FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER’S THESIS

Title:	Security analysis of USB drive
Student:	Bc. David Jagoš
Supervisor:	Ing. Jiří Buček
Study Programme:	Informatics
Study Branch:	Design and Programming of Embedded Systems
Department:	Department of Digital Design
Validity:	Until the end of winter semester 2019/20

Instructions

Research existing vulnerabilities of encrypted USB drives. Perform a security analysis of Kingston DataTraveler Vault Privacy encrypted flash drive. Try to find its vulnerabilities in order to extract stored data without knowledge of the password.
- Analyze the internal structure of the flash drive.
- Analyze the client software supplied with the flash drive.
- Analyze and document undocumented APIs.
- Perform experiments in order to analyze the dependency of the encryption key on the password.
Evaluate the results and discuss potential impact of your findings.

References

Will be provided by the supervisor.

doc. Ing. Hana Kubátová, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 23, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Security analysis of USB drive

Bc. David Jagoš

Department of Digital Design
Supervisor: Ing. Jiří Buček

May 9, 2018

Acknowledgements

I'd like to thank my supervisor, Ing. Jiří Buček for his counsel and the immense patience he has shown.

Declaration

Citation of this thesis

Jagoš, David. *Security analysis of USB drive.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstrakt

Tato práce shrnuje bezpečnost flash disků s hardwarovou podporou šifrování a poskytuje bezpečností analýzu disku Kingston DataTraveler Vault Privacy.

Klíčová slova šifrování flash disků, Kingston, DataTraveler Vault Privacy, DTVP, Phison, PS2251-63, PS2263

Abstract

This thesis provides an overview of the security of flash drives with hardware encryption support and a security analysis of Kingston DataTraveler Vault Privacy.

Keywords flash drive encryption, Kingston, DataTraveler Vault Privacy, DTVP, Phison, PS2251-63, PS2263

Contents

List of Figures

Introduction

We live in a world of information and protecting that information is becoming more important with each passing year. The never-ending barrage of information leaks forces use to realize just how much damage can be done when sensitive information falls into the wrong hands.

Private citizens risk their personal information being leaked and misused for identity theft or blackmail and corporations could lose their intellectual property or leak business strategies. These scenarios may not seem too frightening, but there are others, much more serious ones as well. Governments have sensitive information about all of us, not to mention lists of undercover police and intelligence operatives, locations of military bases, personnel rosters and details of national defense systems just to name a few. And let's not forget dissidents who could end up being persecuted by oppressive regimes if their data got in the wrong hands. In these scenarios, security of information is literally a matter of life and death, whichever side you're on.

One of the most prominent ways we store and transport data are USB flash drives. Their small size and portability make them very practical, but also very easy to lose. In a survey conducted by the Ponemon Institute on behest of Kingston [1] 70 percent of respondents reported the organization they were working for suffered a loss of sensitive or confidential information as a result of losing a flash drive. That is a staggering statistic. If those flash drives were recovered by a third party and the data on them was unprotected, this could result in huge loses as well as prosecution by authorities for mishandling personal information.

It is clear that if we are to store sensitive data on a flash drive, we need to protect it. But how? There are of course traditional software based encryption tools such as archive managers with encryption support (e.g. 7-zip) and tools for creating encrypted virtual volumes as well as encrypting physical drives (e.g. BitLocker, TrueCrypt and its successor VeraCrypt), but they all share one common weakness—there is no limit on how many passphrases the attacker may try, as well as no rate limiting other than the inherent computational complexity of the cryptographic algorithms used. If you rely purely on software solutions to secure your data and lose your flash drive, an attacker will be able to try passphrase after passphrase, potentially at the speed of millions per second, for as long as he or she needs to break in. Are you really confident your passphrase will be able to withstand such a barrage? And what it you accessed your encrypted data on a different computer—the purpose of a flash drive is to be portable after all—which was running a keylogger?

Flash drives with hardware encryption support offer answers to many of these issues, but do they really solve them, or are those just empty promises? The relatively long history of failed products in this category doesn't seem very encouraging, but perhaps manufacturers have learned their lesson from past failures. Furthermore, there are no standards or certifications designed specifically for flash drives with hardware encryption support. There are some more general standards for cryptographic modules like the FIPS-140 which are often applied to these flash drives, but they are hardly comprehensive for this use case.

With identity theft on the rise and the General Data Protection Regulation looming over companies' head, there has arguably never been a greater demand for secure flash drives.

Privacy is a right and security of information is increasingly often a legal necessity, but do we have the technical means to ensure them?

This thesis attempts to answer that question.

In chapter 1 I explain what are flash drives with hardware encryption support and summarize their defining characteristics. In chapter 2 I go over past attacks against flash drives with hardware encryption support. In chapter 3 I introduce the Kingston DataTraveler Vault Privacy and explain why it was chosen as the subject of this thesis. In chapter 4 I divine the properties of PS2251-63—the controller used in Kingston DataTraveler Vault Privacy. Finally, in chapter 5 I analyze the software shipped with Kingston DataTraveler Vault Privacy and use the knowledge gained to try and devise attacks against the flash drive.

Chapter 1

Flash drives with hardware encryption support

First, let's properly define what is a flash drive with hardware encryption support. This chapter lays out their key characteristics and how they differ from their normal counterparts.

1.1 Where is the encryption performed

The main difference between flash drives with hardware encryption support and ordinary flash drives is where the encryption is performed. This is their defining characteristic.

1.1.1 Ordinary flash drives

If encryption is used at all with an ordinary flash drive, it is performed by the host computer. This can be problematic, because the cryptographic secrets are—at least for the duration of encryption or decryption—present on the host computer. If the computer is compromised at that time, the secrets can be retrieved by an attacker. Furthermore, the nature of these tools makes them a much easier target than a dedicated physical device doing the encryption—it is much easier to reverse engineer them or substitute them with a modified malicious version.

1.1.1.1 Encrypted archives

The most commonly used tool for protecting data on a flash drive is an archive manager. Archive managers weren't designed with data encryption in mind—that function is only a bonus—which makes quite cumbersome for this use (the archive or a significant part of it must be recompressed when every time its contents are changed) as well as lacking in security (for a file to be viewed or edited, it must first be extracted onto hard drive from where its unencrypted copy could be recovered later), but their ubiquity still makes them the first choice of many.

The most notable products in this group are 7-zip and WinRAR.

1.1.1.2 Encrypted virtual disks

A much better option are products utilizing encrypted container files whose contents are exposed to the host system as virtual disk. This not only greatly improves performance as changing a single file doesn't require recompressing and re-encrypting the entire container, but also security. As far as the host operating system is concerned, any file being viewed or modified is already present in the clear, so there's no need to copy onto an unencrypted disk; the files unencrypted version will only exist in RAM.

A huge practical advantage of this approach is that the encryption software can be stored on the flash drive alongside the encrypted container.

The most notable products in this category are the discontinued TrueCrypt and its successor VeraCrypt.

1.1.1.3 Full disk encryption

The last common method for protecting data on ordinary flash drives is full disk encryption. In this scenario the entire content of the drive (including partition headers) is encrypted. This brings a marginal performance boost over using a virtual drive as this approach removes a layer of abstraction, but it doesn't bring any significant security benefits and is much less practical than using an encrypted virtual drive for reasons outlined in the previous section.

The most notable products in this category are BitLocker, TrueCrypt and VeraCrypt.

1.1.2 Flash drives with hardware encryption support

In flash drives with hardware encryption support, the encryption is performed by the controller of the flash drive. The encryption/decryption is completely transparent to the host computer—as far as it is concerned a completely ordinary flash drive is connected.

If this scheme is implemented properly, no cryptographic secrets ever leave the flash drive, so even if the host computer is compromised, an attacker can only steal data currently saved on the drive, but will not able to unlock it him or herself later. This approach, however, often incurs a significant performance penalty, especially with small random reads and writes, as the controllers are relatively slow compared to modern computers and the data usually needs to be encrypted and decrypted in larger blocks for increased security.

1.2 User input methods

Flash drives with hardware encryption support can be split into categories by how users authenticate themselves.

1.2.1 Software

Some flash drives authenticate the user via software running on the host computer. These drives usually present themselves to the system as a CD-ROM drive containing the software necessary for unlocking the encrypted partition. After the user is authenticated, a USB mass storage device is presented to the host computer.

This approach allows users to comfortably input even very long passphrases, but it suffers from the same weakness as all software-based encryption: the user's passphrase is, even if only for a short time, present in the memory of the host computer.

1.2.2 Hardware

Hardware-based approaches to authenticating users guarantee that no cryptographic secrets ever leave the flash drive, but they are often less practical.

1.2.2.1 Keypads

Some drives opt for a physical keypad on the drive itself where a user can enter his or her password. The physical dimensions of a flash drive severely limit the number of distinct characters available (usually decimal digits) and the bad ergonomics limits the practical length of a password. As a result, passwords input via a physical keypad on a flash drive tend to have very low entropy. An example can be seen in figure 1.1.

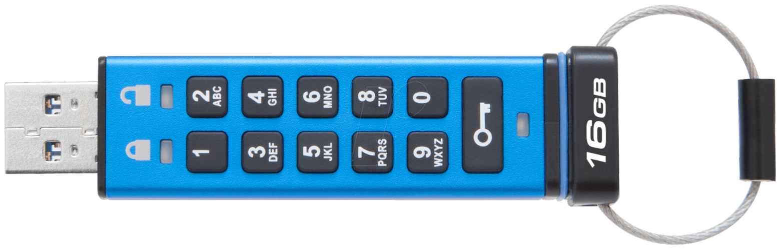


Figure 1.1: Kingston DataTraveler 2000 16 GB [2]

1.2.2.2 Fingerprint scanners

Fingerprint scanners offer much better ergonomics as well as much higher entropy, making them, in theory, almost a perfect solution. In practice, however, most finger print scanners are easily fooled (sometimes even with just a printout of the user's fingerprint). An example can be seen in figure 1.2.

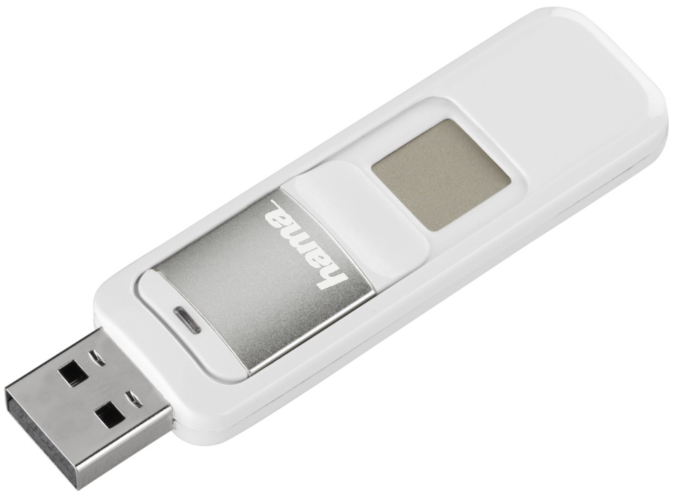


Figure 1.2: Hama "ProtectionKey" FlashPen [3]

1.2.2.3 RFID readers

This approach is mostly used in portable hard drives with hardware encryption support. The drive is unlocked with an RFID tag. The obvious flaw of this scheme is that for practical use, the RFID tag needs to be transported with the hard drive and if an attacker can get one, there's no reason to think he or she won't be able to obtain the other.

1.3 Typical hardware configuration

The typical hardware configuration of a flash drive with hardware encryption support can be seen in figure 1.3. The host system communicates with the flash drive over USB. The encryption process is completely transparent, so the communication is exactly the same as with an ordinary USB mass storage device. The controller performs the encryption and decryption, as well as all the standard housekeeping tasks associated with using a flash memory. Some controller manufacturers provide the controller and flash memory in a single package for increased security (no eavesdropping or MitM possible) as well

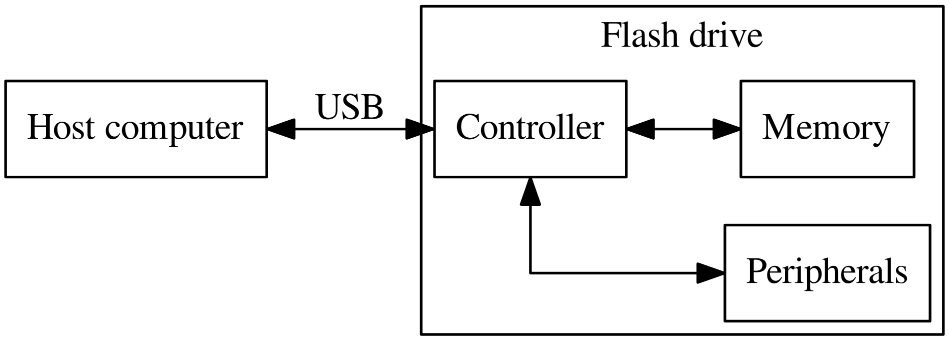


Figure 1.3: Typical hardware configuration of a flash drive with hardware encryption support

as simpler PCB layout. Finally the controller may also communicate with peripherals such as keypads or fingerprint readers.

Chapter 2

Past attacks

In this chapter I will go over some past attacks on encrypted flash drives.

2.1 The SySS hack

In late December 2009 SanDisk issued a security bulletin [4] announcing several drives from their Cruzer Enterprise lineup had “a potential vulnerability in the access control mechanism” and that they released a software update addressing the issue. Kingston [5] and Verbatim [6] soon followed suit. Kingston even issued a recall, offering to replace affected drives with newer models.[7]

The vulnerability was discovered by SySS GmbH, a German pen-testing company.[8] While reverse engineering the control application for one of the affected drives, SySS researchers noticed that the application always sent the same binary string to the flash drive to unlock it, irrespective of the password used. They then tried simply replaying this sequence and the drive unlocked. Every one of the affected drives across different manufacturers could by unlocked by simply sending it this magic sequence.

This tells us something very important: Flash drive manufacturers aren't the ones implementing the cryptography, that's done by a third party, presumably the controller manufacturer.

This vulnerability affected the following drives:

- SanDisk Cruzer Enterprise USB flash drive, CZ22 - 1GB, 2GB, 4GB, 8GB
- SanDisk Cruzer Enterprise FIPS Edition USB flash drive, CZ32 - 1GB, 2GB, 4GB, 8GB
- SanDisk Cruzer Enterprise with McAfee USB flash drive, CZ38 - 1GB, 2GB, 4GB, 8GB
- SanDisk Cruzer Enterprise FIPS Edition with McAfee USB flash drive, CZ46 - 1GB, 2GB, 4GB, 8GB

- Kingston DataTraveler BlackBox (DTBB)
- Kingston DataTraveler Secure - Privacy Edition (DTSP)
- Kingston DataTraveler Elite - Privacy Edition (DTEP)
- Verbatim Corporate Secure USB Flash Drive - 1GB, 2GB, 4GB, 8GB
- Verbatim Corporate Secure FIPS Edition USB Flash Drives - 1GB, 2GB, 4GB, 8GB

Please not that these drives were FIPS 140-2 Level 2 certified US National Institute of Standards and Technology.

2.2 The Google hacks

In July 2017 three Google engineers gave a talk at BlackHat USA titled "Attacking Encrypted USB Keys the Hard(ware) Way." Sadly, they didn't publish a paper, so all we have is a recording of the talk[9] and a PDF with slides[10]. They also chose not to directly identify any of the products they cracked. I did manage to identify most of them, but even in cases where I failed to identify the product, we can still gleam the kinds of mistakes the vendors made designing it.

2.2.1 Generic flash drive with a fingerprint scanner

This flash drive is being sold under many obscure brand names as well as without any branding whatsoever (see figure 2.1). As such, the expectations of quality aren't too high.

The flash drive has a separate fingerprint manager chip and flash controller and the two communicate over UART (see figure 2.2). The message sent from the fingerprint manager to the flash controller isn't some form of a fingerprint digest as one might expect, it is a simple number—an ID of the recorded fingerprint that matched the scanned finger. Furthermore the IDs aren't even randomized, they're assigned sequentially, starting from a known value.

All an attacker needs to do to break into this flash drive is tap into the UART lines and send the ID of the first recorded fingerprint to the flash controller. This is very similar to the SySS hack. The attack is harder to carry out as the device needs to be physically opened and two wires need be connected (not too much harder though—the UART pins are nicely laid out and labeled on the PCB), but it is worse in that it can't be fixed with a software patch.

Figure 2.1: Generic flash drive with a fingerprint scanner[10]

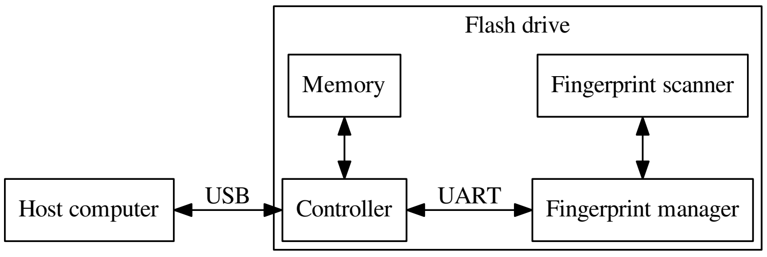


Figure 2.2: Structure of a flash drive with a fingerprint scanner

2.2.2 Digittrade RS64 RFID portable hard drive

While not a flash drive, the weakness demonstrated here could very well apply to one.

The drive is unlocked with a simple RFID tag which can be surreptitiously cloned using equipment anyone can buy for a couple of dollars. If a drive like this is left unattended under the presumption that the data on it can only be accessed by authorized personnel, a malicious actor could quickly read the RFID tag belonging to one of the authorized operators and later create a copy of this tag and use it to access the data.

Another critical flaw in this product is that while it does limit the number of login attempts, the value of this counter seems to be stored in a volatile memory and is reset to 0 after a restart. The process of trying several IDs and restarting could be easily and cheaply automated for a brute-force attack. The feasibility of this attack would depended on the length of the ID of the used

Figure 2.3: Digittrade RS64[10]

RFID tag as well as on whether the IDs are assigned at random or sequentially. This information was sadly not revealed during the presentation.

2.2.3 DM PD061 flashdrive with a fingerprint reader

This flash drive uses a fingerprint scanner to authenticate users and a software tool to manage users. Multiple users can be registered simultaneously and to manage users you need an administrator password. It also has an extremely silly undocumented feature: you can request the administrator password and the flash drive will give it to you. In cleartext. No authentication required.

Figure 2.4: DM PD061[10]

With this password an attacker can then log into the user management program, register him or herself as a new user and unlock the drive using his or her own fingerprint.

2.3 Unverified claims

While working on this thesis I came across a number of people claiming they had broken the encryption of various flash drives from obscure ones to Kingston's IronKey D300. The individuals making these claims never publicly released them, nor have they shown any proof of concept, making their claims unverifiable. While a lot of these claims were obvious lies, some seemed plausible and two of them were made by highly reputable gurus on their respective flash drive hacking forums. I'm not going to cite any specific claims here (frankly, I didn't even bother writing them down as I came across them), but I will say I strongly believe at least some of them to be true and that there are many more compromised flash drives than the public is aware of.

Chapter 3

Kingston DataTraveler Vault Privacy

Originally released in 2009, the Kingston DataTraveler Vault Privacy is quite an old device. It was chosen for analysis despite its age because it was the drive Kingston offered as a replacement for drives compromised in the SySS hack,[7] so there was a reasonable expectation it should be secure and I wanted to assert how well that expectation was met.

Figure 3.1: Kingston DataTraveler Vault Privacy 8GB

In this chapter I will go over the drive's specifications, its hardware structure and the management software bundled with it.

3.1 Specifications

The Kingston DataTraveler Vault Privacy boasts the following features[11]:

- All data is encrypted with 256-bit AES in CBC mode.
- Up to 24MB/s read speed.
- Up to 10MB/s write speed.
- Can be mounted in a read-only mode.

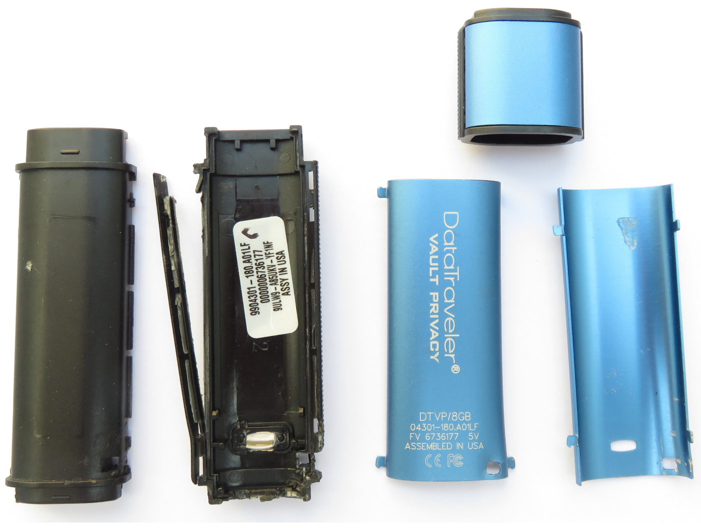


Figure 3.2: Enclosure taken apart

- The drive destroys stored data after 10 unsuccessful login attempts.
- Windows, Linux and OS X support.
- USB 2.0 compliant.
- IPX8 rating (waterproof up to 4 feet).
- Operational temperature from 0 °C to 60 °C.
- Storage temperature from −20 °C to 85 °C.

The drive comes in 4GB, 8GB, 16GB, 32GB and 64GB variants.

3.2 Hardware

The circuit board is placed in a black plastic enclosure with an outer aluminum shell (figure 3.2).

The drive is built around the Phison PS2251-63-6 flash controller and one or two 48-pin TSOP NAND flash chips. My testing unit came with a single MT29F64G08CFACA flash chip (figures 3.3 and 3.4).

With the help of the flash chip's datasheet[12] I can decode the part number: It is a 64Gbit MLC NAND flash with an 8-bit shared address and data bus and asynchronous I/O.

I could not find a datasheet for the PS2251-63-6; I will address this issue in chapter 4.

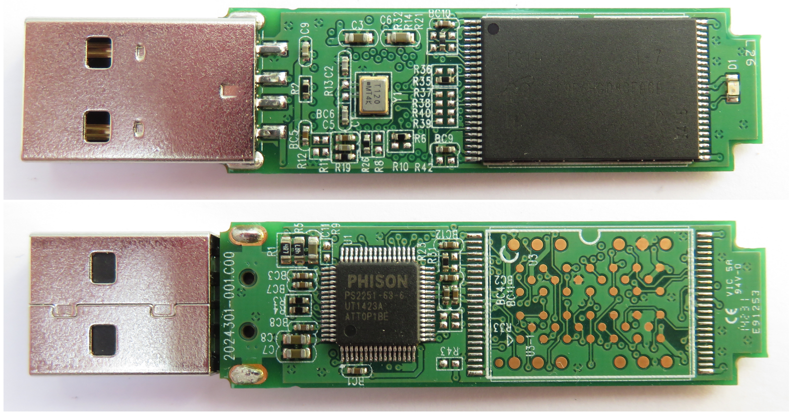


Figure 3.3: Front and back views of the PCB

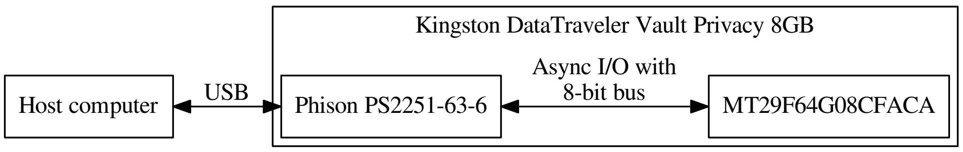


Figure 3.4: Kingston DataTraveler Vault Privacy 8GB hardware diagram

3.3 Software

When the flash drive is connected, it presents itself to the host computer as a CD-ROM drive containing the control software and a manual.

The Windows control software is a simple GUI-base application allowing the unlocking of the drive or resetting it (figure 3.5), changing setting such as the password or contact Information (figure 3.6) and viewing the device information (figure 3.7).

The OS X application is visually and functionally identical to its Windows counterpart. It is written in Java.

Lastly the Linux control software consists of 5 command line tools:

- dtvp_about
- dtvp_forgotpassword
- dtvp_initialize
- dtvp_login
- dtvp_logout

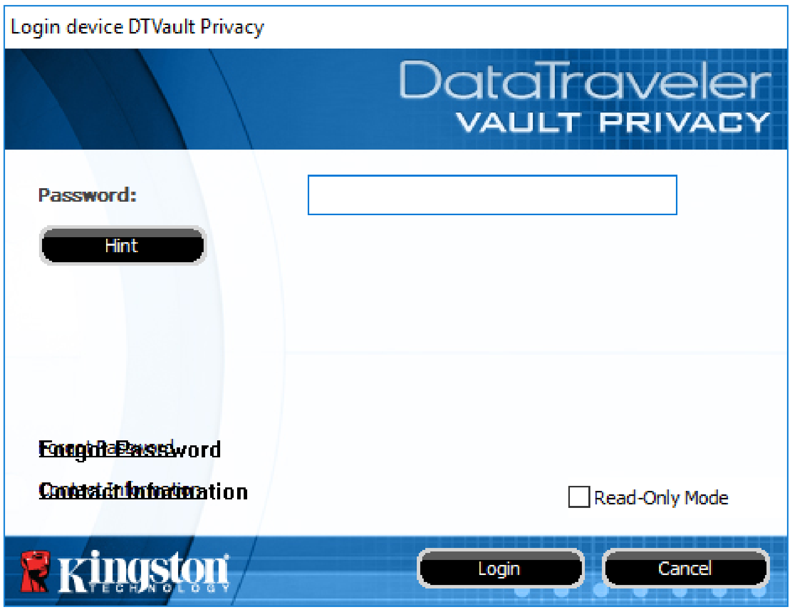


Figure 3.5: Windows unlock dialog

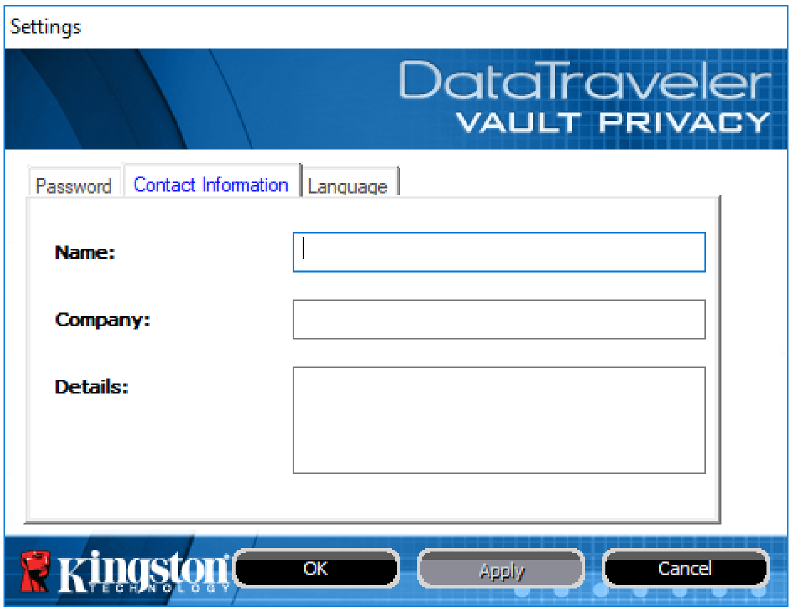
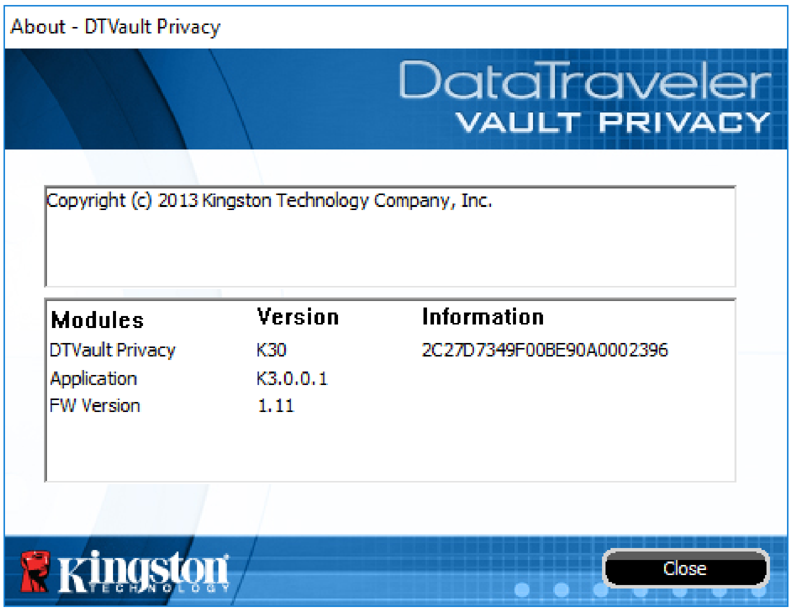


Figure 3.6: Windows settings dialog



Copyright (c) 2013 Kingston Technology Company, Inc.

Modules	Version	Information
DTVault Privacy	K30	2C27D7349F00BE90A0002396
Application	K3.0.0.1	
FW Version	1.11	

Figure 3.7: Windows device information dialog

Password is limited to length between 6 and 16 characters and has to contain characters from at least three of the following groups: uppercase letters, lowercase letters, digits and special characters.

Chapter 4

Phison flash drive controllers

Since I could not find a datasheet for the PS2251-63-3, alternatively called PS2263 (datasheets are not public, presumably they're only officially available to Phison's customers under an NDA and are therefore often found in very dubious places), I decided to compile information from datasheets of a couple other models of Phison flash controllers (specifically PS2251-31/PS2231 and PS2251-33/PS2233) as well as bits of information shared on various flash drive repair and hacking forums to get an idea of the general structure and feature set of these controllers.

I focused on USB 2.0 NAND flash controllers.

4.1 General characteristics

They are an all-in-one solution specifically designed for USB flash drives. The following characteristics seem to be common to all models:

- USB 2.0 support
- four endpoints
  - Endpoint 0: 64 Bytes CONTROL transfer
  - Endpoint 1: 512 Bytes BULK transfer for IN transaction
  - Endpoint 2: 512 Bytes BULK transfer for OUT transaction
  - Endpoint 3: 64 Bytes INTERRUPT transfer for IN transaction
- supports SLC and MLC NAND with 2k and 4k pages
- 3.3V and 1.8V NAND
- in-system programming via USB
- 8051 compatible processor core

Following are characteristics supported only by some models which I believe to be relevant to the PS2251-63-6:

- MLC NAND with 8k pages
- Secure hidden area
- 256-bit AES hardware module
- hardware RNG

It seems reasonable to assume that the encryption key or a part of it would be stored in the secure hidden area.

4.2 Firmware

As expected, I couldn't find a PS2251-63 firmware image. I did manage to find leaked firmware images for 6 different Phison flash controllers, but none of them were for controllers with hardware AES support (or secure hidden area support), so there would be little to gain from analyzing them.

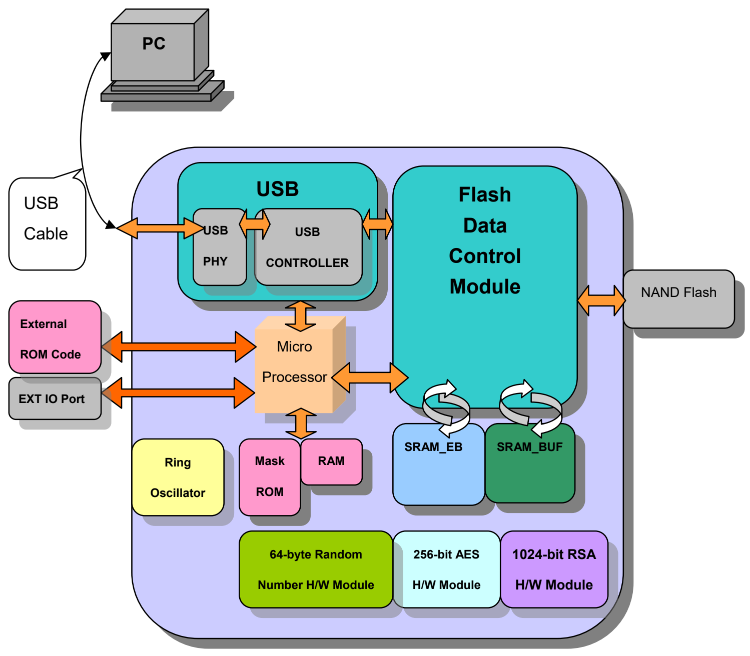


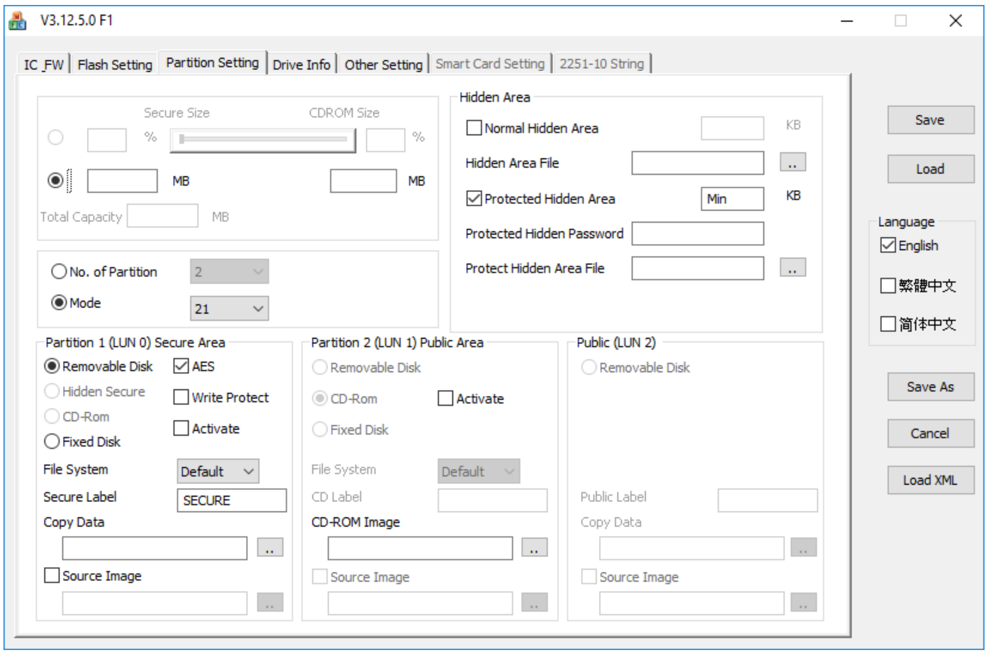
Figure 4.1: PS2251-33 block diagram

According to a block diagram from a PS2251-33 datasheet (figure 4.1), the controller stores one part of the firmware in mask ROM (programmed during chip fabrication, unchangeable, probably a bootloader of sorts) and the other (presumably main) part is stored off-chip. Since there no chips on the board other than the controller and the NAND flash, the firmware must be stored in the flash memory.

There is a leaked tool set called MPALL for flashing firmware on flash drives based on Phison controllers via USB. According to USBDEV.ru, there also is a tool for dumping the firmware via USB, both however require a so-called "burner file" which is unique for each controller and is usually leaked alongside the firmware file. Unfortunately the burner file for PS2251-63 seems not have been leaked, so if the firmware were to be obtained, it would need to be retrieved from the NAND flash directly.

4.3 Features

While the firmware for PS2251-63 hasn't been leaked, the firmware configuration tool bundled with MPALL does support it, so, combined with a leaked manual for an older Phison firmware configuration tool, we can gleam what features this controller has to offer.



Secure Size %
CDROM Size %
MB
MB
Total Capacity MB

No. of Partition 2
[✓ Mode] 21

Hidden Area
Normal Hidden Area KB
Hidden Area File
[✓ Protected Hidden Area] Min KB
Protected Hidden Password
Protect Hidden Area File

Language
[✓ English]
繁體中文
简体中文

Partition 1 (LUN 0) Secure Area
[✓ Removable Disk]
Hidden Secure
CD-Rom
Fixed Disk
[✓ AES]
Write Protect
Activate
File System Default
Secure Label SECURE
Copy Data
Source Image

Partition 2 (LUN 1) Public Area
Removable Disk
[✓ CD-Rom]
Fixed Disk
Activate
File System Default
CD Label
CD-ROM Image
Source Image

Public (LUN 2)
Removable Disk
Public Label
Copy Data
Source Image

Figure 4.2: Phison firmware configuration tool

The most interesting part of the configuration tool is partition settings tab seen in figure 4.2 (the rest is about flash NAND used, device name and IDS and how the LED should behave). Here we can see that the controller supports up to 3 partitions which can be presented to the system as a removable drive, a CD-ROM or a hard drive (the greyed-out options are disabled because of the selected mode, not because the controller doesn't support them). The first partition can be encrypted with AES. We can also choose data to be preloaded onto the partitions. Finally, we can configure a hidden area and a protected hidden area which seems to be protected with a password.

According to a patent awarded to Phison[13], the hidden area is a part of the flash the host computer will not be allowed to access by the controller. Combined with the fact that the controller seems to have no writable non-volatile on-board memory, the hidden area (and the protected hidden area especially) seems like a good place for cryptographic secrets to live.

Chapter 5

Analysis of Kingston DataTraveler Vault Privacy

In this chapter I reverse-engineer the control software supplied with Kingston DataTraveler Vault Privacy, evaluate its properties and use my findings in an attempt to devise attacks against the flash drive.

5.1 Challenges

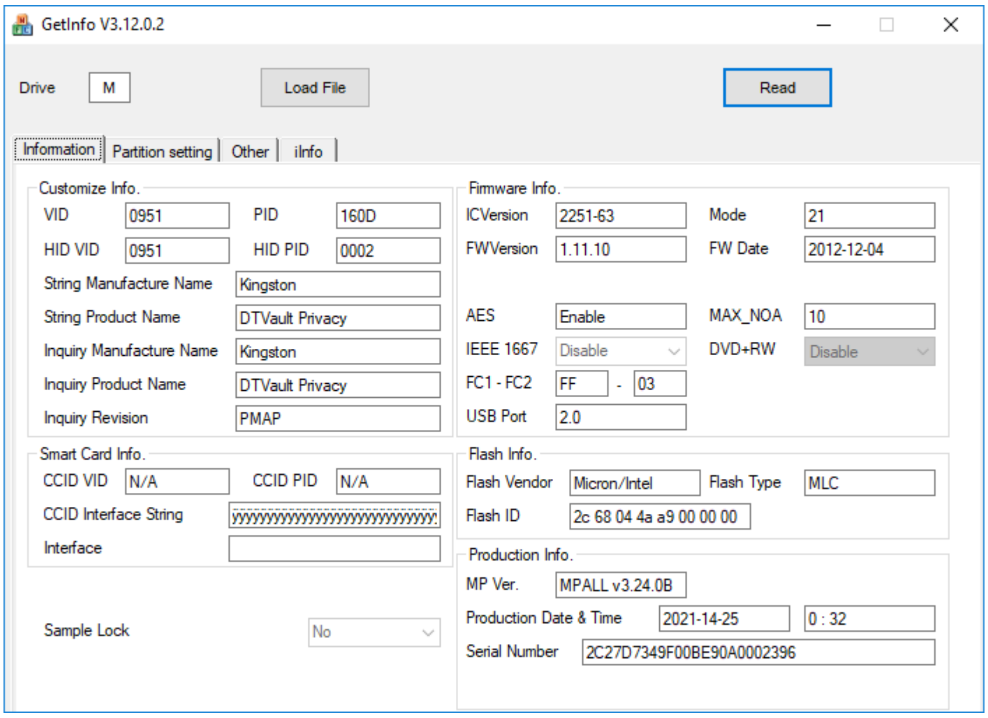
A huge setback to all my efforts was the fact that we were only able to source one testing unit. Being released in 2009, the drive hadn't been for sale in shops for some years. I searched high and low, combing through Amazon, eBay and even Craigslist. There were some offers, but the new drives were prohibitively expensive (by then more or less only the 64GB and 32GB models were left for sale new and those always were hideously expensive). Even though not ideal for research as they theoretically could have been tampered with, I would have settled for second-hand drives, but none of the sellers were willing to ship internationally. I even tried asking on reddit, offering to pay cash or exchange the drive for the updated USB 3.0 model. I did have a couple takers initially, but when it came down to business, they backed out citing privacy concerns.

With only one testing unit, I had to be very careful and had to forego any experiments that could damage it.

5.2 Configuration

The MPALL tool set has one last trick up its sleeve: the GetInfo tool.

As can be seen on the Information tab (figure 5.1), the unit is using AES encryption, the wrong password limit is set to 10 and it set to mode 21 which, as the Phison firmware configuration tool manual tells us, means running with


GetInfo V3.12.0.2

Drive: M | Load File | Read

Information | Partition setting | Other | iInfo

Customize Info.

VID	0951	PID	160D
HID VID	0951	HID PID	0002
String Manufacture Name	Kingston		
String Product Name	DTVault Privacy		
Inquiry Manufacture Name	Kingston		
Inquiry Product Name	DTVault Privacy		
Inquiry Revision	PMAP		

Firmware Info.

ICVersion	2251-63	Mode	21
FWVersion	1.11.10	FW Date	2012-12-04
AES	Enable	MAX_NOA	10
IEEE 1667	Disable	DVD+RW	Disable
FC1 - FC2	FF - 03		
USB Port	2.0		

Smart Card Info.

CCID VID	N/A	CCID PID	N/A
CCID Interface String	yyyyyyyyyyyyyyyyyyyyyyyyyyyyy		
Interface			

Sample Lock: No

Flash Info.

Flash Vendor	Micron/Intel	Flash Type	MLC
Flash ID	2c 68 04 4a a9 00 00 00		

Production Info.

MP Ver.	MPALL v3.24.0B	
Production Date & Time	2021-14-25	0 : 32
Serial Number	2C27D7349F00BE90A0002396	

Figure 5.1: GetInfo – Kingston DataTraveler Vault Privacy – Information tab

two partitions—one set as a removable drive and the other as a CD. So far everything checks out.

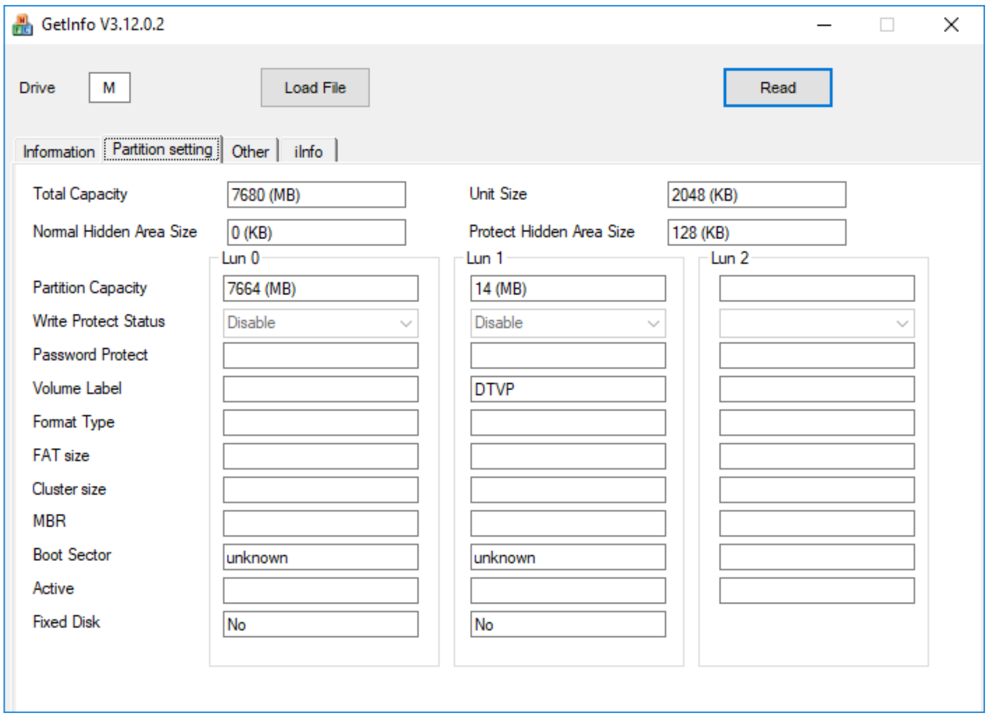
The really important bit of information comes on the Partitions settings tab (figure 5.2). Here we can see the drive has no hidden area and 128kB protected hidden area, further supporting my earlier hypothesis about the drive using the protected hidden area to store cryptographic secrets.

5.3 Management software analysis

The Windows management program is a standard 1.12MB C++ MFC application. There is no obfuscation, it could be reverse-engineered relatively easily. It is digitally signed.

The OS X application is written in Java. I didn't look into it much since I do not own an OS X machine.

When I loaded up the Linux command line tools, I got a very nice surprise—Kingston forgot to strip the binaries! That means I have the original name (and prototype) of every single function and global variable as well as some extra debug symbols, such as source file names. Pretty much the only thing missing from having a full source code are function parameter names. The file and function names are identical across the five binaries, suggesting only



Drive	M		
Total Capacity	7680 (MB)	Unit Size	2048 (KB)
Normal Hidden Area Size	0 (KB)	Protect Hidden Area Size	128 (KB)

	Lun 0	Lun 1	Lun 2
Partition Capacity	7664 (MB)	14 (MB)	
Write Protect Status	Disable	Disable	
Password Protect			
Volume Label		DTVP	
Format Type			
FAT size			
Cluster size			
MBR			
Boot Sector	unknown	unknown	
Active			
Fixed Disk	No	No	

Figure 5.2: GetInfo – Kingston DataTraveler Vault Privacy – Partition setting tab

the main function was ever changed. It would seem they didn't even bother changing the main source file name—it's "DTVP_Login.cpp" in all five tools.

- crtstuff.c
- DTVP_Login.cpp
- KT_SDK.cpp
- Rijndael.cpp
- FunctionalityLayer.cpp
- HashSDK_PWS.cpp
- integer.cpp
- nbtheory.cpp
- RSA.cpp
- SCipher.cpp
- sha2.cpp

- rand.cpp
- CommandLayer.cpp
- hashFFour_SDK_PWS.cpp
- hashFOne_SDK_PWS.cpp
- hashFThree_SDK_PWS.cpp
- hashFTwo_SDK_PWS.cpp
- TransportLayer.cpp

Most of the files seem to be dealing with standard cryptographic functions and can be safely ignored. DTVP_Login.cpp seems to only contain the main function. That, together with functions from KT_SDK.cpp, Functionality-Layer.cpp and CommandLayer.cpp, could be interesting. Bellow I've compiled a list of functions with very appealing names that might be worth exploring:

- CS03::HA__Get_Current_Number_of_Attempts
- CS03::HA__Read_Page
- CS03::HA__Read_Protected_Page
- CS03::HA__Read_Secure_Page
- CS03::HA__Write_Page
- CS03::HA__Write_Protected_Page
- CS03::HA__Write_Secure_Page
- int, unsigned char*, int)
- CS03::ReadHiddenArea
- CS03::WriteHiddenArea
- CSCSICommand::ClearPassword
- CSCSICommand::DirectReadPage
- CSCSICommand::DirectWritePage
- CSCSICommand::LoginPassword
- CSCSICommand::ReadKeyFromFlash
- CSCSICommand::ReadKeyFromSRAM

- CSCSICommand::SetPassword
- CSCSICommand::WriteKeyToFlash
- CSCSICommand::WriteKeyToSRAM
- KTChangeCredentials
- KTReadHiddenPage
- KTUnblockCredentials
- KTWriteBlockRaw
- KTWriteHiddenPage

The authentication process begins by an RSA handshake and all sensitive communication is encrypted using AES-128.

```
50 44 4D 47 FE 87 31 30 B9 61 88 45 02 C6 78 20
```

This key is rather ubiquitous; it seems to pop in all sorts of Kingston products.

5.4 Attacks

There are three kinds of attacks I'm going to explore. In order of desirability they are: retrieving the password or encryption key, performing the cracking offline (i.e. without the flash drive connected) and resetting the password try counter.

5.4.1 Retrieving the password or encryption key

Assuming a drive is properly designed, there is no chance of the password or the encryption key being stored anywhere on the flash drive—at most a hash of the password or a part of the key—but that is a pretty big assumption to make, so lets investigate all the functions that retrieve data from the flash drive one way or another.

Using a debugger script to repeatedly call the KTReadHiddenPage function, I tried to dump the first 256 pages of the protected hidden area. Aside from pages 9, 10 and 12 which contained information like device name, help URL or password hint, they were all filled with a repeating random 16 byte pattern which is a result of "decrypting" zeros with the standard Kingston key. Internally the KTReadHiddenPage function uses CS03::HA__Read_Protected_Page (which is not used by anything else), so that's off the list too.

CS03::HA__Read_Page is the same as CS03::HA__Read_Protected_Page without the decryption step. It is only used to retrieve the serial number.

CS03::HA__Read_Secure_Page is never called and I couldn't get it to work. Maybe I didn't identify the parameters correctly or maybe it requires some form of authentication (the Phison flash controller datasheets does mention a host with "Trusted host ID" being able to access the secure area).

CS03::ReadHiddenArea is used by KTReadBlockRaw which, despite its name, seems to be used exclusively for reading data from the unprotected hidden area. Since this device doesn't have any unprotected hidden area, this function is useless for me.

The CSCSICommand::ClearPassword, CSCSICommand::LoginPassword and CSCSICommand::SetPassword write to the flash drive and could therefore potentially brick it, so I didn't dare test them.

CSCSICommand::ReadKeyFromFlash retrieved 5 bytes of data and CSCSICommand::ReadKeyFromSRAM retrieved 32 bytes of data. While that is the correct length for an AES-256 key, the data doesn't look random at all and doesn't change between drive resets. For what it's worth, here it is:

```
12 01 00 02 00 00 00 40 51 09 0D 16 10 01 01 02
03 01 04 03 09 04 FF FF FF FF FF FF FF FF FF FF
```

Without the firmware image, there isn't much else I can do in terms of software. The contents of the NAND flash could be dumped via hardware, that would, however, require desoldering the chip which brings with it a fairly high chance of destroying it. Since I only have one testing unit, that was out of the question.

5.4.2 Offline cracking

In an offline attack, the cryptographic functions of the controller would have to be analyzed in detail, replicated and then run on an image of the data stored on NAND flash. While not as good as retrieving the password or the encryption key, it is the next best thing—it gives the attacked an unlimited number of password attempts and high rate of trying them.

Since I didn't find a software way to dump the contents of the flash (and the hardware approach is too risky in my situation) and I do not have a firmware image, I was forced to forego this attack.

5.4.3 Try counter reset

Just like the offline attack, the counter reset attack would allow the attacker to try an unlimited number of passwords, however, compared to the offline attack, it offers much lower throughput. Since normal login takes about 2 seconds with this drive, we're talking about dozens of attempts per minute at most. And that's not factoring in the time it would take to reset the counter.

In order to find out where this information is stored, let's examine the CS03::HA__Get_Current_Number_of_Attempts function which retrieves the number of unsuccessful attempts and the maximum number of attempts. Unfortunately it does this by sending a specific SCSI command to the controller, so it is not much use to us as it does not reveal the location of these values. While unlikely, a write version of this command might exist. If it does though, its hypothetical structure is not obvious from the code I've analyzed.

KTGEtDeviceInfo obtains this information through calls to CS03::SD__Query_Service, so that doesn't help us much either.

Once again, I have failed on the software front, however, since I've already established that the counter value must be stored in the NAND flash, it could certainly be reverted by dumping the contents of the flash before making an attempt and writing it back afterwards. This process would of course be terribly slow, but it could be significantly sped up by dumping the contents of the flash after making an unsuccessful attempt, comparing it with the original dump, identify areas where the images differ and only writing back those areas between attempts.

This approach would lead to a rapid local degradation of the flash memory as the wear-leveling is normally done by the flash controller which would be effectively bypassed now. This could be mitigate by sourcing flash chips with the same geometry as the original and using those instead.

Once again, I had to forego this attack because of a high risk of damaging the only testing unit I have, however I am very confident it would work, however impractical it may be.

5.5 Dependency of the encryption key on the password

Since I have found no safe way of reading the raw data from flash memory, I could not easily determine whether the encryption key changed with the password, was randomly generated upon each reset, or didn't change at all. I tried to hook up a 16-channel logic analyzer to the control and data pins of the flash chip and read the same section of data (filled with zeros) before and after a password change, but I was unable to get a consistent reading. As it turns out, data isn't stored on a flash chip sequentially, the way it was sent to the controller. The memory address space is broken up into logical blocks and those are mapped to different physical blocks for wear-leveling purposes. Also for wear-leveling purposes, the data is XORed with a pseudo-random mask before being stored. I couldn't get a consistent reading because the controller kept shuffling the data around. This is yet another issue that could be resolved by dumping the flash memory.

5.6 Future work

I believe I have explored all the options I safely could. The need to dump or alter the contents of the NAND flash has popped up multiple time during my research, so I believe that to be the logical next step. There still are a few unused functions in the binaries that need to be tested. After that, fuzzing the device—especially at the SCSI command level—might be in order. This logically follows dumping the flash memory as rewriting the contents should resolve most fault conditions potentially caused by the fuzzing.

Conclusion

Reverse engineering is an unfortunate topic for a thesis as the amount of work required is hugely disproportional to the amount of text that can be written about it. Most of the effort goes into understanding the inner workings of the program analyzed that is far too detailed for a publication such as this.

Never the less I believe I did achieve the objective of this thesis—even if my analysis was not exhaustive as a result of the aforementioned problems with sourcing testing units, I did satisfactorily demonstrate that the Kingston DataTraveler Vault Privacy is a significant improvement over the drives it replaced after 2010 hack. I also devised a possible attack against the drive. Although untested and hugely impractical, I am very confident it is feasible.

Finally, I charted a course for anyone who is interested in continuing this research.

Bibliography

[1] Ponemon Institute, LLC. The State of USB Drive Security in Europe. 2011, [cit. 2018-05-09]. Available from: https://media.kingston.com/pdfs/Ponemon/Ponemon_research_EMEA_summary_UK_1111.pdf

[2] Kingston DataTraveler 2000 16 GB. [cit. 2018-05-09]. Available from: https://cdn-reichelt.de/bilder/web/xxl_ws/E910/KINGSTON_DT2000_16GB_02.png

[3] Hama "ProtectionKey" FlashPen. [cit. 2018-05-09]. Available from: https://www.hama.com/bilder/00124/abx/00124197abx.jpg

[4] Corporation, S. Security Bulletin December 2009. 2009, [cit. 2018-05-09]. Available from: https://web.archive.org/web/20091220042009/http://www.sandisk.com/business-solutions/enterprise/technical-support/security-bulletin-december-2009

[5] Corporation, K. T. Kingston's Secure USB Drive Information Page. 2009, [cit. 2018-05-09]. Available from: https://web.archive.org/web/20091224102747/http://www.kingston.com/driveupdate/

[6] Verbatim Americas, LLC. Important Security Update December 2009. 2009, [cit. 2018-05-09]. Available from: https://web.archive.org/web/20100108171617/http://www.verbatim.com/security/security-update.cfm

[7] Corporation, K. T. Kingston Digital to Replace Affected Secure USB Flash Drives with Upgraded Security Architecture, New Drives. 2010, [cit. 2018-05-09]. Available from: https://www.kingston.com/us/company/press/article/40506

[8] Schmidt, J. NIST-certified USB Flash drives with hardware encryption cracked. *The H*, 2010, [cit. 2018-05-09]. Available from: http:

//www.h-online.com/security/news/item/NIST-certified-USB-Flash-drives-with-hardware-encryption-cracked-895308.html

[9] Picod, J. and Audebert, R. and Bursztein, E. Attacking Encrypted USB Keys the Hard(ware) Way. 2017, [cit. 2018-05-09]. Available from: https://www.youtube.com/watch?v=jVKl3GuazEs

[10] Picod, J. and Audebert, R. and Blumenstein, S. and Bursztein, E. Attacking encrypted USB keys the hard(ware) way. 2017, [cit. 2018-05-09]. Available from: https://www.blackhat.com/docs/us-17/thursday/us-17-Picod-Attacking-Encrypted-USB-Keys-The-Hard%28ware%29-Way.pdf

[11] Corporation, K. T. Kingston DataTraveler Vault - Privacy datasheet. 2013, [cit. 2018-05-09]. Available from: https://www.kingston.com/datasheets/DTVP_en.pdf

[12] Micron Technology, Inc. *Micron Technology 64Gb, 128Gb, 256Gb, 512Gb Asynchronous/Synchronous NAND Features*. November 2009, rev. A.

[13] File protecting method and system, and memory controller and memory storage apparatus thereof. 2012, United States, [cit. 2018-05-09]. Available from: http://www.freepatentsonline.com/8954692.html

Appendix A

Acronyms

USB	Universal Serial Bus
RAM	Random Access Memory
MitM	Man in the Middle
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
MLC	Multi-level Cell
I/O	Input/Output
SLC	Single-level Cell
SCSI	Small Computer System Interface

Appendix **B**

Contents of enclosed CD

```
dtvp_cd ........................ DTVP read-only CD partition contents
├── datasheets ........................................ datasheets directory
├── tools ........................................... related tools directory
│   └── MPALL .................................. various versions of MPALL
├── src ............................................ source codes directory
│   └── thesis .................................. source codes of the thesis
└── text ........................................ the thesis text directory
    └── thesis.pdf ......................... the thesis text in PDF format
```