returns the deletion proof identified by the identifier to the user. ⑨ The user verifies the measurement results of outsourced data according to the measurement results in the deletion proof.

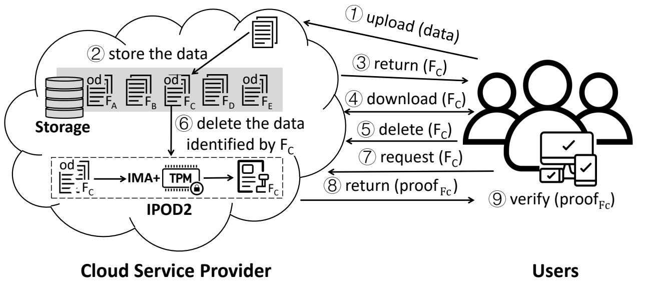


FIGURE 1. The system model of our scheme. Outsourced data is stored in CSP with a unique identifier (i.e., 'F_i' on the file) and a special attribute (i.e., 'od' on the file) used to distinguish it from the local files of CSP.

4.2. Threat Model

Aligned with [49,50,52], we assume that CSP possesses trusted hardware, i.e., TPM, to support Trusted Computing, and trusted boot [53] ensures the integrity of the kernel. Similar to other existing mechanisms based on Trusted Computing [49, 50], we do not consider DoS, runtime attacks, and physical attacks on the infrastructure of CSP. Users are unprivileged entities in the system provided by CSP, who can access and manage data through the interfaces provided by CSP. Users are subject to access permissions and restrictions defined by CSP and lack privileged access to the underlying system.

In our threat model, we focus on two types of adversaries, data thief [9, 10, 14–16] and misbehaving user [16]. A data thief, who can obtain opportunities to access the data disk of CSP and attempts to recover and access the deleted data. A misbehaving user, who doubts the credibility of the proof and alleges that CSP has failed to delete data as required, seeking compensation. [14–16] can not resist such a misbehaving user, because the deletion proof in these schemes is too simplistic and lacks credibility assurance, CSP does not have any detailed and trustworthy proof to prove that the data has been deleted as per the user's request.

4.3. Goals

To construct an irrecoverable and verifiable outsourced data deletion scheme, the design of IPOD2 needs to achieve the following goals:

Irrecoverability: The deleted outsourced data should become irrecoverable immediately, and no one can recover or obtain any information about the deleted data.

Verifiability: The deletion process of outsourced data can be verified by the user. To meet this, trusted and detailed deletion proof should be provided.

Efficiency: The processes of outsourced data deletion and deletion verification do not incur significant overhead to CSP and users. The content of the deletion proof should be concise for users.

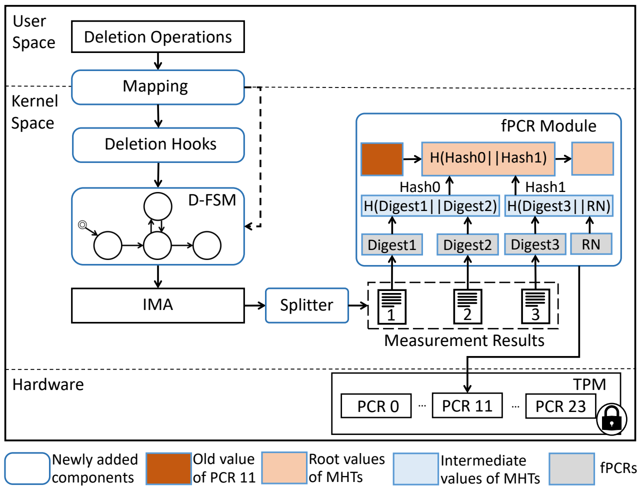


FIGURE 2. Overview architecture of IPOD2.

Privacy: During the verification process, users cannot obtain any information about other data or users through the deletion proof.

5. DESIGN OF IPOD2

5.1. Overview

IPOD2 aims to ensure the irrecoverability of deleted data and enable the user to determine whether the outsourced data has been deleted as required based on detailed and trustworthy deletion proof. Specifically, IPOD2 utilizes an overwriting-based method to delete outsourced data and extends IMA to measure the operations in the deletion process. The measurement results of the operations are further protected by TPM and used as deletion proof to prove the deletion process of outsourced data.

To construct IPOD2, there are four challenges to address. First, IMA can only identify the system calls rather than the operations. Second, since IMA is event-driven, it is necessary to specify measurement objects and measurement events in IPOD2. Third, the measurement results associated with the deletion process should be complete and protected by TPM. Last, after deletion, the user can verify whether the data has been deleted as required according to the measurement results.

To address these challenges, IPOD2 consists of four key components. Figure 2 shows the overview architecture of IPOD2. First, to enable IMA to identify the operations in the deletion process, we map operations in the user space to system calls in the kernel space through the combination of static and dynamic analysis. Second, based on the mapping, we design deletion hooks and a Deletion Finite State Machine (D-FSM) to identify the operations according to the execution of system calls. Third, we modify the measurement strategy of IMA to measure all operations in the deletion process and append more information (e.g., timestamp, operator, operation type) about the operations in the measurement results to make it more detailed. Besides, we split the measurement results related to different data and introduce a file-based PCR (fPCR) module to provide independent and TPM-based protection for the measurement result of each data. Last, we provide a verification mechanism based on remote attestation, which enables the user to verify whether the data has been deleted as required according to the measurement results.

To provide a comprehensive understanding of IPOD2, we present a step-by-step exploration of its key components in the subsequent subsections. In Section 5.2, we provide the construction process of mapping operations in the deletion process to system call sequences. In Section 5.3, based on the mapping, we describe the detailed process through which IPOD2 identifies operations according to system calls. In Section 5.4, we give the details of measuring operations by IMA and protecting the measurement results by TPM. In Section 5.5, we present the verification process for outsourced data deletion.

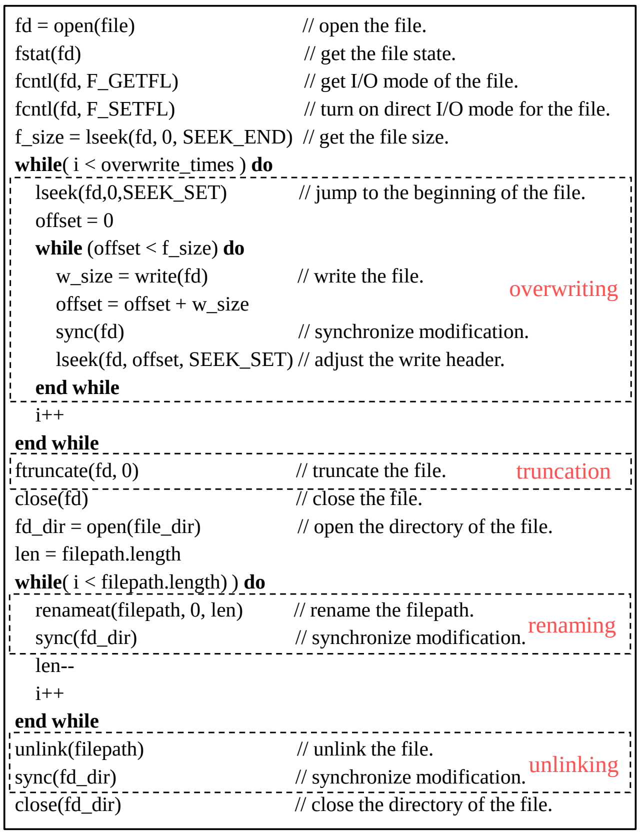
5.2. Mapping Operations to System Calls

Since IMA can only identify system calls in the kernel space rather than operations in the user space, to enable IMA to measure operations in the deletion process, we need to map these operations to system call sequences first. To achieve this, we use a combination of static and dynamic analysis methods to obtain this mapping.

Specifically, we select three commonly used overwriting-based deletion programs [22–24] on Linux for analysis. These programs encompass four distinct operations: overwriting, truncation, renaming, and unlinking. First, we conduct static analysis on the source code of these programs to extract a generic system call sequence representing these programs and understand the function of each system call. Based on the function of system calls, we establish a mapping from operations to system calls. Subsequently, to validate the accuracy of this mapping, we conduct dynamic analysis on these deletion programs using the `strace`[2]. By adjusting pertinent parameters (e.g., the number of overwrites, file size, and the length of the file name), we obtain diverse system call sequences. We then verified the correctness of the mapping by comparing the differences between these sequences.

Based on our analysis, Figure 3 presents the generic system call sequence representing these deletion programs, and indicates the mapping between operations and system calls. As shown in Figure 3, truncation is mapped to a single system call `ftruncate`, while other operations are mapped to a set of system calls that execute in a specific order. Taking the overwriting operation as an example, the process first invokes `lseek` to reposition the write offset to the file's beginning. Sub-

[2]https://strace.io/



```
fd = open(file)                         // open the file.
fstat(fd)                               // get the file state.
fcntl(fd, F_GETFL)                      // get I/O mode of the file.
fcntl(fd, F_SETFL)                      // turn on direct I/O mode for the file.
f_size = lseek(fd, 0, SEEK_END)         // get the file size.
while( i < overwrite_times ) do
  lseek(fd,0,SEEK_SET)                  // jump to the beginning of the file.
  offset = 0
  while (offset < f_size) do
    w_size = write(fd)                  // write the file.
    offset = offset + w_size
    sync(fd)                            // synchronize modification.
    lseek(fd, offset, SEEK_SET)         // adjust the write header.
  end while
  i++
end while
ftruncate(fd, 0)                        // truncate the file.
close(fd)                               // close the file.
fd_dir = open(file_dir)                 // open the directory of the file.
len = filepath.length
while( i < filepath.length) ) do
  renameat(filepath, 0, len)            // rename the filepath.
  sync(fd_dir)                          // synchronize modification.
  len--
  i++
end while
unlink(filepath)                        // unlink the file.
sync(fd_dir)                            // synchronize modification.
close(fd_dir)                           // close the directory of the file.
```

FIGURE 3. The system call sequence of the overwriting-based deletion method. The corresponding deletion-related operations are marked in red.

sequently, the process utilizes `write` to modify the content of the target file with random data, followed by `sync` to synchronize the modified content to the disk and `lseek` to reposition the write offset. This sequence is iterated until the entire target file is overwritten.

5.3. Identifying Measurement Objects

The original IMA is event-driven and used to measure the integrity of loaded files or modules, which means that IMA responds to specific events (e.g., file access) and performs integrity measurements for targeted files associated with these events. To enable IMA to measure the operations in the deletion process, we need to identify measurement objects and generate measurement events for them. To achieve this, we implant deletion hooks in the system calls involved in the sequence (obtained in Section 5.2) to monitor their execution. Based on the mapping between operations and system calls, we construct a D-FSM to identify operations in the deletion process according to the execution of system calls. Since IPOD2 is solely concerned with the deletion process of outsourced data, we set a special attribute for outsourced data and IPOD2 only measures the deletion process of the data possessing this attribute.

Specifically, to enable the kernel to distinguish outsourced data, we utilize the Linux extended attribute mechanism (`xattr`) to set a special attribute for files corresponding to outsourced data. Each outsourced data is associated with a D-FSM to identify operations in its deletion process. When a system call with the implanted deletion hook is executed, the hook checks whether the operated file has the special `xattr`. If the operated file has the special `xattr`, the deletion hook sends the execution information of this system call to the corresponding D-FSM. Otherwise, the deletion hook ignores the execution of this system call. The operations identified by D-FSM according to the execution information of system calls are measurement objects in IPOD2. After an operation is identified, D-FSM generates a measurement event for it. We provide the formal definition of D-FSM and its state transition rules below.

Formally, we define D-FSM as a 5-tuple (Q, Σ, T, Q_0, Q_F), which consists of a finite set of state Q, a finite set of input symbols Σ, a transition function $T : Q \times \Sigma \rightarrow Q$, an initial state $Q_0 \in Q$, and a set of final state $Q_F \subseteq Q$. An input symbol in Σ is the execution information of a system call. The final state set Q_F includes two states, i.e., the success final state Q_S, and the exception final state Q_E. Q_S indicates that the deletion process has been executed completely and correctly, and D-FSM ends successfully. Q_E indicates that there are unexpected operations during the deletion process, and thus D-FSM ends abnormally.

The state transition table of D-FSM is shown in Table 3, which is constructed using the acquired system call sequence. This table defines the state transition rules of D-FSM and enables D-FSM to identify corresponding operations in the deletion process based on the execution of system calls. The 'Operation' part in Table 3 refers to the operations identified by D-FSM. After identifying an operation, D-FSM generates a measurement event of this operation to trigger IMA to measure it.

Based on the analysis in Section 5.2, the overwriting-based deletion process has a specific initialization system call sequence (`open`→`fstat`→`fcntl`→`fcntl`). D-FSM proceeds to identity operations in the deletion process after identifying this sequence. To prevent misidentification due to operations or programs with the same preceding sequence of system calls, in any state of D-FSM, a system call is considered an unexpected system call (i.e., UNEXP in Table 3) if it has no transition rules defined in Table 3. Since each operation in the deletion process has been mapped to a system call or a set of ordered system calls, once an unexpected system call occurs, D-FSM enters the exception final state (Q_E) and generates a measurement event of the unexpected system call (i.e., UNEXP_info in Table 3) to trigger IMA to measure it. Then, D-FSM returns to the initial state (Q_0). The sequence diagram of D-FSM is shown in Figure 4.

TABLE 3. The state transition table of D-FSM.

Current State	Event	Next State	Operation
Q_0	open	**Init_open**	-
Init_open	fstat	**Init_fstat**	-
Init_fstat	fcntl	**Init_fcntl**	-
Init_fcntl	fcntl	**Init_fcntl**	INITIAL
Init_fcntl	lseek	**Do_lseek**	-
Do_lseek	write	**Do_write**	-
Do_write	write	**Do_write**	-
Do_write	sync	**Do_sync**	OVERWRITE
Do_sync	lseek	**Do_lseek**	-
Do_sync	fcntl	**Do_fcntl**	FCNTL
Do_sync	ftruncate	**Do_ftruncate**	FTRUNCATE
Do_fcntl	lseek	**Do_lseek**	-
Do_ftruncate	close	**Do_close**	CLOSE
Do_close	rename	**Do_rename**	-
Do_rename	rename	**Do_rename**	-
Do_rename	sync	**Do_rename**	RENAME
Do_rename	unlink	Q_S	UNLINK
Any state	UNEXP[1]	Q_E	UNEXP_opinfo[2]

UNEXP[1]: unexpected system call. UNEXP_info[2]: information of unexpected system call .

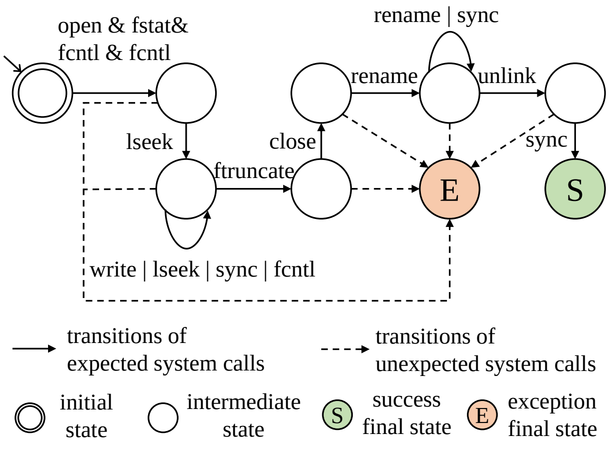

open & fstat& fcntl & fcntl
rename | sync
rename
unlink
lseek
close
ftruncate
sync
E
S
write | lseek | sync | fcntl
transitions of expected system calls
transitions of unexpected system calls
initial state
intermediate state
S success final state
E exception final state

FIGURE 4. The sequence diagram of D-FSM.

5.4. Measuring Operations in the Deletion Process

Directly using IMA to measure operations in the deletion process suffers from two limitations: 1) The original IMA does not perform duplicate measurements on the same measurement objects. However, the overwriting-based deletion process involves multiple overwriting and renaming, all these operations are required to be measured to generate detailed measurement results for the deletion process. 2) The original IMA records all measurement results together. During the verification process in IPOD2, users need to retrieve the measurement results corresponding to specific data. Storing all measurement results together would not only incur a significant overhead of communication and verification but also leak information about other data or users.

To enable IMA to measure the repeated operations in the deletion process, we modified the IMA's measurement logic for measurement objects in IPOD2. Specifically, we found that IMA uses a *measured* flag to mark repeated measurement objects and ignores them during measurement. In IPOD2, we reset the flag of all measurement objects before measurement to enable IMA to measure them repeatedly. Besides, to make the measurement results of operations in the deletion process more detailed, we enrich the content of measurement results related to the operations in IPOD2. If IMA receives a measurement event from D-FSM, it appends the information on the timestamp, operator, and operation type to the measurement results.

To partition the measurement results of different outsourced data, we introduce a Splitter to split the measurement results and store them separately based on the identifier of the data involved in measurement results. However, the number of PCRs in TPM is limited to 24, making it impossible to protect the measurement results of each data with independent physical PCRs.

To address this issue, we designed a file-based Platform Configuration Register (fPCR) module to provide independent and TPM-based protection for the measurement results of different data. The fPCR module maintains a set of fPCRs and assigns one fPCR for each data. An fPCR refers to a software PCR, which simulates the physical PCR to record the digest of the measurement results associated with the corresponding data. As illustrated in Figure 2, for the purpose of utilizing a single physical PCR to protect any number of fPCRs, fPCRs are organized in the form of Merkle Hash Tree (MHT), where its leaf nodes are fPCRs. Besides, to facilitate management and flexibly handling varying amounts of data, the number of fPCRs on each MHT is fixed, and fPCR module will create a new MHT when the fPCRs are exhausted. The assigned fPCRs' values are the digest of the measurement results (e.g., Digest1 in Figure 2), while the unassigned fPCRs are filled with random numbers (i.e., RN in Figure 2). The MHT's root value is calculated based on the leaf nodes' values, and all MHTs' root values are extended into the selected physical PCR (i.e., PCR_{11}) sequentially.

In this way, we establish a chain of trust from TPM to the measurement results, where the physical PCR in TPM protects the integrity of MHTs, MHTs protect the integrity of all fPCRs, and fPCRs protect the integrity of the measurement results.

Once a measurement result is generated, the workflow of fPCR module is as follows:

1. Extend the current measurement result into the corresponding fPCR as follows.

$$\textit{target-}fPCR := Hash(\textit{target-}fPCR, MR.digest)$$

   where *target-$fPCR$* refers to the fPCR corresponding to the outsourced data associated with the current measurement result, $MR.digest$ refers to the digest of the current measurement result.

2. Recalculate the root value of MHT containing the *target-$fPCR$* as follows.

$$\begin{cases} root := \textit{target-}fPCR \\ root := Hash(root||node_i) \end{cases}, (node_i \in AAI) \quad (1)$$

   where *root* refers to the root value of MHT containing the *target-$fPCR$*, AAI refers to the Auxiliary Authentication Information of *target-$fPCR$*, which is an array that includes the sibling nodes on the path from *target-$fPCR$* to *root*, $node_i$ refers to a sibling node contained within AAI.

3. Record the current value of PCR_{11} as *historyPCR*, which is used to verify *target-$fPCR$*.

4. Calculate the digest of all MHTs' root values as follows.

$$\begin{aligned} rDigest &:= \{0\text{x}0\}_{40} \\ rDigest &:= HASH(rDigest \mid\mid root_i) \end{aligned} \quad (2)$$

   where *rDigest* refers to the digest of all MHTs' root values, initialized as $\{0\text{x}0\}_{40}$ at the beginning of the computation and calculated by sequentially hashing all MHTs' root values, $root_i$ refers to a MHT's root value.

5. Extend the physical PCR_{11} with *rDigest* as follows.

$$PCR_{11} := PCR\_Extend(PCR_{11}, rDigest)$$

5.5. Verifying the Deletion of Outsourced Data

This section shows how to enable the user to verify the outsourced data deletion. Our verification process is derived from the remote attestation [41], which includes two stages: 1) upon receiving the user's verification request, CSP collects the deletion proof and sends it to the user; 2) the user verifies the outsourced data deletion based on the deletion proof.

Message Transferring: When verifying the deletion of outsourced data, the user sends a request with the identifier of the data and a *nonce*. *nonce* is used to ensure the freshness of the response of CSP and prevent replay attacks. Upon receiving the request, CSP collects the deletion proof of the corresponding data and sends it to the user. The deletion proof (D_P) consists of five parts, which are detailed below.

$$D\_P = \{Sign\{PCRs||nonce, AIK\}, PCRs, MHTs, fPCR, AAI, MRs\}$$

where $Sign\{PCRs||nonce, AIK\}$ denotes the signature generated by TPM's AIK for *PCRs* and *nonce*, *PCRs* consist of the values of PCR_{0-7} and PCR_{11}, $MHTs$ include the old value of PCR_{11} (*historyPCR*) and root values of all MHTs, $fPCR$ refers to the value of the fPCR associated with MRs, AAI refers to the the sibling nodes of $fPCR$, and MRs is the measurement results of the data deletion process.

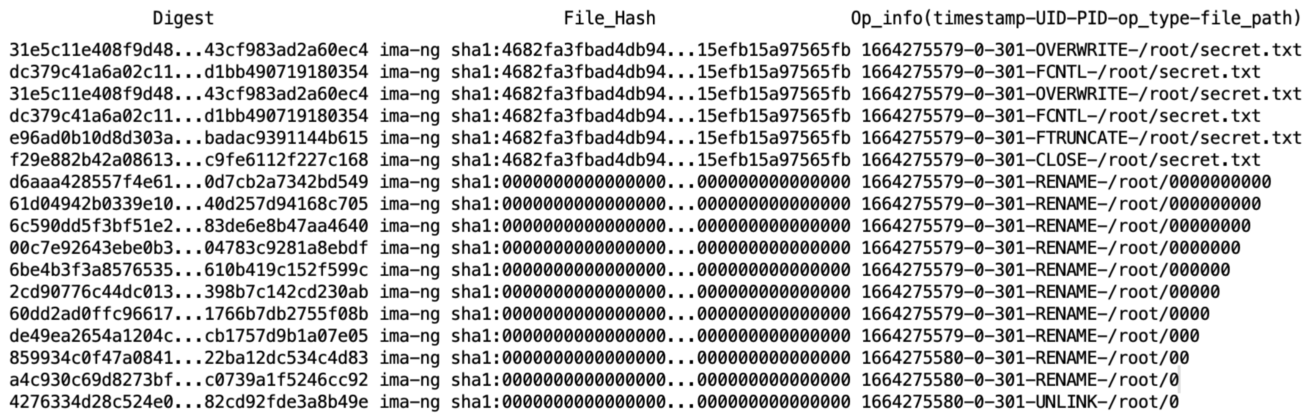
Workflow of Verification: IPOD2 establishes a chain of trust from TPM to the measurement results. Based on the deletion proof provided by CSP, the verification process in our scheme includes two stages: 1) The user verifies whether the measurement results have been tampered with or forged. 2) Based on the content of the measurement results that have not been tampered with or forged, the user verifies whether the data has been deleted as required.

To determine whether the measurement results have been tampered with or forged, the user verifies the measurement results with the following five steps:

- The user utilizes TPM's AIK certificate to verify the correctness of $Sign\{PCRs||nonce, AIK\}$. Afterward, the user compares the values of received *PCRs* with those in the correct signature to ensure that the values of the received *PCRs* have not been tampered with or forged. If the values of the received *PCRs* match those in the signature, we consider the values of the received *PCRs* are trusted. *nonce* is used to assess the freshness of the signature.

- Based on the trusted PCR_{0-7} values, the user verifies whether the bootstrapping process of CSP's platform is correct.

- Based on the trusted PCR_{11} value, the user verifies whether the received $fPCR$ is trusted. Specifically, the user generates a digest of the received $MHTs$ according to the equation 2. The received $MHTs$ is trusted if the digest equals PCR_{11} value. The user can obtain the trusted root value of MHT containing the received $fPCR$ (denoted as $root_{fPCR}$). The user calculates a simulated root value of the received $fPCR$ and AAI according to the equation 1. The received $fPCR$ is considered trustworthy, if the simulated root value equals to $root_{fPCR}$.



```
                 Digest                                        File_Hash                  Op_info(timestamp-UID-PID-op_type-file_path)
31e5c11e408f9d48...43cf983ad2a60ec4 ima-ng sha1:4682fa3fbad4db94...15efb15a97565fb 1664275579-0-301-OVERWRITE-/root/secret.txt
dc379c41a6a02c11...d1bb490719180354 ima-ng sha1:4682fa3fbad4db94...15efb15a97565fb 1664275579-0-301-FCNTL-/root/secret.txt
31e5c11e408f9d48...43cf983ad2a60ec4 ima-ng sha1:4682fa3fbad4db94...15efb15a97565fb 1664275579-0-301-OVERWRITE-/root/secret.txt
dc379c41a6a02c11...d1bb490719180354 ima-ng sha1:4682fa3fbad4db94...15efb15a97565fb 1664275579-0-301-FCNTL-/root/secret.txt
e96ad0b10d8d303a...badac9391144b615 ima-ng sha1:4682fa3fbad4db94...15efb15a97565fb 1664275579-0-301-FTRUNCATE-/root/secret.txt
f29e882b42a08613...c9fe6112f227c168 ima-ng sha1:4682fa3fbad4db94...15efb15a97565fb 1664275579-0-301-CLOSE-/root/secret.txt
d6aaa428557f4e61...0d7cb2a7342bd549 ima-ng sha1:0000000000000000...000000000000000 1664275579-0-301-RENAME-/root/0000000000
61d04942b0339e10...40d257d94168c705 ima-ng sha1:0000000000000000...000000000000000 1664275579-0-301-RENAME-/root/000000000
6c590dd5f3bf51e2...83de6e8b47aa4640 ima-ng sha1:0000000000000000...000000000000000 1664275579-0-301-RENAME-/root/00000000
00c7e92643ebe0b3...04783c9281a8ebdf ima-ng sha1:0000000000000000...000000000000000 1664275579-0-301-RENAME-/root/0000000
6be4b3f3a8576535...610b419c152f599c ima-ng sha1:0000000000000000...000000000000000 1664275579-0-301-RENAME-/root/000000
2cd90776c44dc013...398b7c142cd230ab ima-ng sha1:0000000000000000...000000000000000 1664275579-0-301-RENAME-/root/00000
60dd2ad0ffc96617...1766b7db2755f08b ima-ng sha1:0000000000000000...000000000000000 1664275579-0-301-RENAME-/root/0000
de49ea2654a1204c...cb1757d9b1a07e05 ima-ng sha1:0000000000000000...000000000000000 1664275579-0-301-RENAME-/root/000
859934c0f47a0841...22ba12dc534c4d83 ima-ng sha1:0000000000000000...000000000000000 1664275580-0-301-RENAME-/root/00
a4c930c69d8273bf...c0739a1f5246cc92 ima-ng sha1:0000000000000000...000000000000000 1664275580-0-301-RENAME-/root/0
4276334d28c524e0...82cd92fde3a8b49e ima-ng sha1:0000000000000000...000000000000000 1664275580-0-301-UNLINK-/root/0
```

FIGURE 5. An example of the measurement results of data deletion. These measurement results are corresponding to the file `/root/secret.txt`. Each measurement result contains three parts: the digest of the measurement result (Digest), the hash value of the file being operated (File_Hash), and the metadata of the sub-operation (Op_info). We omit some values in the 'Digest' and 'File_Hash' parts for better clarity in the display. The 'Op_infor' part includes the operation time (timestamp), the operator (UID and PID), and the operation type (op_type).

- Based on the trusted $fPCR$ value, the user verifies whether the received measurement results are trusted and have not been tampered with or forged by anyone as follows.

  $$\begin{cases} sDigest = \{0x0\}_{40} \\ sDigest = Hash(sDigest||Hash(e_i)) \end{cases}, (e_i \in MRs)$$

  $$sDigest \overset{?}{=} fPCR$$

  where $sDigest$ refers to a simulated digest of the received measurement results (MRs) and is initialized to $\{0x0\}_{40}$ at the beginning of the computation, e_i refers to an entry included in MR_s. If $sDigest$ equals the value of $fPCR$, it indicates that the received measurement results (MRs) are trusted.

Based on the trusted measurement results, the user can determine whether CSP has performed the expected deletion of the outsourced data as required. Figure 5 gives an example of the measurement results of the deletion process, which records the detailed information of the operations in the deletion process. According to the content of measurement results, it is clear that after initialization, the data is overwritten three times, truncated one time, renamed ten times, and unlinked one time. This is a complete and correct overwriting-based deletion process. The measurement results indicate that CSP has deleted the data as required. Besides, the user can further assess whether the execution of operations aligns with his/her expectations by examining the detailed information (e.g., timestamp, operator) recorded in the measurement results.

Since the verified measurement results are protected by TPM and are trusted, they cannot be tampered with or forged by anyone. If CSP has faithfully completed the deletion, the trusted measurement results can be used to prove the CSP's innocence and resist the slander of misbehaving user.

6. EVALUATION

To evaluate our design, we implement a prototype of IPOD2 on Ubuntu 20.04 with Linux kernel 5.4.120. We evaluate the prototype on Lenovo Savior R9000 with a 12-cores, 2.66GHz Intel i7-9750H CPU. For each test, we run 20 times to get the average result.

We evaluate IPOD2 to answer the following three key questions: (1) How much does IPOD2 affect the performance of the kernel? (2) How is the performance of the deletion measurement and proof verification in IPOD2? (3) What are the advantages of IPOD2 compared with existing schemes?

6.1. Overhead of IPOD2

To answer the first question, we evaluate the modified kernel in three aspects: 1) the number of added codes; 2) the overhead of the modified system calls; and 3) the overhead of the specific APIs.

The Number of Added Codes: We count the number of added codes in IPOD2 according to the SLOC[3]. The added codes can be divided into two categories: Struct (custom data structures) and Func (additional functions and methods). As shown in Table 4, the added codes are distributed in five parts: Deletion Hooks, Splitter, D-FSM, fPCR Module, and IMA. The added codes concentrated in the Deletion Hooks and fPCR Module, which are 325 LOCs and 839 LOCs respectively. The total number of added codes in IPOD2 is 1528 LOCs, less than 0.01% of the number of total codes in the kernel (27.8 million LOCs), which indicates that IPOD2 does not make significant

[3]`https://en.wikipedia.org/wiki/Source_lines_of_code`

TABLE 4. Added code of IPOD2.

Partition	Category (LoC)		
	Struct	Func	**Total**
Deletion Hooks	17	308	**325**
Splitter	-	27	**27**
D-FSM	9	159	**168**
fPCR Module	75	764	**839**
IMA	23	146	**169**
Total	**124**	**1404**	**1528**

modifications to the kernel.

Overhead of the System Calls: We invoke each system call 10 million times in each test to reduce deviation. As shown in the left part of Table 5, for most system calls, the additional cost of the 'IPOD2' setting is no more than $0.1\mu s$ and the additional percent is no more than 10%. For system calls like `write` and `unlink`, the additional cost of the 'IPOD2' setting is less than $0.3\mu s$ and the additional percent is less than 15%.

Overhead of the Specific APIs: We select several representative basic APIs to evaluate the overall performance of the kernel. We evaluate the overhead of these basic APIs by using LMbench3 [25], which is a well-known benchmark tool. As shown in the right part of Table 5, for most APIs, the result of the 'IPOD2' setting is the same as the 'Origin' setting. For some APIs (e.g., `exec`), the overhead of the 'IPOD2' setting is slightly larger than the 'Origin' setting (the additional percent is no more than 1%). This is because deletion hooks check `xattr` of files, which brings tiny additional overhead. According to the evaluation results, the overall performance of the kernel in the 'IPOD2' setting is close to that of the 'Origin' setting.

6.2. Performance of Deletion and Verification in IPOD2

To answer the second question, we evaluate the overhead of the deletion and verification in IPOD2. In the following experiments, we choose files with sizes from 100K to 1G. The used deletion algorithm includes 3 overwrites, 10 renames, 1 truncate, and 1 unlink.

Performance of Deletion: Figure 6 presents the deletion overhead of the 'Origin' setting and 'IPOD2' setting. We regard the increment of the 'IPOD2' setting relative to the 'Origin' setting as the measurement overhead introduced by IPOD2. As shown in Figure 6, the measurement overhead slowly increases with the file size. This is because overwriting larger files involves more `write` system calls, resulting in more executions of the deletion hook and state transitions of D-FSM, consuming more time. When the file is small (i.e., 100K), the measurement overhead is equal to the deletion overhead in the 'Origin' setting. As the file size increases, the impact of measurement overhead on deletion overhead becomes smaller.

TABLE 5. The performance evaluation results of system calls and APIs. The 'Origin' setting refers to the original kernel, and the 'IPOD2' setting refers to the kernel with IPOD2. The percent overhead calculation against the 'Origin' setting is shown at the end of the 'IPOD2' setting.

System Call	**Origin**	**IPOD2**	**API**	**Origin**	**IPOD2**
overhead(μs) - smaller is better			overhead(μs) - smaller is better		
open	0.92	0.98(+6.52%)	null call	0.28	0.28(0.00%)
fstat	0.39	0.40(+2.60%)	null I/O	0.36	0.36(0.00%)
fcntl	0.32	0.34(+6.52%)	stat	0.60	0.60(0.00%)
lseek	0.36	0.38 (+5.55%)	open/close	1.65	1.65(0.00%)
write	0.30	0.34(+13.33%)	signal install	0.34	0.34(0.00%)
ftruncate	0.31	0.32(+3.23%)	signal handle	0.90	0.90(0.00%)
close	0.59	0.63(+6.78%)	fork process	76.80	76.80(0.00%)
rename	3.37	3.63(+7.71%)	exec process	377.60	377.80(+0.01%)
unlink	1.88	2.14(+13.83%)	sh process	3025.40	3031.30(+0.20%)
fdatasync	50.33	50.58(+0.50%)			
fsetxattr	1.50	1.54(+2.67%)			
read	0.44	0.45(+2.27%)			

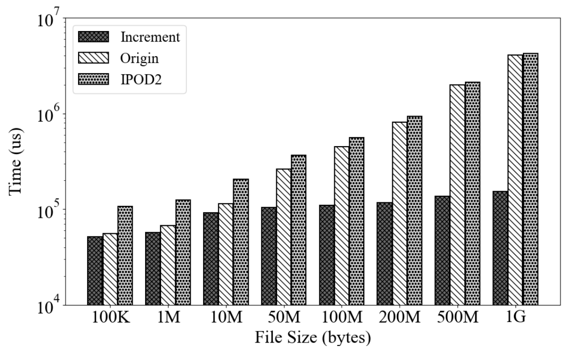


FIGURE 6. Deletion overheads for each setting. The 'Origin' setting refers to the deletion overhead in the original kernel, and the 'IPOD2' setting refers to the deletion overhead of the kernel with IPOD2. 'Increment' setting refers to the measurement overhead of IPOD2 during the deletion process.

Performance of Verification: Figure 7 presents the time cost of the different phases in the verification process. The process of collecting the deletion proof by CSP consists of two phases: *tpm-quote* (CSP collects the PCRs' value and generates a signature of TPM) and *proof-collect* (CSP collects the measurement results, corresponding fPCR, and MHTs' values). The process of verifying the deletion proof by the user consists of two phases: *tpm-checkquote* (the user verifies the signature and PCRs' value) and *proof-verify* (the user validates

TABLE 6. The time cost of the deletion and verification processes in different schemes. We evaluate the time cost of different files with the size of 100KB, 1MB, and 10MB.

Schemes \ Processes	Outsourced Data Deletion									Verification		
	Preprocessing (*ms*)			*Deletion* (*ms*)			*Total* (*ms*)			*Total* (*ms*)		
	100KB	1MB	10MB	100KB	1MB	10MB	100KB	1MB	10MB	100KB	1MB	10MB
Tang et al. [10]	4	40	400	300 (fixed)			304	340	700	-		
Xue et al. [15]	50	140	520	130 (fixed)			180	270	650	130 (fixed)		
Zhang et al. [16]	0			735	750	827	735	750	827	825	831	889
IPOD2	**0**			**147**	**151**	**219**	**147**	**151**	**219**	**25 (fixed)**		

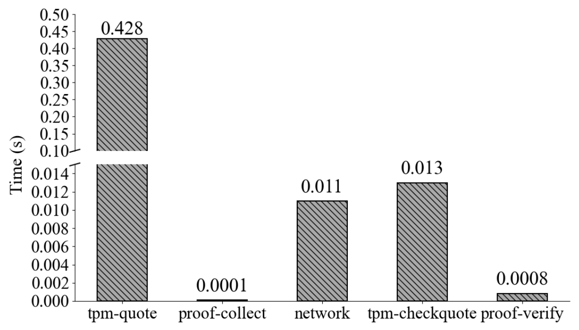


FIGURE 7. The time cost of the different phases in the verification process.

the deletion proof based on the trusted PCRs' value). Besides, the *network* phase includes the communication overhead of the user (request) and CSP (response). In IPOD2, the size of measurement results is related to the operations in the deletion process, which is independent of the data size. For a fixed deletion process, the size of the measurement results is constant for files of different sizes. As shown in Figure 7, the time cost of the whole verification process is $452.9ms$, where *tpm-quote* is $428ms$ (94.54% of the whole process). The verification overhead of deletion proof on the user side is less than 14 ms, including *tpm-checkquote* phase and *proof-verify* phase.

6.3. Comparing IPOD2 with Existing Schemes

In this section, we compare IPOD2 with the other three representative existing schemes [10, 15, 16] in terms of the overhead of data deletion and proof verification.

We follow [10] to reproduce the cryptographic operations used in their scheme by invoking the OpenSSL library[4]. And we follow [15] to reproduce the project, where elliptic curves are chosen by calling the Miracl library[5] API and set security parameter $\lambda = 80$ to satisfy the security requirements. Regarding [16], consistent with the settings in their paper, the used PDP scheme is derived from [54], the hash functions used in the deletion and verification processes are implemented by APHash, and each deletion process includes one overwriting where a user-defined data sequence used to overwrite. In IPOD2, we overwrite the data three times with the random data during the deletion process.

Deletion Process: All these Key management-based schemes [10, 15] take encrypting data as their preprocessing step for data deletion. They encrypt the data before outsourcing and later delete the data by discarding the corresponding key [10] or re-encrypting the data [15]. To evaluate the overall overhead of outsourced data deletion in these schemes, we consider the overhead of encrypting data as part of the overhead of outsourced data deletion overhead in these schemes. Zhang et al. [16] and IPOD2 are overwriting-based schemes without the preprocessing process. To accurately evaluate and compare the deletion overhead of all schemes, we select data with different sizes, ranging from 100KB to 10MB.

As shown in Table 6, since Tang et al. [10] encrypt data with a symmetric encryption technique, its preprocessing overhead exhibits a linear relationship with the data size. Xue et al. [15] utilize KP-ABE for data encryption, with the preprocessing overhead also raised with the data size. These two schemes benefit from operating on fixed-size data during deletion, i.e., deleting the corresponding key [10] or re-encrypting a part of the ciphertext [15]. Consequently, their deletion overhead is fixed, where the deletion overhead of [15] is minimal as it only requires one exponentiation. As for [16], during the deletion process, it first derives the data sequence used for overwriting based on a user-defined seed, then overwrites the original data with this data sequence, and finally generates tags of overwritten data based on PDP for verification. In contrast, for IPOD2, during the deletion process, it overwrites the original data with random data three times. Therefore, the deletion overhead of IPOD2 is significantly smaller than that of [16]. From the perspective of the overall deletion overhead of the scheme, across these schemes, IPOD2 incurs the minimum deletion overhead regardless of the deleted data size.

[4]`http://www.openssl.org/`
[5]`https://certivox.org/display/EXT/MIRACL`

Verification Process: Regarding the verification overhead of the schemes, we evaluate the overall verification overhead including the transmission and verification overheads of deletion proofs. In [15], since the small size of the re-encrypted result, the transmission overhead of deletion proof is negligible. The verification of deletion proofs includes the re-encryption and the comparison of the re-encryption results on the user side, where the overhead of comparison is negligible. Therefore, the overall verification overhead is equal to the deletion overhead. As for [16], it returns the overwritten results as deletion proofs for users to verify, resulting in the transmission and verification overheads of deletion proofs increasing with the size of the deleted data. In IPOD2, the deletion proof returned by CSP is small and independent of data size. Meanwhile, the verification of the deletion proof involves several hash operations and the comparison of the hash result and PCR's value. According to the evaluation of verification in Section 6.2, IPOD2 incurs the fixed overall verification overhead regardless of the deleted data size, which is the smallest among these schemes.

7. DISCUSSION

Even though IPOD2 satisfies the properties of the outsourced data deletion scheme and addresses the security and privacy problems in the deletion measurement and proof verification process, IPOD2 can be improved in the future, and we discuss it as follows.

Host Reboot: PCRs, fPCRs, and the measurement results generated by IMA will be reset and cleared after the host reboot, which causes the loss of the previous measurement results. But actually, the current cloud platform generally adopts uninterruptible power supply and hot backup technologies to ensure the availability and continuity of services. The hosts that are backups of each other perform the same operations and provide the same services. Therefore, we have reason to believe that the host reboot has little impact on our scheme.

Applicability: IPOD2 introduces a new design for a verifiable outsourced data deletion solution based on trusted hardware. The current prototype of IPOD2 is implemented on a Linux kernel with TPM, and it applies to magnetic media storage devices that support in-place updates by overwriting. We plan to address the porting of IPOD2 to other platforms and adaptation for SSDs and file systems with journaling or snapshots in future work.

8. CONCLUSION

This paper introduces IPOD2, a new solution for irrecoverable and verifiable outsourced data deletion. IPOD2 extends IMA to measure the operations in the deletion process and protects the measurement results by TPM for verification. Based on the mapping between operations in the deletion process and system calls, IPOD2 utilizes the deletion hooks and D-FSM to identify the operations according to the execution sequence of system calls. The identified operations are further measured by IMA, and the measurement results of them are protected by TPM. In this way, the user can verify whether the data has been deleted as required according to the trustworthy measurement results. The performance evaluation demonstrated that IPOD2 has good performance and brings acceptable overhead for each participant.

SOURCE CODE AVAILABILITY STATEMENT

IPOD2 is publicly available as an open-source project: `https://github.com/PKURoC/IPOD2`

CONFLICT OF INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

ACKNOWLEDGEMENTS

This work was supported by the National Key R&D Program of China [Grant No. 2022YFB2703301]; and the National Natural Science Foundation of China [Grant No. 61672062, No. 61232005].

REFERENCES

[1] Liu, Z., Wang, S., and Liu, Y. (2023) Blockchain-based integrity auditing for shared data in cloud storage with file prediction. *Comput. Networks*, 236, 110040.

[2] Wang, T., Zhou, Y., Ma, H., and Zhang, R. (2023) Flexible and controllable access policy update for encrypted data sharing in the cloud. *Comput. J.*, 66, 1507–1524.

[3] Lin, H., Zhao, Z., Gao, F., Susilo, W., Wen, Q., Guo, F., and Shi, Y. (2021) Lightweight public key encryption with equality test supporting partial authorization in cloud storage. *Comput. J.*, 64, 1226–1238.

[4] Parast, F. K., Sindhav, C., Nikam, S., Yekta, H. I., Kent, K. B., and Hakak, S. (2022) Cloud computing security: A survey of service-based models. *Comput. Secur.*, 114, 102580.

[5] Luo, F., Al-Kuwari, S. M., Lin, C., Wang, F., and Chen, K. (2022) Provable data possession schemes from standard lattices for cloud computing. *Comput. J.*, 65, 3223–3239.

[6] Islam, S., Ouedraogo, M., Kalloniatis, C., Mouratidis, H., and Gritzalis, S. (2018) Assurance of security and privacy requirements for cloud deployment models. *IEEE Trans. Cloud Comput.*, 6, 387–400.

[7] Hao, J., Liu, J., Wu, W., Tang, F., and Xian, M. (2020) Secure and fine-grained self-controlled outsourced data deletion in cloud-based iot. *IEEE Internet Things J.*, 7, 1140–1153.

[8] Sadowski, J., Viljoen, S., and Whittaker, M. (2021) Everyone should decide how their digital data are used - not just tech companies. *Nature*, 595, 169–171.

[9] Geambasu, R., Kohno, T., Levy, A. A., and Levy, H. M. (2009) Vanish: Increasing data privacy with self-destructing data. *Proceedings of the 18th USENIX Security Symposium (USENIX Security 2009)*, Montreal, Canada, 10-14 August, pp. 299–316. USENIX Association, Berkeley.

[10] Tang, Y., Lee, P. P. C., Lui, J. C. S., and Perlman, R. J. (2012) Secure overlay cloud storage with access control and assured deletion. *IEEE Trans. Dependable Secur. Comput.*, 9, 903–916.

[11] European Parliament and Council of the European Union (2016). Regulation (EU) 2016/679 of the european parliament and of the council on the protection of natural persons with regard to the processing of personal data and on the free movement of such data. `https://eur-lex.europa.eu/eli/reg/2016/679/oj`. (accessed on 1 April 2024).

[12] Gamvros, A. and Wang, L. (2021). PIPL: A game changer for companies in china. `https://www.dataprotectionreport.com/2021/08/pipl-a-game-changer-for-companies-in-china/`. (accessed on 20 October 2023).

[13] Castelluccia, C., Cristofaro, E. D., Francillon, A., and Kâafar, M. A. (2011) Ephpub: Toward robust ephemeral publishing. *Proceedings of the 19th annual IEEE International Conference on Network Protocols (ICNP 2011)*, Vancouver, BC, Canada, 17-20 October, pp. 165–175. IEEE Computer Society, Washington, DC.

[14] Mo, Z., Xiao, Q., Zhou, Y., and Chen, S. (2014) On deletion of outsourced data in cloud computing. *Proceedings of 7th IEEE International Conference on Cloud Computing (IEEE CLOUD 2014)*, Anchorage, AK, 27 June - 2 July, pp. 344–351. IEEE Computer Society, Washington, DC.

[15] Xue, L., Yu, Y., Li, Y., Au, M. H., Du, X., and Yang, B. (2019) Efficient attribute-based encryption with attribute revocation for assured data deletion. *Inf. Sci.*, 479, 640–650.

[16] Zhang, M., Zhang, H., Yang, Y., and Shen, Q. (2019) PTAD: provable and traceable assured deletion in cloud storage. *Proceedings of IEEE Symposium on Computers and Communications (ISCC 2019)*, Barcelona, Spain, 29 June - 3 July, pp. 1–6. IEEE, New York.

[17] Perlman, R. J. (2007) File system design with assured delete. *Proceedings of the Network and Distributed System Security Symposium (NDSS 2007)*, San Diego, California, 28 February - 2 March, pp. 83–88. The Internet Society, Reston.

[18] Tang, Q. (2015) From ephemerizer to timed-ephemerizer: Achieve assured lifecycle enforcement for sensitive data. *Comput. J.*, 58, 1003–1020.

[19] Mo, Z., Qiao, Y., and Chen, S. (2014) Two-party fine-grained assured deletion of outsourced data in cloud systems. *Proceedings of the 34th IEEE International Conference on Distributed Computing Systems (ICDCS 2014)*, Madrid, Spain, 30 June - 3 July, pp. 308–317. IEEE Computer Society, Washington, DC.

[20] Perito, D. and Tsudik, G. (2010) Secure code update for embedded devices via proofs of secure erasure. *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS 2010)*, Athens, 20-22 September, pp. 643–662. Springer, Berlin.

[21] Kissel, R., Regenscheid, A., and Scholl, M. NIST Special Publication 800-88 Revision 1: Guidelines for media sanitization. `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-88r1.pdf`. (accessed on 7 December 2023).

[22] Plumb, C. shred - linux man page. `https://linux.die.net/man/1/shred`. (accessed on 20 October 2023).

[23] Durak, B. wipe - linux man page. `https://linux.die.net/man/1/wipe`. (accessed on 20 October 2023).

[24] Jagdmann, D. srm - linux man page. `https://linux.die.net/man/1/srm`. (accessed on 20 October 2023).

[25] McVoy, L. W. and Staelin, C. (1996) lmbench: Portable tools for performance analysis. *Proceedings of the USENIX Annual Technical Conference (USENIX ATC 1996)*, San Diego, California, 22-26 January, pp. 279–294. USENIX Association, Berkeley.

[26] Shamir, A. (1979) How to share a secret. *Commun. ACM*, 22, 612–613.

[27] Goyal, V., Pandey, O., Sahai, A., and Waters, B. (2006) Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006)*, Alexandria, VA, 30 October - 3 November, pp. 89–98. ACM, New York.

[28] Ateniese, G., Burns, R. C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z. N. J., and Song, D. X. (2007) Provable data possession at untrusted stores. *Proceedings of ACM Conference on Computer and Communications Security (CCS 2007)*, Alexandria, Virginia, 28-31 October, pp. 598–609. ACM, New York.

[29] Reardon, J., Basin, D. A., and Capkun, S. (2013) SoK: Secure data deletion. *2013 IEEE Symposium on Security and Privacy (S&P 2013)*, Berkeley, CA, 19-22 May, pp. 301–315. IEEE Computer Society, Washington, DC.

[30] Gutmann, P. (1996) Secure deletion of data from magnetic and solid-state memory. *Proceedings of the 6th USENIX Security Symposium (USENIX Security 1996)*, San Jose, CA, 22 – 25 July, pp. 77–89. USENIX Association, Berkeley.

[31] Pfitzner. Pfitzner deletion method. `https://www.lifewire.com/data-sanitization-methods-2626133`. (accessed on 2 April 2024).

[32] Schneier, B. (2007) *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons, Hoboken.

[33] Naval Information Systems Management Center. Information systems security (INFOSEC) program guidelines. `https://irp.fas.org/doddir/navy/5239_26.htm`. (accessed on 17 October 2023).

[34] U.S. Department of Defense. National industrial security program operating manual (NISPOM). `https://www.federalregister.gov/documents/2020/12/21/2020-27698/national-industrial-security-program-operating-manual-nispom`. (accessed on 2 April 2024).

[35] Communication Security Establishment of Canada. It security guidance 06: Clearing and declassifying

electronic data storage devices. https://cyber.gc.ca/en/guidance/it-media-sanitization-itsp40006. (accessed on 1 December 2023).

[36] The United States Air Force. Air force system security instruction 5020. https://cryptome.org/afssi5020.htm. (accessed on 1 December 2023).

[37] Communications Electronics Security Group. HMG IA/IS 5 Secure Sanitisation of Protectively Marked Information or Sensitive Information document. https://www.lifewire.com/data-sanitization-methods-2626133. (accessed on 16 October 2023).

[38] New Zealand Information Technology Security Authority. The new zealand information security manual. https://www.gcsb.govt.nz/publications/the-nz-information-security-manual/. (accessed on 3 October 2023).

[39] TCG Specification. (2007) Architecture overview. *Specification Revision*, 1, 1–24.

[40] Arthur, W., Challener, D., and Goldman, K. (2015) *A Practical Guide to TPM 2.0: Using the New Trusted Platform Module in the New Age of Security*. Apress, New York.

[41] Intel. Remote Attestation. https://tpm2-software.github.io/tpm2-tss/getting-started/2019/12/18/Remote-Attestation.html. (accessed 20 December 2023).

[42] Sailer, R., Zhang, X., Jaeger, T., and van Doorn, L. (2004) Design and implementation of a TCG-based integrity measurement architecture. *Proceedings of the 13th USENIX Security Symposium (USENIX Security 2004)*, San Diego, CA, 9-13 August, pp. 223–238. USENIX, Berkeley.

[43] Son, J., Koo, S., Choi, J., Cho, S., Baek, S., Jeon, G., Park, J., and Kim, H. (2017) Quantitative analysis of measurement overhead for integrity verification. *Proceedings of the Symposium on Applied Computing (SAC 2017)*, Marrakech, Morocco, 3-7 April, pp. 1528–1533. ACM, New York.

[44] Jaeger, T., Sailer, R., and Shankar, U. (2006) PRIMA: policy-reduced integrity measurement architecture. *Proceedings of the 11th ACM Symposium on Access Control Models and Technologies (SACMAT 2006)*, California, 7-9 June, pp. 19–28. ACM, New York.

[45] Luo, W., Liu, W., Luo, Y., Ruan, A., Shen, Q., and Wu, Z. (2016) Partial attestation: Towards cost-effective and privacy-preserving remote attestations. *Proceedings of IEEE Trustcom/BigDataSE/ISPA*, Tianjin, 23-26 August, pp. 152–159. IEEE, New York.

[46] Rauter, T., Höller, A., Kajtazovic, N., and Kreiner, C. (2015) Privilege-based remote attestation: Towards integrity assurance for lightweight clients. *Proceedings of the 1st ACM Workshop on IoT Privacy, Trust, and Security (IoTPTS@AsiaCCS 2015)*, Singapore, 14 April, pp. 3–9. ACM, New York.

[47] Stumpf, F., Fuchs, A., Katzenbeisser, S., and Eckert, C. (2008) Improving the scalability of platform attestation. *Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing (STC 2008)*, Alexandria, VA, 31 October, pp. 1–10. ACM, New York.

[48] Davi, L., Sadeghi, A., and Winandy, M. (2009) Dynamic integrity measurement and attestation: towards defense against return-oriented programming attacks. *Proceedings of the 4th ACM Workshop on Scalable Trusted Computing (STC 2009)*, Chicago, Illinois, 13 November, pp. 49–54. ACM, New York.

[49] Luo, W., Shen, Q., Xia, Y., and Wu, Z. (2019) Container-ima: A privacy-preserving integrity measurement architecture for containers. *Proceedings of the 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, Chaoyang District, Beijing, 23-25 September, pp. 487–500. USENIX Association, Berkeley.

[50] Benedictis, M. D. and Lioy, A. (2019) Integrity verification of docker containers for a lightweight cloud environment. *Future Gener. Comput. Syst.*, 97, 236–246.

[51] Lu, D., Han, R., Shen, Y., Dong, X., Ma, J., Du, X., and Guizani, M. (2021) xTSeH: A trusted platform module sharing scheme towards smart iot-ehealth devices. *IEEE J. Sel. Areas Commun.*, 39, 370–383.

[52] Noman, A. and Adams, C. (2014) Hardware-based dlas: Achieving geo-location guarantees for cloud data using tpm and provable data possession. *Proceedings of International Conference on Computer and Information Technology (ICCIT 2014)*, Dhaka, Bangladesh, 22-23 December, pp. 236–246. IEEE, New York.

[53] Richard, M., Ning, S., Shane, W., Jimmy, W., and Ren, Q. Trusted boot (tboot). https://trustedcomputinggroup.org/resource/trusted-boot/. (accessed on 7 December 2023).

[54] Ateniese, G., Burns, R. C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z. N. J., and Song, D. X. (2007) Provable data possession at untrusted stores. *Proceedings of the 2007 ACM Conference on Computer and Communications Security (CCS 2007)*, Alexandria, Virginia, 28-31 October, pp. 598–609. ACM, New York.