MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

Hardware-encrypted disks in Linux

Master's Thesis

ŠTĚPÁN HORÁČEK

Brno, Fall 2022

MASARYK UNIVERSITY

FACULTY OF INFORMATICS

Hardware-encrypted disks in Linux

Master's Thesis

ŠTĚPÁN HORÁČEK

Advisor: Ing. Milan Brož, Ph.D.

Department of Computer Systems and Communications

Brno, Fall 2022


FACULTAS ARTIS INFORMATICÆ
UNIVERSITAS MASARYKIANA

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Štěpán Horáček

Advisor: Ing. Milan Brož, Ph.D.

Abstract

As disk encryption becomes a standard part of the security of many devices, many look towards solutions which might improve the performance, efficiency or security of the process. Such a solution can be seen in hardware disk encryption. However, hardware disk encryption is not without its flaws.

This work takes a look at the hardware disk encryption and its security, focusing on the popular Opal standard. Several relevant state-of-the-art attacks are introduced and analysed in the context of hardware disk encryption. In order to be able to understand the problematics more deeply, this work implements a set of tools for Opal devices. Namely, a utility to manage Opal disks, a utility to discover the proprieties of an Opal disk, and a utility to acquire random bytes generated by an Opal disk. Afterwards, these tools are used to analyse the security of the disks against a selection of attacks and to provide insight into how many optional features are offered by the disks.

Keywords

disk encryption, self-encrypting drive, Linux, storage security

Contents

1 Introduction

Whether it is a disk of a corporate laptop, personal desktop, or server storage, all of these devices may very likely contain sensitive information such as company internal data, session cookies, or even medical records. According to Verizon 2022 Data Breach Investigation Report [1], lost or stolen assets made up almost 5% of the analysed security incidents (not necessarily data breaches, due to the difficulty of finding out if there was one with lost or stolen assets). Furthermore, this percentage does not account for other possible attacks, where the disk could be compromised without stealing the device, such as unauthorised access to server disks.

As such, it can be seen that some form of protection for the confidentiality of data on the disk is necessary. Disk encryption may seem like an obvious solution. However, traditional software disk encryption has several flaws, such as increased energy demand [2] or the requirement of keys being present in the memory at all times, presenting a considerable vulnerability. Hardware disk encryption can help solve many issues of software disk encryption, but it also introduces other new issues.

In this thesis, we will introduce the idea of hardware disk encryption, the basic terms used in this area, and the categorisation of different approaches to hardware disk encryption. Afterwards, we will focus on the Opal standard as a representative of the self-encrypting disk approach to hardware disk encryption. Next, we will look at possible attacks on devices using hardware disk encryption. Finally, we will introduce our own tool to manage Opal disks and read their properties, and analyse data acquired using this tool.

2 Hardware disk encryption

Hardware disk encryption is a technology that provides confidentiality of data stored on a storage device using encryption provided by hardware specialised for encrypting data transferred to and decrypting data transferred from disk [3].

Generally, disk encryption is done by encrypting the saved data using a data encryption key (DEK), also sometimes known as the media encryption key (MEK). In order to allow changing of passwords and having multiple passwords, the DEK is not cryptographically bound to any particular password and instead is saved in an encrypted form. It is encrypted using a key encryption key (KEK) generated from a password and, therefore, cryptographically bound to that specific password.

The data encryption process can be conducted in logically and physically different places, depending on the type of disk encryption. Even though software disk encryption often uses cryptographic co-processors for hardware acceleration, since they are for general usage and are not limited to disk input/output, we consider them out of the scope of this thesis. In the following sections of this chapter, we will further describe two types of hardware disk encryption.

2.1 Self-encrypting drives

Self-encrypting drive (SED) is a storage device with built-in disk encryption hardware [4]. Having built-in disk encryption hardware enables the disk to encrypt the incoming data and decrypt the outgoing data automatically. The data saved on the disk is protected two-fold: by access control and by encryption. Access control denies input/output operations until the host is authorised. Encryption protects the device even from attacks with physical access, where the access control could be bypassed.

The encryption of the data is usually active even if the security mechanism of the device is not activated and a password does not protect the device. This has the advantage that a non-activated disk can be activated and protected by a password without a need to re-encrypt all data on the disk. Thanks to this, it is also possible to quickly erase

data on the disk by erasing the DEK. SEDs also might not require management by the host system after the initial unlocking. Thanks to the usage of specialised encryption chips, SEDs offer increased performance, reduced CPU usage, and reduced power consumption [2] compared to software encryption. Since the encryption keys are not actively used by the host system, they do not have to be in the memory, which provides protection against malicious software running in the host system.

In order to provide an interface to control the encryption of the disk, such as unlocking or changing the password, there exist several standards. Other than the series of standards defined by the Trusted Computing Group discussed in detail later in Chapter 3, there are other standards used by SEDs.

2.1.1 ATA Security Feature Set

ATA Security Feature Set [5] allows one to protect the data and configuration of the disk. It offers commands to set a password, unlock the disk, erase the data on the disk, freeze the device, and disable the password.

In order to authenticate to the disk, there are two passwords, the Master password, initially set by the manufacturer, and the User password. The usage of the Master password to unlock or disable the security can be disabled by setting the Master Password Capability bit while setting the User password. Even if the Master password is disabled for unlocking the disk or disabling the locking, it can still be used for erasing the disk. The disk protection is enabled by setting the User password. After the device is turned on again, the device is locked and to access the data on the disk or use some of the ATA commands, it needs to be unlocked by providing a password using the unlock command. By disabling the User password, the protection of the device can be disabled.

However, the ATA Security Feature Set itself does not provide encryption of the data saved on the disk, only access control. Nevertheless, some of the SED solutions use ATA Security commands as their interface to provide access to their own implementation of the disk encryption and disk unlocking [6].

2.1.2 Proprietary solutions

There also exist proprietary alternatives to the previously introduced standards. Such disks are then often managed either by software downloaded from the manufacturer's site or by using software from a special partition on the disk. An example of such are disks from Western Digital's My Passport and My Book series [7] or disks in IBM PureData System for Analytics N3001 [8].

2.2 Inline encryption hardware

Compared to the previously described self-encrypting drives, inline encryption hardware is separated and independent from the disk. Inline encryption hardware is similar to classic cryptographic accelerators. However, inline encryption hardware, instead of using the same memory for input and output data, optimises the input/output process by writing and reading the data directly to and from the disk [9].

Inline encryption hardware shares several of SEDs advantages, such as reduced power consumption, reduced CPU usage, or increased performance, compared to purely software encryption. However, unlike SEDs, inline encryption devices usually do not have any access control or authentication capabilities. Instead, they allow one to simply insert a DEK into a keyslot, select a keyslot, and remove a DEK from a keyslot. The data passing through the inline encryption device to the disk is then encrypted using the DEK from the selected key slot. Similarly, data passing from the disk are decrypted by the inline encryption hardware. Since key management is entirely in control of the host system, which has knowledge of the file system mapping, inline encryption can be used not only for block-level encryption but also for filesystem-level encryption. But this flexibility has a downside in that the host needs to actively control the encryption.

An example of inline encryption hardware standard is the Qualcomm Inline Crypto Engine [10] used by some Android devices.

3 TCG Opal 2.0

TCG Opal Security Subsystem Class 2.0 (hereinafter referred to simply as "Opal standard") is a specification for storage devices, aiming to provide confidentiality of stored data while the conforming disk is powered off [11]. It is one of the representatives of the self-encrypting drive approach to hardware disk encryption, and as a prominent standard for hardware disk encryption for personal computers, we will focus primarily on it in the rest of this thesis. Information for this chapter comes primarily from the Core standard [12], defined later, and the Opal standard [11].

Opal offers several features to help with securing data on the disk. Some of those features are locking ranges, allowing to deny access to only part of the disk, authorities, allowing to specify a set of capabilities to credentials, or shadow MBR, to allow pre-boot authentication even with a fully encrypted disk. Together with the architecture, protocols, and many different properties, the standard can get very complex and yet its understanding is fundamental for any elementary work with devices implementing it. In this chapter, we are planning to introduce these features together with the context within which they are contained.

3.1 Structure of the standard

The Opal standard is defined as a subsystem extending the *TCG Storage Architecture Core Specification* [12] (in the future referred to simply as "Core standard"). The Core specification introduces many powerful and interesting features. However, this might increase the cost of the design and manufacturing of a fully compliant device. In order to solve this problem, there are defined several different Security Subsystem Classes (SSC), determining only a small set of mandatory features, leaving most of the features optional. Together with the set of mandatory features, SSCs also state the limits of certain properties or may specify additional features. Each of these SSCs is focused on a different use case. These SSCs are namely:

- *Opal SSC* is targeted at corporate and personal usage. As this is the most available subsystem, it will be the target of our focus and be described more closely in the rest of this chapter.
- *Opalite SSC* is simplified Opal. Does not mandate features such as locking ranges, decreases the minimal number of admin and user authorities, or additional tables to store arbitrary data tables [13].
- *Pyrite SSC* is encryption-less Opalite. However, it does not mandate encryption of data saved on disk and instead may offer only logical access control [14].
- *Ruby SSC* is focused on data centres and server drives. Ruby SSCs may provide only a single global locking range, a weaker configuration of access control, and no pre-boot authentication support [15]. Replacing the older Enterprise subsystem.

The list of officially certified products and their SSCs can be found at TCG's Storage Certified Products web page[1]. The web page contains a list of products that were certified and requested to be published. At the time of writing, the list contains one product. The certified test house ULINK Technology offers its own list of certified disks on its web site[2]. Some data protection solutions also provide a list of supported drives with their SSCs. However, in all of those cases, the listed drives implement mainly only Opal, Pyrite or Enterprise SSCs. We have not been able to find any listings of Opalite or Ruby SSCs, which we suspect is mostly because of the short length of existence in the case of Ruby SSC, and being only a simplified version of other SSC in the case of Opalite SSC.

Other than the Core and SSC standards defining the fundamentals of the devices, there are defined also Feature Sets. These Feature Sets expand the standards with less fundamental features. Some of those are the Additional DataStore [16], providing storage for arbitrary data, the Block SID Authentication [17], allowing to disable the owner's authority until a power cycle, the PSK Secure Messaging [18], providing secure communication between the host and the device using a

1. `https://trustedcomputinggroup.org/membership/certification/storage-certified-products/`
2. `https://ulinktech.com/tcg-certification-compliance-test-suite-list/`

pre-shared key, the PSID [19], defining a way to reset the device into the manufactured state, or the Single User Mode [20]. However, we will focus primarily on those that are mandatory in the Opal specification. Finally, there are also other expanding standards, such as the Storage Interface Interactions Specification (SIIS) [21], describing the mapping of secure commands used by TCG devices to commands of specific disk interfaces and reset types, or the Secure Messaging addendum [22], describing TLS-based secure communication between the host and the device.

3.2 Architecture

The Core standard [12] splits the storage device implementing it into two parts: the Trusted Peripheral and Security Providers.

A *Trusted Peripheral* (TPer) is a component located on the disk that provides the security of the data on the disk. TPer mediates the communication between its Security Providers and the host. In order to maintain the state of the device, tables are used. There are two types of tables: byte tables, consisting of raw bytes, and object tables, with defined columns and rows, called objects. Each template defines its tables, such as the key tables containing DEKs, or the meta table called the Table table, which contains information about all the other tables in the SP. Although the Core standard specifies a way to create new tables or delete existing ones, this feature is not required by the Opal standard.

Security Provider (SP) is defined as a set of tables, methods and access control. SPs are derived from a set of templates, which define a set of tables and methods aimed at one functionality, a subset of which the SP implements. The templates are described more closely in a later Section 3.7. The Opal 2.0 standard defines that at least the Admin SP and Locking SP must be present in the TPer. The Admin SP is tasked with administrating the TPer and other SPs, which may include creating new SPs, deleting existing SPs, or providing information about SPs. The Locking SP provides access to functionality such as managing locking ranges, locking the drive, or managing access control. Both of these SPs are described more closely in later Section 3.8 and Section 3.9.

3.3 Communication

In order to send commands to the TPer or the SPs and receive responses, the Core standard [12] defines a specific protocol. This protocol is split into several layers.

Starting from the lowest layer described in this thesis is the interface layer. The interface layer corresponds to the communication with the control unit of the disk. Depending on the disk interface, there can be different security commands used to communicate. The Core standard abstracts these commands into two commands: IF-RECV and IF-SEND for receiving data from disk and sending data to disk, respectively. These commands facilitate all the communication necessary to communicate with the higher layers and are described more closely in Section 4.2.

The following layer is the TPer layer, a layer used primarily for the allocation of ComID, a number identifying the caller described in more detail in Subsection 3.3.1. Since on this layer, the ComIDs are not used yet, and so the state of the communication cannot be kept, the communication is only one way, each "session" consisting only of one command.

The Communication layer supports ComIDs and therefore allows two-way communication and is used primarily for further ComID management, such as getting the state of ComID or resetting the state of ComID.

The Management layer facilitates establishing of a session between an SP and the host. Since the session number is not issued yet, this layer uses Control Sessions with a static session number.

The final layer is the Session layer. At this layer, the session is already established, and communication can be performed using methods sent in packets. Most of the communication occurs in this layer with the use of method invocations.

3.3.1 ComID

One of the information required in order to send a command is the *ComID*, a number identifying the caller (e.g. application of the host). It ensures that responses to method calls will be received by the correct application since there can be at one time multiple callers. There

are three types of ComIDs: statically allocated ComID, dynamically allocated ComID and special ComIDs. Statically allocated ComID can be acquired through the Level 0 Discovery process, described in Section 3.6. The Discovery process simply informs about the base ComID number and the number of static ComIDs. Dynamically allocated ComID can be acquired using the `GET_COMID` command. There also exist special ComIDs that are used to invoke the Level 0 Discovery or for ComID management.

Every ComID is in one of a few states. These states are Inactive, Issued and Associated. If a ComID is in an Issued or Associated state, it is considered Active. After `GET_COMID`, a ComID becomes Issued, and if a session is opened under it, it becomes Associated. An associated state becomes Inactive again once all sessions opened under it have been closed. The Opal standard does not require dynamically allocated ComIDs to be supported. In Opal, statically allocated ComIDs are always Active.

3.3.2 Packetization

The communication in the last two layers is based on packets. There are three packet types that are nested, each packet type providing a different service. ComPacket is the outermost one. Each ComPacket contains data for communication under only a single ComID. Each ComPacket can contain several Packets. The usage of Packet is intended for session reliability, specifying sequential numbers and the acknowledgements for them. Every Packet then consists of one or more SubPackets. Finally, SubPacket may be either of Data type or Credit Control type. Data SubPacket contains one or more method calls or results of method calls. Credit Control Subpackets may be used for buffer management to inform the other communication party about the remaining buffer amount.

3.3.3 Methods

The commands of the lower layers to TPer and SPs and their respective responses are transported by the Data SubPackets in the form of methods. These are expressed using a set of lexical tokens in a specific order, described by Figure 3.1. Other than the constant tokens such as call

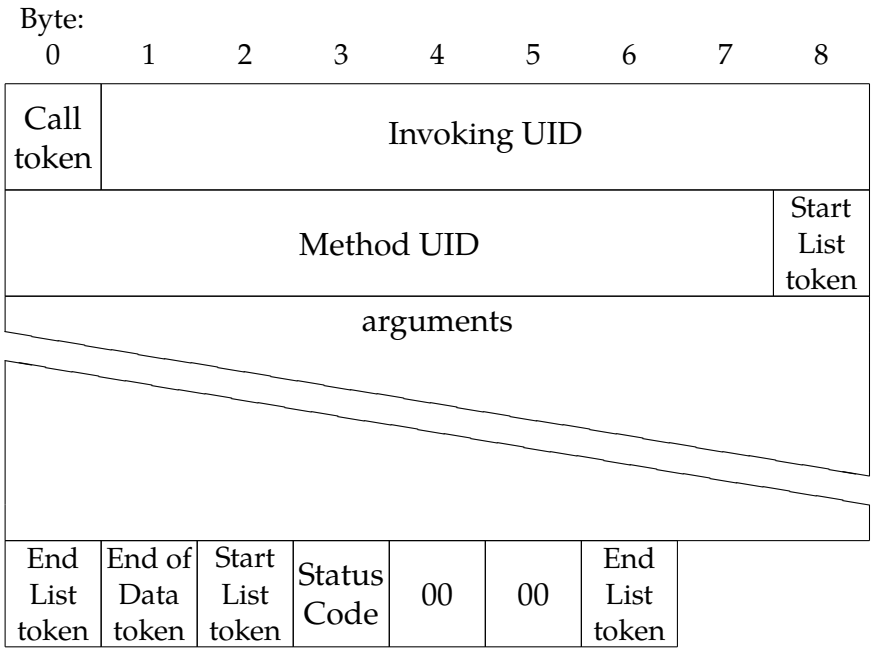

Figure 3.1: Format of method invocation

tokens or end list tokens, each method invocation contains Invoking UID, Method UID, arguments, and status code list. Invoking UID is used to identify which structure the method is invoked on. Other than tables and rows of tables, there are also special Invoking UIDs which can be used. These special Invoking UIDs are ThisSP, signifying the SP with which the current session is open, and SMUID, signifying the session manager. Method UID specifies the method to be called. Since SMUID methods are used for managing sessions, when the host is not connected to any of the SPs, SMUID methods cannot be found in the MethodID table of an SP. Each method specifies its type of Invoking UID, whether it is a table, an object, or any of the special UIDs, and its parameters. There are two kinds of parameters: the required parameter, identified by its position, and the optional parameter, used in a structure using an integer to identify the parameter, followed by the data of the parameter.

There are two types of arguments: required arguments, which all have to occur in the specified order, and optional arguments, which

can be left out. The optional arguments are identified by a preceding unsigned integer.

The last 5 bytes of the Method invocation are called Status List. It is a list consisting of a Status Code and two reserved bytes defined to be 0 bytes. A non-zero Status Code corresponds to an error response such as insufficient space, frozen SP, or usage of a method without sufficient authority.

A structure similar to method invocation is also used for a response. Since it is a direct response to the invoked method, it does not contain a call token, followed by the Invoking UID and Method UID. However, the TPer can also respond with a different method, such as in case it needs to close the session. In those cases, the same structure as for method invocation is used to also identify the function.

3.3.4 Sessions

Whereas the earlier introduced ComID is used to differentiate the senders of the method and the recipients of the method responses, sessions are used for the parallelisation of communication of one such actor. Each session can have different authorised authorities and methods in the process, but each session is still bound to one ComID. Since each ComID corresponds to one actor, both the actor and the TPer can send multiple methods calls or responses in one ComPacket. Sessions use a system of readers-writer locks to enable several concurrently running read sessions without causing concurrency issues. Each session is identified by its session number. The session number consists of the TPer session number selected by the TPer and the host session number selected by the host.

There are two types of sessions: control session, used for session management, and regular session. Since the control session is used to establish a regular session with an SP, it is not connected to any SP. The control session is used for managing the regular sessions and, as such, provides methods such as:

- *Properties* method enables one to find a compromise between the communication capabilities of the host and the TPer or simply find the communication capabilities of the TPer. Using this method, the properties defined earlier in Section 3.6, such as maximum packet size, are established.

- *StartSession* and *SyncSession* methods provide a way to start an unencrypted session.
- *StartTrustedSession* and *SyncTrustedSession* methods provide a way to complete challenge-response authentication or to set up secure messaging/key exchange.
- *CloseSession* method used to close the session.

Each ComID has one associated control session with a lifespan the same as the ComID since a control session is required to create a regular session. On the other hand, there can be multiple regular sessions.

Depending on the authorities used, it is also possible to start a secure messaging session, which provides message authenticity and/or confidentiality. Authority's operation type decides what kind of trusted messaging is to be used. These are further described in the Subsection 3.4.1. However, Opal does not mandate the support of secure messaging.

3.3.5 Transactions

In order to facilitate a safe execution of sequences of methods, the Core standard specifies transactions. Similarly to transactions in database systems, this feature enables one to revert the effects of a sequence of methods. This is done automatically in case the transaction is not finished or if the transaction is manually aborted. However, not all the effects of methods are rolled back, such as logs. Nested transactions are supported, in which case the transaction is committed when the outermost transaction is finished.

3.3.6 Possible limitations

Due to the range of possible values of the parameters, it is very likely that many features shown in this chapter might be unavailable in a specific device. For example, Opal specifies that the device must be able to handle at least one method per SubPacket, at least one SubPacket per Packet, and at least one Packet per ComPacket. This means that Opal-compliant do not necessarily need to be able to process multiple

methods in one ComPacket. Opal also specifies that at least one active transaction per session, at least one active session per all the ComIDs, and at least one static ComID are supported, making only one active transaction at a time possible. Because of this, many of the structures meant to support features such as asynchronous communication only introduce unnecessary overhead.

3.4 Access Control

In order to provide access to methods only to authorised actors, the standard also defines access control. The access control provides a way to allow access to methods of the SP only after the knowledge of a secret has been proved. Information required to verify knowledge of the secret is called a credential and is stored in one of the corresponding SP's credentials tables. Each SP may contain several credential tables, one for each type of credential, such as a password, an RSA key pair, or an AES key.

In order to support the authentication of different users or require multiple users at once, access control rules are defined using *Access Control Lists* (ACL) containing *Access Control Elements* (ACE). ACEs are Boolean expressions with inputs being authentication of authorities.

Each combination of method and object has a defined ACL, which needs to be satisfied in order for the method to be invoked on the object. An ACL is satisfied once one of the ACEs contained within it is satisfied, where authenticated authority variables evaluate as true.

Other than explicit authentication, implicit authentication is also possible. Implicit authentication may be used with, e.g. the Exchange operation type where the session key by the SP is encrypted using a pre-shared key and returned to the host.

Authority may be authenticated either during session startup using parameters of the SessionStart method or, if there is more than one authority, the additional authorities may be authenticated using Authenticate method. The Authenticate method authenticates only using explicit methods. In case Authenticate method is used to authenticate using a password, one invocation of the method with a password as the parameter is enough. In the case of challenge and response operation type (Sign, SymK, HMAC), the first invocation

only specifies the authority to be authenticated and receives challenge in response, and the second invocation provides the response.

Each authority is an object in the authority table of the corresponding SP and has a type which is either individual or class. Class authorities correspond to a set of authorities so that they can be easily assigned and changed in bulk.

3.4.1 Authorities

Authorities are constructs used to represent an actor. Authorities are saved in the Authority table of every SP, provided by the Base Template, and so are separate for each of the SP. Each authority is associated with one credential. This credential is an object from a table, depending on the type of the credential.

The Base Template defines several optional authorities such as the Admin authorities, representing the owner of the SP, the Makers authorities, representing the manufacturer of the TPer, the Security Identifier authority (SID), representing the owner of the TPer, and TPer authorities, representing the TPer itself. The Base Template also defines a special authority Anybody without any credentials. This authority can be used in reading public info of the SPs or to take ownership of the TPer.

Each authority has assigned one operation type. This operation type determines how this authority can be used to authenticate. Some of these types are Sign, used to authenticate using asymmetric cryptography, HMAC, used to authenticate using challenge and response with HMAC, and Exchange, used to share session key encrypted using the authority's credential.

However, the only operation type mandated by Opal to be implemented is the Password type, allowing one to get explicitly authenticated using a password. This means that any other way of authentication, such as implicit authentication and subsequently also secure messaging, may not be available.

3.5 Life cycle

In order to keep information about the state and the working capacity of an SP, the concept life cycle is introduced. The life cycle describes the condition of an SP using one of several states. In the Core standard [12], the following states are introduced:

- *Nonexistent* state describes SP which does not exist. The SP might not have been created or is already deleted.
- *Issued* state describes SP in functional state.
- *Issued-Disabled* state describes SP which can only be authenticated to and enabled. All other functionality is disabled. An SP can be enabled and disabled using the SPInfo table of the corresponding SP.
- *Issued-Frozen* state describes SP with no functionality enabled. An SP can be frozen and unfrozen using the SP table of the Admin SP.
- *Issued-Disabled-Frozen* state describes SP which is both disabled and frozen as described in the two previous items.
- *Issued-Failed* state describes SP after a fatal failure from which it cannot recover.

3.5.1 SP Issuance

SP issuance is the process of creating a new SP from the Base Template and a set of other templates. This is achieved by calling the Admin SP's method IssueSP. After an SP is issued, it can be personalised. Personalisation of an SP is the process of creating new tables (using the method CreateTable), filling in existing tables (using methods CreateRow and Set), and setting up access control (using the method AddACE).

3.5.2 Life cycles in Opal

Opal standard [11] extends the states defined in the Core specification with a new set of life cycle states: Manufactured-Inactive, Manufactured, Manufactured-Failed, Manufactured-Disabled, Manufactured-Disabled-Frozen, Manufactured-Disabled. Each of these new states mirrors the corresponding "issued" state from the Core specification. Compared to the "issued" states, the "manufactured" states are created during manufacturing by the manufacturer instead of during the subsequent use by the TPer owner. This is reflected by the fact that these manufactured SPs are not issued and deleted in Opal. Instead, they are activated and reverted. Activation automates the personalisation using hardcoded, preconfigured values. Reverting SP maintains its existence, returning it to the factory state instead. Note that this means that the Manufactured-Inactive corresponds to the Nonexistent state.

3.5.3 Taking ownership

In order to take ownership of an Opal drive, there are several approaches defined in the Opal standard [11]. The most frequent approach is for the device to store the default PIN of the SID authority, called MSID, in a table accessible without any credentials. This allows anyone to read the default PIN, authenticate themselves as the SID authority, and change the PIN. However, the process of taking ownership is not required in any way, and the drive will function as expected even if the PIN remains unchanged.

3.5.4 Feature sets

The system of authorities is further expanded upon by several feature sets. The following feature sets are mandated by the Opal standard.

Physical Presence SID (PSID) is a special authority that is authorised to call only the Revert method. The PIN for this authority may not be found using the interface of the disk. Every one of our disks supplied this information on a label on the disk. This authority provides a way to reset the TPer into factory state even in case the SID PIN is lost [19].

Block SID Authentication feature disables SID authority until the device is powered on again. One of the possible use cases is by BIOS to protect a drive with default MSID [17].

3.6 Capability discovery

In order to find out the properties and features of a particular device, there exists the so-called Discovery process. This process is divided into three levels, each with different reported information and a different approach to accessing the information.

3.6.1 Level 0 Discovery

Level 0 Discovery provides basic information about the secure device and is performed using special security command of the device. The information about the device is provided through feature descriptors. The presence of a feature descriptor header means that the feature is supported, and the fields of the header describe the basic properties of that feature.

Some of the frequent features described by these descriptors are:

- *TPer Feature Descriptor* reports supported communication features such as ACK/NACK support, ComID management, buffer management, or asynchronous communication.
- *Locking Feature Descriptor* reports information relevant to the Locking SP, such as support of data encryption, whether the disk is locked or the status of master boot record shadowing.
- *Geometry Feature Descriptor* reports information about the geometry of the disk, such as block size or alignment granularity.
- *Single User Mode Feature Descriptor* reports information about Single User Mode Feature Set, such as whether it is activated, or how many locking ranges can be activated under this mode at once.
- *DataStore Feature Descriptor* reports information about DataStore Feature Set, such as the maximum size of the DataStore table or maximum count of DataStore tables.

- *Opal 2.0 Feature Descriptor* reports information specific to Opal SSC, such as the base ComID, number of ComIDs, or the maximum number of users and admins authorities of Locking SP.

3.6.2 Level 1 Discovery

Level 1 Discovery provides information about the capabilities of the communication channel and is performed using the `Properties` control session method. The Level 1 Discovery is used not only as a way for the host to find out the capabilities of the TPer, but to determine shared limits depending on the capabilities of both the TPer and the host, as the host can also communicate its capabilities to the TPer.

The reported properties that are mandatory for Opal are:

- `MaxMethods` property describes the maximum number of methods per received SubPacket.
- `MaxSubpackets` property describes the maximum number of SubPacket per received Packet.
- `MaxPacketSize` property describes the maximum size of a received Packet.
- `MaxPackets` property describes the maximum number of Packets per received ComPacket.
- `MaxComPacketSize` property describes the maximum size of a received ComPacket.
- `MaxResponseComPacketSize` property describes the maximum size of sent ComPacket.
- `MaxSessions` property describes the maximum number of active sessions.
- `MaxIndTokenSize` property describes the maximum size of an individual token.
- `MaxAuthentications` property describes the maximum possible authenticated authorities.

- `MaxTransactionLimit` property describes the maximum number of active transactions.
- `DefSessionTimeout` property describes the default length of session timeout in milliseconds.

There are also defined properties which are not required in Opal, as they describe features that are not mandated:

- `MaxReadSessions` property describes the maximum number of reader sessions.
- `MaxAggTokenSize` property describes the maximum size of a combined token.
- `MinSessionTimeout` and `MaxSessionTimeout` properties describe the minimum and maximum possible session timeout, respectively.
- `DefTransTimeout`, `MaxTransTimeout` and `MaxTransTimeout` properties describe the default, minimum possible and maximum possible transaction timeout.
- `MaxComIDTime` property describes the maximum timeout for ComID allocation.
- `ContinuedTokens` property describes the support of continued tokens (tokens with size unknown at their beginning consist of several tokens, each carrying part of the final value).
- `SequenceNumbers` property describes the support of packet numbering (in Packet).
- `AckNak` property describes the support of ACK/NAK (in Packet) in communication.
- `Asynchronous` property describes the support of asynchronous communication.

Other than the defined properties, the Level 1 Discovery process may also report vendor-specific properties.

3.6.3 Level 2 Discovery

Level 2 Discovery is the act of reading a table using the Get method of an SP. This includes reading any table, such as the access control table or table of locking ranges. Some of the SPs' important tables are described in later Subsection 3.8.2 and Subsection 3.9.2.

3.7 Templates

Since SPs may share some of their functionality, such as authentication, modification of tables, or retrieval of data from tables, there exist templates to define this shared functionality. Each of the SPs then implements a subset of the functionality of one or more templates. The Core [12] standard defines the following templates:

- *Base Template* defines the shared subset required by every SP and is therefore mandatory. The most important functionality that is defined by this template is access control and metadata.
- *Admin Template* is a template specific to the unique Admin SP. It provides access to methods managing the TPer.
- *Crypto Template* provides methods providing cryptographic methods, such as encryption, hashing, or signing.
- *Locking Template* is a template specific to the unique Locking SP. Provides access to methods managing the lock state of the device and locking ranges.
- *Clock Template* provides methods for indicating current time, measuring lag, and accessing to monotonic counter.
- *Log Template* provides methods for logging activity. The logging can be either carried out manually by a user (or a user application) or automatically by the TPer (as a result of invocations of methods of SPs containing this template, including invocations during read-only sessions). Only one system log table may exist on each SP, but multiple user log tables may be present on one SP. The logs are saved on non-volatile storage. Depending on the

security setting of the table, they may be either buffered or saved after each entry.

Even though the Core standard defines several templates, Opal does not mandate them. Due to the fact that issuance of SPs is optional in Opal and that the only mandatory SPs are the Admin SP and Locking SP, Opal devices need to implement only the Base Template, Admin Template and Locking Template.

3.8 Admin SP

Admin SP is one of the two SPs mandated by the Opal standard [11]. It is a unique SP which holds information about the TPer and all the present SPs. It allows the creation, deletion and general modification of the SPs. It is created from the Base and the Admin Template. This SP is initialised in the "manufactured" state so that it can function as a starting point of TPer management.

The description of methods and tables that follows is specific for Opal SSC and may be different for other SSCs.

3.8.1 Methods

Even though the templates from which the Admin SP consists define many more methods, the Opal standard mandates only some.

- Base Template methods:
  - *Next* method returns UIDs of objects in the table. Allows specifying the returned amount and the first UID.
  - *GetACL* method returns the ACL of the method and object or method and table combination.
  - *Get* method returns the value of the object or the contents of the byte table.
  - *Set* method sets the value of the object or the contents of the byte table.
  - *Authenticate* method performs an additional explicit authentication.

- Admin Template methods:
  - *Revert* method reverts the SP referred to by the invoking ID to its initial state.
  - *Activate* method activates the SP referred to by the invoking ID.
- Crypto Template methods:
  - *Random* method returns the specified amount of random bytes.

3.8.2 Tables

Similarly to the methods, the Opal standard does also not require every table defined by the template to be implemented. Instead, only a selected few are required.

- Base Template tables:
  - *SPInfo* table contains metadata about the SP.
  - *SPTemplates* contains templates used by the SP.
  - *Table* table contains metadata about tables used by the SP.
  - *MethodID* table contains metadata about methods used by the SP.
  - *AccessControl* table contains ACLs for combinations of methods and tables.
  - *ACE* table contains definitions of ACEs as described in Section 3.4.
  - *Authority* table contains list of authorities.
  - *C_PIN* table contains password credentials of authorities.
- Admin Template tables:
  - *TPerInfo* table metadata about the TPer, versions, implemented SSCs, free space for issuance, ProgrammaticResetEnable

  - *Template* table metadata about templates, number of instances
  - *SP* table metadata about SPs, current life cycle, frozen status

- Opal additional tables:
  - *DataRemovalMechansim* table describes the mechanism for data removal, such as overwrite erase, block erase, or cryptographic erase

3.9 Locking SP

Locking SP is the second SP mandated by the Opal standard [11]. It is a unique SP, which takes care of disk encryption, locking, unlocking, and tasks related to it.

The description of methods and tables that follows is specific for Opal SSC and may be different for other SSCs.

3.9.1 Methods

Locking SP shares the Base Template and Crypto Template methods with Admin SP and also adds Locking Template methods.

- Locking Template methods:
  - *GenKey* method re-generates a key or a credential. But, in the case of Opal, this method is mandated to be able to be used at least on the `K_AES_*` DEK keys.
  - *RevertSP* method reverts the Locking SP. Compared to the method Revert, this method can revert only the SP to which the session is connected, allowing reverting without having access to Admin SP's Admin authority.

3.9.2 Tables

Locking SP shared the Base Template and Opal-specific tables with Admin SP, but it also extends the methods offered by the Base Template and adds Locking Template tables.

- Extra Base Template table:
  - *SecretProtect* table specifies protection mechanisms for secrets, such as DEKs in the case of Opal. Possible mechanisms are binding to C_PIN values or vendor specific [23].
- Locking Template tables:
  - *LockingInfo* table metadata for locking mechanisms, such as the maximum number of locking ranges or supported types of encryption.
  - *Locking* table contains the state of locking ranges, more closely described in Subsection 3.9.3.
  - *MBRControl* table contains metadata for MBR shadowing (Enable, Done, DoneOnReset).
  - *MBR* byte table containing the shadow MBR.
  - *K_AES_128* and *K_AES_256* tables contain the DEKs for each of the locking ranges. It can be accessed only by using the GenKey method.
- Non-template tables:
  - *DataStore* byte table for storage of arbitrary data [16].

3.9.3 Locking range

The *locking range* (LR) feature gives the user a way to specify an LBA range on the disk that can be locked independently on the rest of the disk. Each locking range also has its own ACE to control who can lock and unlock the range. Because each LR is also encrypted using its own DEK, it is possible to re-generate a new DEK for a single LR, discarding data of only one LR.

There also always exists a special range called the global locking range that covers any area on the disk that is not already covered by a regular locking range.

In order to control the access to the data on a disk covered by a specific locking range, the Locking table contains columns ReadLockEnabled, WriteLockEnabled, ReadLocked, and WriteLocked. The

data inside the LR cannot be read when ReadLockEnabled and ReadLocked are both set to true. Otherwise, the LR range is locked, and the data cannot be read. Similarly, for writing and WriteLockEnabled and WriteLocked. The separation of the locking right into the LockEnabled and Locked column gives the ability to have separate ACE for the LockEnabled column, meaning that, for example, an admin can decide that the user can only lock reading for a specific LR but not writing.

Each locking range also has a defined column, LockOnReset containing a list of types of resets on which the LR gets locked. For Opal devices, reset Power Cycle must always be selected, and Programmatic reset can be selected additionally. The last reset type Opal defines for the drives is the optional Hardware Reset. All of these resets are defined in the TCG SIIS specification [21], includes a mapping of each of those errors to a particular event depending on the disk interface.

3.9.4 Single User Mode

In many cases, it might be desirable to prevent admins from accessing user data, even though the admins generally have more competence. The Single User Mode Feature Set [20] defines a way to achieve this using a *Single User Mode*. After activating Single User Mode, only a single User authority is capable of changing their authority object (this includes changing their PIN), changing the proprieties of LRs assigned to them (which includes the lock state of the LR) and generating new keys for those LRs. The only actions available to the Admin authorities related to that LR are destructive actions.

3.9.5 MBR shadowing

This feature enables the disk to provide a fake master boot record (MBR). Instead of the MBR saved on the disk, the disk instead provides the shadow MBR saved in a table on bootup. The shadow MBR may contain software to enable the host to authenticate itself to the disk and unlock it. After the host is authenticated, the shadow MBR may be deactivated, and the regular disk data will be available again.

This feature is controlled using tables `MBR`, a byte table containing the data to be presented while the shadowing is active, and `ControlMBR`,

an object table used to control the shadowing. Table `ControlMBR` one row with columns `Enable`, `Done`, and `MBRDoneOnReset`. When `Enable` is true and `Done` is false, the shadowing is active, and the data from `MBR` table can be read from the disk. The column `MBRDoneOnReset` specifies a list of types of resets during which the `Done` is set to false.

4 Opal interface in Linux

The Linux kernel offers two ways to control an Opal device from the host. Both ways use the Linux ioctl system call. In this section, we will introduce these two approaches.

4.1 Opal ioctl

Since version 4.11, the Linux kernel offers a set of ioctl requests to facilitate control over an Opal disk [24]. The individual implemented ioctl requests for Opal functionality and their function as of Linux kernel v6.1 are:

- `SAVE` command keeps a key for a locking range in the host's RAM so that it can be used to unlock the disk after waking from suspension. Exported symbol `opal_unlock_from_suspend` can be used to unlock the disk using the saved data. The command can be repeated to save keys for multiple locking ranges.

- `LOCK_UNLOCK` command locks or unlocks reading or writing for the selected locking range.

- `TAKE_OWNERSHIP` command takes ownership over the device. In other words, it changes the admin authority password from the default MSID to the selected one.

- `ACTIVATE_LSP` command changes state from "Manufactured-Inactive" to "Manufactured" state. Additionally, it can also facilitate the setup of Single User Mode.

- `SET_PW` command changes the password of the selected Locking SP authority using the admin password.

- `ACTIVATE_USR` command enables the specified user in the Opal tables.

- `REVERT_TPR` command reverts the TPer to the manufactured state using the SID authority.

- `LR_SETUP` command sets the position of LR and enables or disables the locking of reading or writing for that LR.
- `ADD_USR_TO_LR` command sets the ACE for locking and unlocking to the designated user. Contrary to the name, this request does not actually add a user and instead replaces any existing one with the new one.
- `ENABLE_DISABLE_MBR` command changes the `Enable` column of the `MBRControl` table.
- `ERASE_LR` command calls the erase method defined in Single User Mode Feature Set [20]. This method not only destroys the data in the locking range but also resets the assigned user authority of the locking range and the user's password. This is required since a Single User Mode protects these values from change by authorities other than the assigned user.
- `SECURE_ERASE_LR` command re-generate the data encryption key of a range to destroy the previous one.
- `PSID_REVERT_TPR` command resets the TPer to the manufactured state using PSID.
- `MBR_DONE` command changes `Done` column of the `MBRControl`.
- `WRITE_SHADOW_MBR` command writes data into the MBR table.
- `GENERIC_TABLE_RW` command reads a byte table or writes it into a byte table. Neither object tables nor iteration is supported.
- `GET_STATUS` command returns information about the disk, such as it's support of Opal SSC, status of MBR shadowing, or whether the Locking SP is activated.

Although these ioctl requests offer simple access to control the Opal capabilities of a disk, not every feature mandated by Opal is implemented. Some of the limitations are the following:

- Access to only a single admin authority and up to 9 user authorities.

- No way to change the password of the SID authority. This means that the authority representing the owner of the device, which often has control over the entire disk, is stuck with the credentials chosen during the taking of ownership.
- No write-only lock.
- Each locking range can be configured with only a single user capable of locking and unlocking it.
- No way to read or write to object table rows or iterating tables. This prevents the possibility of replacing the missing features using a direct setting of values in a table.

Currently, there is no documentation to be found for the ioctl requests for Opal functionality. The information in the previous list was acquired from our code analysis.

4.2 Drive interface ioctl

Alternative to the Opal ioctl requests are the disk controller ioctl requests. Depending on the disk protocol, a different way of passing the Opal commands is required. The TCG SIIS specification [21] describes the mapping of the abstract IF-RECV and IF-SEND commands to the actual commands of disk interfaces for different drive interfaces.

For NVMe drives, there exists the `NVME_IOCTL_ADMIN_CMD` ioctl request to send commands to the disk, which uses the `nvme_admin_cmd` structure [25]. The IF-RECV and IF-SEND are mapped to the Security Receive and Security Send NVMe's command, respectively [21]. The structure specifies the ComID, and the data to transmit to or from the TPer.

For ATA drives, the situation is more complex. There does not exist an ATA command we could use to send the Opal commands. However, using the SCSI `SG_IO` ioctl request with its `sg_io_hdr_t` structure, we are able to use the ATA PASS-THROUGH command [26]. Using the ATA PASS-THROUGH command, we are able to send arbitrary ATA commands to the device, including the TRUSTED RECEIVE and TRUSTED SEND commands [5] mapped to IF-RECV and IF-SEND respectively.

Using these structures, the Opal commands described in Section 3.3 can be sent to the TPer. Compared to the Opal ioctl requests, this has the advantage of not being limited only to a subset of features that the Opal ioctl requests implement and instead being able to use every Opal feature the device offers.

Although this approach gives the host application access to every feature of the Opal disk, it also requires it not only to implement the command hand-over for each type of disk separately but also to create the methods and parse the method results, both described in the Chapter 3.3, on their own.

5 Security of hardware encryption

Disk encryption is usually not meant to protect the content of the device while it is being actively used. For example, the Opal standard aims to only "Protect the confidentiality of stored user data against unauthorized access once it leaves the owner's control (following a power cycle and subsequent deauthentication)" [11]. Even if the device is successfully protected against such a threat model, it still leaves many possible attack vectors. To extend the threat model into a more realistic scenario, we consider an attacker that has physical access to the disk installed in a "locked" computer. We consider a computer to be "locked" if it is either turned off completely or protected by a lock screen or similar way of preventing unauthorized access to the computer.

In this chapter, we will provide an overview of state-of-the-art attacks relevant to hardware disk encryption. For each attack, we will provide our theoretical analysis of the protection offered by Opal, if properly implemented in a disk, against such an attack. For a selection of the attacks, we will also provide our practical insight. The set of the disks we have tested is made from SanDisk X300s, Kingston KC600, Samsung SSD 850 PRO, Samsung SSD 860 EVO and Samsung SSD 980 disks.

5.1 Evil maid attack

An evil maid attack [27, 28] consists of a situation where the device is left unattended for some time, during which the attacker has full physical access. Such attacks often have two phases. In the first phase, the disk is modified, e.g. by installing custom firmware or inserting a probe to eavesdrop on the communication. Then, the victim uses the disk, providing the password to unlock it. In the second phase, the attacker returns to collect the obtained information and undoes their changes. In some cases, the attacker might not require physical access to the device and instead use a network to receive information and have, e.g. the modified firmware reverse itself or simply leave traces if the discovery of the attack is not a problem.

Impact on Opal devices

For the two phase type of the attack, the Opal disk itself has protection against unauthorized changes. The shadow MBR requires authentication to be changed, and although newer versions of Opal do not mandate an interface for firmware update anymore, it is standard practice to require a newer signed firmware to update. But eavesdropping the communication between the host and the drive is still possible, as Opal does not mandate secure messaging, and therefore any encrypted communication, it is still a real possibility to insert a probe in between the disk and the computer.

In general, there is not much protection against evil maid that does not care about detection, as they need to only fake that the system is real until the password is acquired. They can replace the victim's machine with one that is similar enough until that time. This means that the disk itself can also be replaced, most likely circumventing any protection against such an attack on the disk. Although physical security is the obvious solution, another approach is to let the disk verify itself first. Once again, the Core standard offers a way for the disk to authenticate itself to the host, but the Opal standard does not mandate that it will be supported.

An encrypted communication would be able to prevent the password from being sniffed by a probe between the disk and the host. For this purpose, the Core standard defines the secure messaging feature, which provides confidentiality of the communication. An authentication of the disk to the host would be able to prevent the replacement of the disk. The core standard introduces this feature through different types of authority operations, such as challenge and response. However, the Opal standard does not mandate secure messaging and the only authority operation it does mandate is password, and so it is unlikely that any Opal device would implement them.

5.2 Attack on key generation

Attack on key generation are such attacks which abuse the RNG generating data with low entropy or even entirely predictable. Due to the decreased entropy, it may be possible to potentially predict the value of the DEK. Since Opal specification does not actually specify any

implementation for the RNG, we cannot perform any fruitful analysis of the standard itself. However, we can still try to perform practical analysis.

Impact on Opal devices

The Opal specification offers a few ways how to generate random data. The most useful would be GenKey, as it is used for generating DEKs, but since the destination cells, where the result is recorded, are protected from reading, we cannot easily read them to get the generated values. Another method of generating random data is the method Random. This method allows us to get at least 32 bytes [11] every invocation. Even though the RNG used by this method might be different to the one used by the GenKey method to generate DEKs and has low performance (our slowest tested disk was able to generate about 500 bytes per second), the Random method provides easy access to the generated data. To test the randomness of the RNG of the disks, we have decided to use the Random method and the *dieharder*[1] random number generator tester. From the disks we have inspected, we have found the following about their RNG:

- *SanDisk X300s* generates always the same bytes for the first Random invocation of a session. The value of the first bytes seems to change only after the new activation of Locking SP. Following invocations of the Random method in a single session return different bytes. This weakness does seem to also affect the key generation. We tested this by zeroing out several blocks of the drive and regenerating the DEK. Each time we re-generated the key the drive content was different. However, we have decided to also test the randomness of the content. If we zeroed out the content before the key regeneration, the content did not pass the test of randomness. If we, however, wrote random data before the key regeneration, the content did pass the randomness test. This property was different from every other disk, and may suggest that the key generation is truly not entirely random. Looking closer at the contents of the disk generated this way, it is possible to see several patterns described more closely in Appendix B.

1. `https://webhome.phy.duke.edu/~rgb/General/dieharder.php`

- *Kingston KC600* generates random data that passes most dieharder tests, even though the Random method requires extra consideration while it is being invoked. The Random method of this does not actually accept parameters as atoms but as unsigned integers. However, this does not cause difference in behaviour as long as the value of the atom is small enough.

- *Samsung SSD 850 PRO, Samsung SSD 860 EVO and Samsung SSD 980* generate random data that passes most of dieharder tests.

As the generation of random bytes takes a considerable amount of time, we have generated only a limited number of bytes for the test. This is most likely the cause of some tests not passing, as they have reused the input several times.

5.3 Cold boot attack

In this attack, a property of volatile memories is used. Even though volatile memories lose the data stored in them after powering off, they do not lose the data instantly. This introduces the cold boot attack, which takes advantage of this fact. One of the ways this attack can be implemented is to take a RAM stick from the victim's locked computer and install it into the attacker's computer, where it can be read. Another possible approach is the reboot the victim's computer into the attacker's system.

Impact on Opal devices

Even though this attack works quite well against software-based disk encryption and software encryption in general, against purely SED, it does not work that effectively. This is because, compared to software encryption, there is no need to store the encryption key in the computer's memory. The key is instead stored in the memory of the disk controller, where it can be physically protected against removal. However, this advantage holds only for the case where the key or the disk password is truly not kept also in the computer's memory, which may not always be the case with SEDs. In the Linux Opal ioctl commands introduced in an earlier Section 4.1, there is introduced

functionality to save the password for locking range in the memory. This functionality is introduced in order to allow the computer to wake from suspension automatically since the disk might need to be unlocked in such a case. But since the key is stored in the computer's memory, it is possible to acquire it using the cold boot attack.

5.4 Hot plug attack

Hot plug attacks [27] are attacks similar to cold boot attacks. However, compared to the cold boot attacks, where the RAM is inserted into the attacker's system before the data in it is naturally destroyed, hot plug attacks instead move the disk into the attacker's system. This attack abuses the fact that SATA disks often have a data cable and a separate power cable. Because of this, it is possible to switch the data cable into a different host device without powering the disk off and therefore locking it.

However, this approach is certainly not limited only to SATA disks with separate data and power cables, as it is possible to translate the attack to many different physical disk interfaces. An attack called hot unplug attack [29] generalized this attack. Before unplugging the original cable with the power pins, substitute power pins can be connected together with the original ones, keeping the disk device powered while the original cable gets unplugged.

Impact on Opal devices

The Core standard introduces HotPlug as one of the reset types. This event might seem like a possible solution to such attacks. However, its meaning is not defined in the standards. TCG Storage Interface Interactions Specification [21], the standard defining, among other things, the mapping of reset types to disk interface events, does not mention the HotPlug reset type. None of the disks we have tested supported this reset type, so we are left to only speculate about its possible efficacy.

5.5 DMA attack

These attacks use the DMA interface, such as Thunderbolt or PCIe, to read the DEK from the host's memory or to circumvent the lock screen. As long as the DEK is not in the host's memory, the first approach can be prevented. Although, as we mentioned in Section 5.3, this might not always be the case, even with SEDs. The second approach can be hard to prevent on the SED side, However, modern systems already have implemented protections against such attacks, so the risk of such an attack might already be minimized.

5.6 Vulnerable interface

An attack on the interface is such an attack where problems in the interface are taken advantage of. This might include attacks such as using vendor commands to read the disk directly, even if the disk is locked. In the past [6], some hardware-encrypted disks were vulnerable to such attacks, allowing the usage of undocumented vendor commands to change the firmware of the device to a custom-made one.

Impact on Opal devices

As can be seen from the previous Subsection 5.2, the implementation of the interface is not perfect for some of our tested disks. Even though issues like the disk expecting a different type of format are not directly connected to security in our case, such problems might be a sign of other issues possibly related to security.

This might be especially problem if such malfunctioning interface gives its user false sense of confidence. While testing our disks, we found out several interface flaws with Kingston KC600. One of such flaws is the fact that the disk accepts any value as reset type. While specifying types of resets that should lock a specific interface, it is possible to specify any type, not only types that are not supported, but also types which are not even defined in the standards. This means that after "successfully" setting LR to be reset after a hot plug event, it might never actually be once such event happens.

6 Existing tools

In order to give users the ability to manage their SED disks without the need to create their own programs to access existing interfaces, there exist several tools implementing the protocols introduced in Chapter 2 and Chapter 3.

Note that some of the the tools use ATA TRUSTED commands for managing ATA Opal devices. When using these commands, it is necessary to first set the `allow_tpm`[1] flag of the kernel libATA library to allow Trusted Computing commands [5]. This can be done either by specifying it as a kernel flag `libata.allow_tpm=1`[2].

In this chapter, we will look at several existing popular open-source tools that allow one to manage the encryption of disks.

6.1 hdparm

The *hdparm*[3] tool provides a command line interface to control the parameters of ATA disks. Among others, this tool gives access to control ATA Security Feature Set. The examples of SED-relevant commands for the functionality described in Subsection 2.1.1 are:

- Enable the ATA Security Feature Set:
  `hdparm --security-set-pass pwd /dev/sda`
- Disable ATA Security Feature Set:
  `hdparm --security-disable pwd /dev/sda`
- Unlock the disk:
  `hdparm --security-unlock pwd /dev/sda`
- Erase the disk:
  `hdparm --security-erase pwd /dev/sda`

---

1. The usage of TPM in the name appears to be purely a historical blunder as it does not seem to be ever used for TPM, a cryptoprocessor standard defined by TCG, the same group defining TCG Storage standards.
2. `https://wiki.archlinux.org/title/Self-encrypting_drives`
3. `https://sourceforge.net/projects/hdparm`

Implicitly the user password is referred to in the commands, but `--user-master` can be specified to select master or user password explicitly. While enabling the ATA Security Feature Set, it is also possible to use the `--security-mode` option to choose between high and maximum security modes. A command to lock the drive is missing because a drive with an enabled ATA Security Feature Set is locked when powered off.

6.2 sedutil

sedutil[4] is a tool for the management of SED disks maintained by the Drive Trust Alliance. It currently supports the Enterprise and Opal SSCs (and additionally the remaining Pyrite, Opalite and Ruby SSCs in a fork of the project[5]) and NVMe and SATA interfaces.

Even though it is currently the most prominent open-source project to control SED disks, being exclusively used in guides for using SED on Linux, the code repository is inactive.

This tool does not use the Linux Opal ioctl interface and instead uses the approach described in Section 4.2. This allows the tool not only to be multi-platform but also to use the Opal features in their entirety instead of only the subset introduced in the Opal ioctl interface.

The tool offers the following commands, grouped by their functionality:

- Discovery:
  - `isValidSED` command lists SSCs supported by the selected disk, the name of the disk and its firmware version.
  - `scan` command performs `isValidSED` command on every disk in the system.
  - `query` command performs Level 0 and Level 1 Discovery on the selected disk.
  - `printDefaultPassword` command prints the MSID password of the selected disk.

4. `https://github.com/Drive-Trust-Alliance/sedutil`
5. `https://github.com/ChubbyAnt/sedutil`

- TPer management:
  - `initialSetup` command takes ownership of the disk, initializes the Locking SP and the global LR, and enables MBR.
  - `setSIDPassword` command changes the password of the SID authority.
  - `setAdmin1Pwd` command changes the password of the Locking SP's first admin authority.
  - `setPassword` command changes the password of any supported Locking SP's authority.
- Locking range management:
  - `listLockingRange` command prints information about a single LR, such as the start, length or lock status.
  - `listLockingRanges` command prints information about all LRs, similar to the previous command `listLockingRange`.
  - `rekeyLockingRange` command re-generates the selected LR's key, destroying the data in the LR in the process.
  - `setupLockingRange` command sets the start and length of the locking range.
  - `setLockingRange` command sets ReadLocked and WriteLock of the LR depending on the chosen configuration (read-write, read-only, or locked).
  - `enableLockingRange` command sets ReadLockEnabled and WriteLockEnabled of the LR.
  - `disableLockingRange` command unsets ReadLockEnabled and WriteLockEnabled of the LR.
- Shadow MBR management:
  - `setMBREnable` command sets Enable column of the MBRControl table.
  - `setMBRDone` command sets the Done column of the MBRControl table.

  - `loadPBAimage` command writes the contents of a file into the MBR table to be used as the shadow MBR.

- Reverting the device:
  - `revertTPer` command reverts the TPer using the SID authority.
  - `yesIreallywanttoERASEALLmydatausingthePSID` or the undocumented `PSIDrevert` command reverts the TPer to factory state using the PSID password.
  - `revertNoErase` command reverts only the Locking SP, keeping the global range data.

- Enterprise-specific functionality:
  - `setBandsEnabled` command enables the usage of all locking ranges.
  - `setBandEnabled` command enables the usage of selected locking range.
  - `eraseLockingRange` command cryptographically erases the data in a specified locking range.

6.3 go-tcg-storage

go-tcg-storage[6] is a Go library for managing TCG Storage disks, also providing CLI tools. Even though when compared to the previously introduced sedutil, it is not as popular, it is currently in active development. It supports not only Opal 2.0 and Enterprise SSCs but also Ruby, the successor of Enterprise. It also, compared to sedutil, supports, in addition, the SAS interface. Although it implements a CLI, its capabilities compared to sedutil are very limited, as can be seen from the following paragraphs. The CLI consists of three utilities: `sedlockctl`, `tcgdiskstat`, and `tcgsdiag`. For some of the tested disks, the utilities were not able to start a session due to time out and therefore function correctly.

6. `https://github.com/open-source-firmware/go-tcg-storage`

The `sedlockctl` utility allows one to control a SED disk. At the time of writing it offers the following commands:

- `list` command lists all LRs.
- `lock-all` command locks all LRs.
- `unlock-all` command unlocks all LRs.
- `mbrdone` command sets the MBRDone cell to the desired value.
- `read-mbr` command prints the shadow MBR.

The `tcgdiskstat` utility provides information about supported SSCs and firmware version for each disk, similar to the sedutil's `scan` command. Unlike sedutil's `scan`, it also prints whether encryption is supported, MBR is enabled, or SID authority is blocked.

The `tcgsdiag` utility performs Level 0 and Level 1 Discovery, similar to sedutil's `query` command. It also prints the MSID password, random numbers generated by the disk, and contents of the TPerInfo table.

6.4 TCGstorageAPI

Seagate's *TCGstorageAPI*[7] is a library that provides a Python interface and a CLI tool. Similarly to sedutil, it supports Opal and Enterprise SSCs. Compared to the previous tools, this one offers the most functionality, but this comes at the cost of also the largest dependencies. The tool has the most complete support for Enterprise SSC. The library also includes a command to test an implementation of Opal drive, although it is not as complete as the test cases published by TCG.

When used with a device implementing Opal 1 SSC, the CLI tool reported "SED configuration is Unknown/Unsupported (Type Opal) - Exiting Script", even though the device also implemented a supported Opal 2 SSC.

7. `https://github.com/Seagate/TCGstorageAPI`

6.5 sedcli

sedcli[8] is a tool for managing SED devices. Although the only supported disk interface is NVMe and the only supported SSC is Opal, it offers support for the Key Management Interoperability Protocol and key management servers.

6.6 fscrypt

Since inline encryption requires an active participation of the host during the encryption, a support in the kernel is required. This means that in order to provide inline encryption it is not enough to have a user space tool.

6.6.1 Kernel space library

In the upstream Linux, there exists a support for inline encryption. However, it is currently used only by the kernel space library *fscrypt*. This library works on a filesystem level, which allows it to be more flexible compared to block device-level encryption, as it can limit the encryption to only some files. However, fscrypt does not encrypt all information of the files, and the metadata (excluding the filename) are left unencrypted. Although by default fscrypt uses software encryption, it is possible to switch to inline encryption.

6.6.2 User space tool

fscrypt[9] is also the name of a user space tool for the management of Linux filesystem encryption. Although it does not provide explicit support for inline encryption, thanks to the kernel space library fscrypt providing inline encryption in a transparent way, it can still be used to manage file system encryption with inline encryption.

As of now, block layer inline encryption is supported only by two file systems in Linux: ext4 and F2FS. In the following paragraphs, we will focus only on the ext4 filesystem.

8. `https://github.com/sedcli/sedcli`
9. `https://github.com/google/fscrypt`

First of all, inline encryption needs to be enabled in the kernel configuration. This means that the Linux kernel configuration option `CONFIG_FS_ENCRYPTION_INLINE_CRYPT` needs to be enabled. In order to start using inline encryption on a file system, it needs to be mounted with the option to specify the usage of inline encryption.

```
mount -t ext4 /dev/foo /mnt/foo -o inlinecrypt
```

It is not enough to only specify the inline encryption flag. The encryption itself also must be enabled in the file system. On ext4 file systems, the encryption can be enabled after mounting like so:

```
tune2fs -O encrypt /dev/foo
```

After this, fscrypt can be used as usual, and it will use inline encryption for this filesystem.

7 Opal toolset

In order to be able to study the self-encrypting drives, specifically the most popular Opal drives, at a closer level, we have implemented a toolset for controlling and inspecting Opal devices. We have focused on implementation that would not require any additional tooling or libraries and would allow as much control over the communication with the Opal device. This approach allows the project to be not only used on a system without introducing a plethora of additional packages but also for our code to be used in other projects without needlessly increasing the number of dependencies.

7.1 Structure

The code base is split into three levels. Each level implements a different part of the functionality that depends on the previous levels.

The lowest level is focused abstraction of the communication primitives. This includes primarily functions to build a method to be sent to the device and parse the received response, generic functions to start a session or set a value in a row of a table, and the implementation of IF-SEND and IF-RECV commands abstracting ATA and NVMe system calls.

The middle level joins the lower-level methods into complete functions. Each function represents one of the methods and provides a C interface to invoking the method. Internally the functions start a session with the device, build the method according to the function's arguments, send the method to the device, retrieve the response, and parse the response, to find whether to command was successful and to potentially return the data to the caller.

Finally, the top level combines the two previous levels to create a usable front end usable through a command line.

7.2 Provided utilities

Currently, there are implemented three front ends: `rng` to produce random numbers generated by the selected Opal disk, the `control` to

```
# ./rng /dev/sdb 24 2
e993f67588190fc5d6d0ef7c7febf1944c566e211ba10352
29b0897bb4ddec1d71d89c95fc945a031574936c3c687477
```

Figure 7.1: Usage and output of the `rng` utility

manipulate the status of the selected Opal device, and the `discovery` to inspect the selected Opal device.

7.2.1 Random number generation

The `rng` is the most straightforward front end of the introduced three. Its only goal is to produce random numbers generated by the disk quickly. However, all the disks we have tested implemented only the minimal Opal requirements, which guarantee a limit of only one active session, packets containing only one method, and the Random method providing only 32 bytes at once. Such properties do not leave much space for optimisation. The throughput this utility can achieve with the disks tested by us is between 500 bytes per second and 35000 bytes per second, depending on the disk. Although to obtain a sufficient amount of data, a considerable amount of time is required, this utility may be used to collect random numbers for testing the randomness of the disk's RNG. The program repeatedly opens sessions, and during each session, it repeatedly sends the Random method. The total number of sessions and the number of bytes acquired during each session can be specified using the arguments as shown in Figure 7.1.

7.2.2 Device management

The `control` utility provides access to basic control over the device. The utility serves as a wrapper around the middle-level functions described earlier. As of right now, the utility provides access to the following commands:

- `psid_revert` command performs PSID revert using the Revert method as the PSID authority [19], resetting the device into the manufactured state.

```
# reset device to the manufactured state
./control psid_revert /dev/sda \
    --verify-pin $PSID
# take ownership over the device
./control setup_tper /dev/sda \
    --assign-pin 0000
# create a user
./control setup_user /dev/sda \
    --user 1 --verify-pin 0000 --assign-pin 1111
# create locking range
./control setup_range /dev/sda \
    --locking-range 1 --locking-range-start 0 \
    --locking-range-length 512 \
    --user 1 --verify-pin 0000
# lock reading of the locking range as the user
./control unlock /dev/sda \
    --user 1 --verify-pin 1111 \
    --locking-range 1 --read-locked 1
```

Figure 7.2: Usage of the `control` utility

- `setup_tper` command takes ownership of the TPer by setting the password of the SID authority to the chosen value.
- `setup_user` command enables Locking SP's user authority and sets its password.
- `setup_range` command enables the locking range and sets up its ReadLocked and WriteLocked ACL and its other parameters, such as the range of the locking range, ReadLockEnabled, and WriteLockEnabled.
- `unlock` command sets the ReadLocked and WriteLocked columns of a locking range, locking or unlocking the range for reading or writing.
- `reset` command sends the TPER_RESET command to the device [12].

An example of usage of the `control` utility can be seen in the Figure 7.2, where we present a basic use case scenario.

7.2.3 Disk properties discovery

The final front end, the `discovery` utility, provides information about the Opal disk. This utility provides, other than the Discovery information introduced in Section 3.6 also other useful information, such as the identify commands of the drive interfaces (Identify command for NVMe devices [25] and IDENTIFY DEVICE command for ATA devices [5]). Level 0 Discovery and Level 1 Discovery information are acquired easily, as they are defined as they may be acquired using a simple IF-RECV command with specific ComID or using the Parameters method without any parameters, respectively. The Level 2 Discovery, on the other hand, is more complicated.

For Level 2 Discovery, we would like to include as many tables as possible. The Opal standard defines the mandatory and optional tables, yet we would like to include also any table that is vendor specific. We also have a few limiting factors that may limit the number of tables we will be able to read. First of all, we do not want to change the state of the device. This includes actions such as reverting the

device to the manufactured state or taking ownership of it. Even if the state is potentially reversible without data loss, the risk of the disk not working correctly surpasses the potential gain from more information. This might notably affect the collection of information from the Locking SP if it is not activated Another requirement is that no authorisation should be required. Instead, we will use the password-less Anybody authority. This again restricts the set of tables that are accessible. However, according to the definitions of access rights in the Opal standard [11], this mainly affects secrets, such as passwords of Authorities or DEKs, and the current configuration of locking ranges, such as their start and length. Such information does not speak about the properties and capabilities of the device itself.

The process we use in order to print the contents of every table is as follows: Firstly we iterate using the Next method over the SP table of the Admin SP to get a list of existing SPs. As long as the Admin SP is not frozen or disabled, this should always succeed since the Admin SP is in the manufactured state by default, and its SP table is always accessible by the Anybody authority. With the list of SPs, we now may successively iterate using the Next method over the SPs and, for every one of them, get a list of their tables from the Table table, the same way we did with the SP table. The Table table is again always accessible by the Anybody authority. We then go through the found tables and try to get a list of their rows using the Next method. If this succeeds, we can then use the Get method to get each row of the table. However, some of the tables do not implement the Next method. This means that if we want to get the information from the rows of such tables, we have to guess the UIDs of the rows. Since each table can have up to 2^{32} sparsely distributed rows, simply iterating over all possible values is out of the question. In those cases, we only guess that a row with the lowest possible UID might exist and try to read information from it.

Even after finishing the crawling, there might still be some tables left that we have not found. Even though the Table table should identify the tables of the SP, from our experience, it often did not contain all of them. To solve this issue, we have a list of tables which are described in the Opal standard [11] but were not found in the Table table of some of our tested disks. We then process these tables separately.

During the process, we print the acquired information. We use the JSON format with SPs, tables and rows as dictionaries. The rows then

```
# Admin SP
 "0x0000020500000001": {
# Table table
  "0x0000000100000000": {
# Table table row
   "0x0000000100000001": {
# UID
    "0x00": "0x0000000100000001",
# Name
    "0x01": "(Table) 0x5461626c65",
# CommonName
    "0x02": "(Table) 0x5461626c65",
# TemplateID
    "0x03": "0x0000000000000000",
# Kind = Object
    "0x04": "0x01",
```

Figure 7.3: Excerpt of `discovery` output from Kingston KC600. Comments were added for clarity.

contain the columns as keys and the cells as values. Since the cells can contain diverse types, we have decided to parse only the outermost ones. This means that, for example, for lists, we print the inner bytes in hexadecimal form surrounded by square brackets.

An example of part of the output of this process can be seen in Figure 7.3. Depicted there is a part of the output of this utility containing the start of the Table table's first row. The nested structure with the outermost key corresponding to the Admin SP, followed by keys for the Table table and the row of the Table table for the Table table, can be seen.

8 Data collection

In order to be able to compare a large number of different Opal disks, we have decided to use our `discovery` utility. Data collected using this utility allows us to compare the capabilities and parameters reported by the disks easily. The fact that the utility is designed to not change the state of the analysed device means that we will be able to use community in order to expand our data set without having to inconvenience users collecting the data by erasing their data or activating the Locking SP of their disk and causing possible future issues with unexpected issues if they would use other Opal-managing tools in the future.

We have used the `discovery` utility to analyse SanDisk X300s, Kingston KC600, Samsung SSD 850 PRO, Samsung SSD 860 EVO and Samsung SSD 980 disks. The complete output can be found in the attachments.

8.1 Support of additional features

Using the output from our `discovery` utility, we can find out what non-mandatory features the tested disks support. Such information might provide us, among others, with insight into how realistic it would be to prevent some of the attacks introduced in Chapter 5, such as the evil maid attack, which could be partially prevented by secure messaging. In Table 8.1, we can see which features related to the security the tested disks support.

Table 8.1: Comparison of optional features related to security in disks

Disk model	Single User Mode	Block SID Authentication	Secure Messaging
SanDisk X300s	yes		
Kingston KC600	yes	yes	
Samsung 850 PRO	yes		
Samsung 860 EVO			
Samsung 980		yes	

We have also used the utility to compare the communication capabilities of the disks. Using the Level 2 Discovery information, we found out that none of the disks implemented any additional communication features described in Chapter 3.3, such as support of ACK/NACK or asynchronous communication. The only difference between the disks related to their communication parameters is the largest size of a packet and the maximum number of simultaneous authentications in a session. Otherwise, the parameters are kept at a minimum, with a maximum of one open session at once or one method per packet.

After an analysis of the properties acquired from the tested disks, we can summarise that the disks more or less did the minimum mandated by the standard. The absence of support for secure messaging, although not surprising due to the potential difficulty of implementation, is disheartening due to its possible use in securing the device.

8.2 Minor versions of Opal

If we want to compare the information provided by `discovery` utility with the requirements mandated by the standard, we must first establish which iteration of the standard we should use.

The newest version of the Opal standard is Opal 2.02 [11]. During the three minor versions, Opal underwent several changes. Some of the more noticeable changes are the additional mandatory Feature Sets. The PSID Feature Set was introduced as mandatory in version 2.01 and the Block SID Feature Set in version 2.02. Another significant change is the removal of the Interface Control Template. This template was removed in version 2.01 and was meant to function as a wrapper around vendor-specific drive features, such as upgrading firmware. The newest Opal 2.02 also introduced reporting of the minor version in the Opal SSC V2 Feature Descriptor. Since this change was introduced only in the 2.02 version of the standard, the two previous versions are not easily indistinguishable. In order to infer which version was used in the earlier versions, we will use the presence of the Interface Control Template and the earlier-mentioned feature sets. However, it should be noted that the presence of any of those features is not definitive proof of the version, as even if they are not mandatory, they can still be optional.

Table 8.2: Comparison of attributes relevant to the version

Disk model	PSID	Block SID	Interface Control	Reported minor version	Assumed version
SanDisk X300s	yes			n/a	2.01
Kingston KC600	yes	yes	yes	n/a	2.00
Samsung 850 PRO	yes			n/a	2.01
Samsung 860 EVO	yes			n/a	2.01
Samsung 980	yes	yes		n/a	2.01

As can be seen from Table 8.2, none of the disks reports its minor version, required from the 2.02 version of the standard. Although Kingston KC600 might seem to be compatible with the 2.01 version, considering its features, it contains Interface Control Template. This template was deprecated in 2.01, so we will consider this disk to implement Opal 2.00. As the remainder of the disks have not implemented Interface Control Template, we will consider them to be implementing Opal 2.01.

8.3 Inaccurate information

Other than the reported capabilities of the disks, we inspected also focused on incorrectly reported properties and compared the values reported by the disks to the values, value ranges, and formats described in the corresponding version of the Opal standard.

The disk we found had the highest amount of reported information that does not match the values required by the standard is the Kingston KC600. Even though this disk is the only one that reported many additional columns which are not required, most of the incorrect information is located within these not required columns. The Core standard defines not required features may be implemented and do not need to be tested. Some of such incorrect information is the sizes of tables or the size of the protected storage area, both being reported as always empty. However, the incorrect information being reported by this disk is not limited solely to not required columns. One of the mandatory columns being reported incorrectly is the number of

instances of the Base Template, which is reported as one despite the fact that there are two SPs, each with its own instance. In some cases, the reported values are not incorrect semantically but syntactically. For example, the same Kingston KC600 disk reports its supported SSCs as bytes, although the format defined by the standard is a list of bytes.

What most of the tested disks got wrong is the GUDID. GUDID is supposed to be a unique serial number of the disk corresponding to the NAA IEEE Registered format [30]. The SanDisk X300s reported an absolutely wrong value. A part of the GUDID is a few constant bytes, which in the case of this disk, were different. The three tested Samsung disks have reported empty values in place of any variable content, such as the Company ID or vendor-specific identifier.

Interestingly, Kingston KC600, which reported wrong values in other columns, reported this column mostly correctly. The only discrepancy with the format was that the value reported as the Company ID was registered only as their MAC address range and not as the Company ID with the IEEE Registration Authority.

9 Conclusion

In this thesis, we introduced several approaches to using hardware-encrypted disks in Linux. Afterwards, we have focused on the Opal standard more deeply as a most prominent representative of hardware disk encryption, describing its architecture and features. We described several state-of-the-art attacks relevant to hardware disk encryption and discussed the degree of protection provided by the Opal standard against them. We have introduced several popular tools for managing hardware disk encryption and presented our own toolset. Using the tools from this toolset, we managed to find a vulnerability in the generation of keys, and finally, we used it to compare the capabilities of several Opal-compliant disks.

As can be easily seen from the results presented in Chapter 5 and Chapter 8, even though hardware disk encryption presents many advantages over classic software encryption, today's implementations have flaws. Although most of the attacks require physical access, this is not unthinkable, as disks are often stolen. For some of the attacks, none of the disks provided a way to prevent them, even though a solution exists in the form of optional features. For those features that were required by the standard and implemented by the disks, some disks did not provide a working implementation. Between these bad implementations, there were even some malign ones, such as a disk not informing about setting an invalid value and even reporting the value as changed or a disk with a problematic generation of random values. As such, it may be easy to see the reluctance of some to trust hardware disk encryption and opt for software solutions instead.

9.1 Future work

During our analysis of the properties of Opal-compliant disks, we used data set consisting only of five disks. Since the `discovery` utility is designed to not require any authentication nor to modify the analysed disk, it could be possible to involve the public community to collect more data. In order to collect data about random number generators of Opal-compliant disks using the `rng` utility, involving the public community might be more complicated. Generating enough random

data using the utility might take quite a considerable amount of time. For example, some of the slower disks are able to generate only around 2 MB of random data per hour. Due to this, it might be much more intrusive to the members of the community compared to the few seconds the `discovery` utility takes.

We have also presented a `control` utility to manage the state of the Opal device. Even though in its current state it gives enough control over the device to allow basic usage, many additional features are not supported. Some of those features are Single User Mode or MBR shadowing. Although not necessary for future analysis of Opal devices, this tool could be updated to support these features.We have also presented a `control` utility to manage the state of the Opal device. Even though in its current state it gives enough control over the device to allow basic usage, many additional features are not supported. Some of those features are Single User Mode or MBR shadowing. Although not necessary for future analysis of Opal devices, this tool could be updated to support these features.

A Usage of tools

Before the tools can be used, they need to be compiled first. Part of the project is a `Makefile`, which allows a simple compilation of all the necessary project files by executing the `make` command. Since the code is designed to be self-sufficient, it is not necessary to install any other packages than the build tools. Afterwards, the tools are ready to be used. Note that in order for the tools to work correctly, they must be used with Opal-compliant disks.

Currently, only NVMe and SATA disks are supported. SATA disks require `libata.allow_tpm` option to be set. If this option is not set, a workaround using SCSI commands is used. However, this does not work well for all SATA disks.

In these few following sections, short documentation on the usage of the CLI tools implemented in this thesis can be found.

A.1 Usage of `rng` utility

The `rng` allows to access the generated by RNG of an Opal-compliant disk. This utility accepts the file of the device as the first argument. This file is usually `/dev/sda`, `/dev/nvme0`, or similar. Optionally, the number of bytes per one session, number of sessions, and the level of logging, in this order, can be used as the next arguments. The output is one or more lines of random bytes represented in hexadecimal. Each line represents a separate session. The invocation and the output of this utility can be seen in Listing A.1, Listing A.2, and Listing A.3.

Listing A.1: Execution with default arguments

```
# ./rng /dev/nvme0
65d3e1e12edb77f9aa82014a8472b7028841758fee50b...
```

Listing A.2: Execution specifying number of bytes per session

```
# ./rng /dev/nvme0 16
52c6bd17664b9eef860186a05e4ecf19
```

Listing A.3: Execution specifying number of sessions

```
# ./rng /dev/nvme0 16 4
```

```
2e42ac614b333a5b154e46b1f4c0e491
f7793cb0a8912cbbc8012927e5a64e10
57b5137fa729f7868f8e1639de84b7b4
7cede41adf9af1a17cbc350ef3a8387f
```

A.2 Usage of `discovery` utility

The `discovery` provides a way to print parameters of Opal-compliant disks. Similarly to the `rng` utility, this utility accepts the file of the device as the first argument. The next arguments are the selection of information and the logging level. The selection of information allows to limit the output to only its subset. Depending on the value the output may be limited to either only metainformation (containing output format versioning), output of identify commands, any Discovery level, or a few random bytes. Usage and output of this utility may be seen in Listing A.4 and Listing A.5.

Listing A.4: Execution providing entire output

```
# ./discovery /dev/sdc
{
"Metadata": {
  "Version": "1"
},
"Identify": {
  "Serial number": "S2BENCAHC05716J   ",
  "Firmware version": "EXM04B",
  "Model number": "Samsung SSD 850 PRO 512GB
                  "
},
"Discovery 0": {
  "TPer Feature": {
    "Version": 1,
    "ComID Mgmt Supported": 0,
    "Streaming Supported": 1,
    "Buffer Mgmt Supported": 0,
...
```

Listing A.5: Execution limited to Level 1 Discovery information

```
# ./discovery /dev/sdc 4
{
"Discovery 1": {
  "Properties": {
    "MaxComPacketSize": "0x010200",
    "MaxResponseComPacketSize": "0x010200",
    "MaxPacketSize": "0x0101ec",
    "MaxIndTokenSize": "0x0101c8",
    "MaxPackets": "0x01",
    "MaxSubpackets": "0x01",
    "MaxMethods": "0x01",
    "MaxAuthentications": "0x05",
    "MaxSessions": "0x01",
    "MaxTransactionLimit": "0x01",
    "DefSessionTimeout": "0x00"
  }
}}
```

A.3 Usage of `control` utility

The `control` utility allows to manage Opal-compliant disks. Since it contains many more distinct commands compared to the previous two tools, its invocation is more complex. Each command requires the disk file to be specified and also one or more of options specific to the command. In Listing A.6 an example of possible usage during a life cycle of the disk may be seen, with comments describing each step.

Listing A.6: Workflow

```
# Take ownership over the device, activating
# the Locking SP and assigning pin to the
# Admin authority.
./control setup_tper /dev/sdd                    \
    --assign-pin 0000
# Create two users, each with their own
# password.
./control setup_user /dev/sdd                    \
```

```
    --verify-pin 0000                          \
    --user 1 --assign-pin 1111
./control setup_user /dev/sdd                  \
    --verify-pin 0000                          \
    --user 2 --assign-pin 2222
# Enable first locking range and allow the
# two users to unlock it.
./control setup_range /dev/sdd                 \
    --verify-pin 0000 --locking-range 1        \
    --locking-range-start 0                    \
    --locking-range-length 512                 \
    --user 1 --user 2
# Lock the locking range as User1.
./control unlock /dev/sdd                      \
    --user 1 --verify-pin 1111                 \
    --locking-range 1 --read-locked 1
# Unlock the locking range as User2.
./control unlock /dev/sdd                      \
    --user 2 --verify-pin 2222                 \
    --locking-range 1 --read-locked 0
# Finally, revert the disk to factory state.
./control psid_revert /dev/sdd                 \
    --verify-pin ${PSID}
```

B Patterns in disk content

When data is encrypted using symmetric cipher under CBC mode with one key and decrypted with another, the result should be random. However, this is not the case with one of the disks we tested. Using this approach, SanDisk X300s contains data that may at first sight seem random, but after a deeper analysis, some patterns can be found.

In order to analyse the data present on the Opal disk after the DEK is re-generated, we used our utility `repeat_finder.py`. This simple tool iterates a file and finds repeating chunks of data. In our case, each chunk was 2048 bytes long. Each repeating chunks is printed as list of its locations, divided by the chunk size. An example of part of this output can be seen in Figure B.1. In this concrete example, there are two patterns we can see.

Listing B.1: Found patterns on zeroed disk

```
[0, 128, 32432]
[1, 3, 5, 7, 9, 11, 13, 15]
[2, 130, 32434]
[4, 132, 32436]
[6, 134, 32438]
[8, 136, 32440]
[10, 138, 32442]
[12, 140, 32444]
[14, 142, 32446]
[16, 32672]
[17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39,
    41, 43, 45, 47, 49, 51, 53, 55]
[18, 32674]
```

The first pattern can be seen on the first line, where the chunk repeats after 128 and 32432 chunks, and the pattern repeats several times for even locations. This pattern continues on throughout the entire analysed file, changing the value of the repeated bytes and the offsets and number of repeats once in a while. The second pattern can be seen on the second line, where the chunk is located in every odd location. This pattern continues on throughout the entire analysed file, changing the value of the repeated bytes once in a while.

It should be noted that such patterns emerged only when the disk contents were overwritten with a repeating bytes, such as all zeroes. After repeating the writing of repeating bytes and regenerating of the DEK, the patterns changed.

However, it is enough for two chunks to be the same. The Listing B.2 contains the patterns acquired from a disk that had its content randomized, except for two chunks. The bytes in fourth chunk were the same as the bytes in tenth chunk, an evidence of this can be seen in the listing.

Listing B.2: Found patterns on disk with two same chunks

```
[0]
[1]
[2]
[3, 9]
[4]
[5]
[6]
[7]
[8]
[10]
[11]
[12]
```

The source code of the utility and few of the outputs can be found in the attachments.

Bibliography

1. BASSETT, Gabriel; HYLENDER, C. David; LANGLOIS, Philippe; ALEX, Pinto; WIDUP, Suzanne. *Verizon 2022 Data Breach Investigations Report*. 2022.
2. FUJIMOTO, Aaron; PETERSON, Peter; REIHER, Peter. Comparing the Power of Full Disk Encryption Alternatives. In: *2012 International Green Computing Conference (IGCC)*. 2012, pp. 1–6. Available from DOI: 10.1109/IGCC.2012.6322245.
3. *Protection Profile for Full Disk Encryption: Mitigating the Risk of a Lost or Stolen Hard Disk*. 2011. Version 1.0. Protection profile. Information Assurance Directorate.
4. *Trusted Computing Group and NVM Express Joint White Paper: TCG Storage, Opal, and NVMe*. 2015. Tech. rep. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-opal-and-nvme/.
5. WEBER, Ralph O. (ed.). *Information technology - ATA/ATAPI Command Set - 3 (ACS-3)*. 2013. Standard. American National Standard of Accredited Standards Committee INCITS.
6. MEIJER, Carlo; GASTEL, Bernard van. Self-Encrypting Deception: Weaknesses in the Encryption of Solid State Drives. In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 72–87. Available from DOI: 10.1109/SP.2019.00088.
7. ALENDAL, Gunnar; KISON, Christian; MODG. got HW crypto? On the (in)security of a Self-Encrypting Drive series. *IACR Cryptol. ePrint Arch.* 2015, vol. 2015, p. 1002.
8. *About self-encrypting drives* [online]. 2022. [visited on 2022-11-27]. Available from: https://www.ibm.com/docs/en/psfa/7.2.1?topic=administration-about-self-encrypting-drives.
9. BIGGERS, Eric; TANGIRALA, Satya. *Inline Encryption*. 2021. Available also from: https://kernel.org/doc/html/v5.17/block/inline-encryption.html.
10. *Qualcomm® Inline Crypto Engine (UFS)*. 2021. Security policy. atsec information security.

11. *TCG Storage Security Subsystem Class: Opal*. 2022. Version 2.02. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/storage-work-group-storage-security-subsystem-class-opal.

12. *TCG Storage Architecture Core Specification*. 2015. Version 2.01. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-architecture-core-specification.

13. *TCG Storage Security Subsystem Class: Opalite Specification*. 2015. Version 1.00. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-security-subsystem-class-opalite.

14. *TCG Storage Security Subsystem Class: Pyrite Specification*. 2020. Version 2.01. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-security-subsystem-class-pyrite.

15. *TCG Storage Security Subsystem Class: Ruby Specification*. 2020. Version 1.00. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-security-subsystem-class-ruby-specification/.

16. *TCG Storage Opal SSC Feature Set: Additional DataStore Tables*. 2012. Version 1.00. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-opal-ssc-feature-set-additional-datastore-tables/.

17. *TCG Storage Feature Set: Block SID Authentication Specification*. 2021. Version 1.01. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-feature-set-block-sid-authentication-specification/.

18. *TCG Storage Opal SSC Feature Set: PSK Secure Messaging*. 2015. Version 1.00. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-opal-ssc-feature-set-psk-secure-messaging/.

19. *TCG Storage Opal SSC Feature Set: PSID*. 2015. Version 1.00. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-opal-feature-set-psid/.

20. *TCG Storage Opal SSC Feature Set: Single User Mode*. 2015. Version 2.00. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-opal-ssc-feature-set-single-user-mode/.

21. *TCG Storage Interface Interactions Specification (SIIS)*. 2021. Version 1.10. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/storage-work-group-storage-interface-interactions-specification/.

22. *TCG Storage Core Spec Addendum: Secure Messaging*. 2015. Version 1.00. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-core-spec-addendum-secure-messaging/.

23. *TCG Storage Protection Mechanisms for Secrets Specification*. 2012. Version 1.00. Standard. Trusted Computing Group. Available also from: https://trustedcomputinggroup.org/resource/tcg-storage-protection-mechanisms-for-secrets-specification/.

24. BAUER, Scott. *SED OPAL Library*. 2016. Available also from: https://lore.kernel.org/lkml/1482176149-2257-1-git-send-email-scott.bauer@intel.com/.

25. *NVM Express® Base Specification*. Tech. rep. Available also from: https://nvmexpress.org/developers/nvme-specification/.

26. STEVENS, Curtis E. (ed.). *ATA Command Pass-Through*. 2005. Standard. T10.

27. MÜLLER, Tilo; LATZO, Tobias; FREILING, Felix C.; FRIEDRICH-ALEXANDER. Self-Encrypting Disks pose Self-Decrypting Risks How to break Hardware-based Full Disk Encryption. In: 2013.

28. MÜLLER, Tilo; FREILING, Felix C. A Systematic Assessment of the Security of Full Disk Encryption. *IEEE Transactions on Dependable and Secure Computing*. 2015, vol. 12, no. 5, pp. 491–503. Available from DOI: 10.1109/TDSC.2014.2369041.

29. BOTEANU, Daniel; FOWLER, Kevvie. Bypassing Self-Encrypting Drives (SED) in Enterprise Environments. In: *Black Hat*. 2015.

30. BREHER, Joe (ed.). *SCSI Primary Commands - 5*. 2019. Standard. T10. Available also from: https://www.t10.org/members/w_spc5.htm.