A comparison of encryption tools for disk data storage from digital forensics point of view

Vitor Hugo Galhardo Moia, Marco Aurélio A. Henriques

[1]School of Electrical and Computer Engineering (FEEC)
University of Campinas (UNICAMP)
Campinas, SP, Brasil 13083-852

vhgmoia,marco@dca.fee.unicamp.br

Abstract. *A closer look from a digital forensics point of view may help us to find and understand vulnerabilities in cryptographic tools to determine which ones are the safest and provide the best features. In this paper, we analyze different ways of using cryptography to protect data on hard drives, discuss some vulnerabilities and present particular features of disk encryption tools. Then we analyze the security of some tools according to a set of criteria and evaluate the level of expertise required to use them. We conclude that the encryption subject impacts the scope of tools' vulnerabilities and full disk encryption is the safest and easiest option. Moreover, we highlight some free and easy to use tools that can protect users data properly if basic precautions are taken.*

1. Introduction

Digital Forensics aims to solve crimes committed with computers and electronic devices where evidences may reside on and can be used against criminals in court. The development of anti-forensic (AF) tools [Kessler 2007] with the purpose of thwarting investigations has increased in the past years. Cryptography, the most efficient AF tool, is a strong measure to protect users data against unauthorized access. In the past, when adversaries used encryption to protect their data, forensics experts could recover useful information to assist in investigations, due to some traces left in the operating system, known vulnerabilities in the tools used, weak choice of passwords by users etc.

However, the development of strong encryption tools and their integration into the operating systems have introduced new challenges for forensic analysis. Some methods encrypt the whole disk, even the operating system files and unallocated sectors on disks, prompting a password when the device is initialized before the machine boots, preventing unauthorized access to data. Normally, the only and not viable way to get users data in such case is by means of brute-force or dictionary attacks on the password. This way, the use of encryption tools became one of the major challenges of current digital forensic investigators, and its use is increasing even more since these tools are becoming easy and simple to use.

Nonetheless, not all hope is lost for forensic examiners when encryption is used. A lot of tools do not provide full disk encryption, and traces of useful information and sometimes the decryption keys may be found in the device. Also, there could be known vulnerabilities that can be exploited to gain access to data or, in some cases, the assistance of system administrators or IT. personnel in providing the recovery keys of the system.

In this scenario, if the forensics examiners can recover information, adversaries can too. Better ways to protect users privacy are necessary. However, with so many tools available, how can users distinguish a secure from an insecure tool? Which ones are in fact able to protect their privacy? Which features are the most appropriate? A closer look at these tools from a forensic perspective allow us to identify vulnerabilities and understand the provided features, in order to answer this sort of questions. In this paper, we analyze different ways of using cryptography to protect data stored on hard drivers, discuss some vulnerabilities from a forensic point of view and present particular features of disk encryption tools. Then, we compare some encryption tools according to a set of criteria to analyze security aspects, followed by an evaluation with respect to the minimum level of expertise required to use different features of the tools. We expect that our evaluation will help users choosing the best tools according to their needs.

2. Encryption methods for data at rest

The data encryption processes can be classified according to its subject, such as File Encryption (FE), Virtual Disk Encryption (VDE) and Full Disk Encryption (FDE) [C. 2010]. In this section, we will explain each one of these methods and give examples of some related tools. We will also present and discuss some vulnerabilities in these three methods and particular features of FDE tools.

2.1. Classification of encryption methods

1. **File Encryption (FE):** Method where individual files are encrypted, such as images, audio, videos, documents, personal backups, and so on. Examples of encryption tools available are: GNU Privacy Guard (GPG) [GNU 1999], AxCrypt [AxCrypt 2001], 7-Zip [Pavlov 1999], among others.
2. **Virtual Disk Encryption (VDE):** In this mode users can create an encrypted container, which is a single file responsible for holding as many files and/or folders as they wish. Getting access to its content requires a passphrase to unlock it, which after being correctly supplied, mounts a virtual disk of the container where users can easily store and read data from it. This is a type of on-the-fly encryption, which means that data is automatically encrypted or decrypted as it is loaded in the container or saved elsewhere. Examples of VED tools are: GNU Privacy Guard (GPG), VeraCrypt [IDRIX 2014], CipherShed [PCM 2014], R-Crypto [R-Tools 2007], The Linux Unified Key Setup (LUKS) [Guardian 2014].
3. **Full Disk Encryption (FDE):** This is a more sophisticated level of encryption, where the whole disk, including the operating system and all its files along with users data is encrypted. Unallocated spaces of the hard drive are also encrypted. Usually, before booting a encrypted system there is an interface to collect the right decryption key, preventing access of those who do not posses the credentials. Examples of such tools are: Bitlocker [Microsoft 2006], VeraCrypt, CipherShed, DiskCryptor [NTLDR 2007], FileVault2 [Apple 2011], The Linux Unified Key Setup (LUKS), among others.

2.2. Vulnerabilities to be exploited

Usually, when encryption is used to protect data, there is not much what forensic examiners can do to break keys and recover useful information. Current cryptographic tools are well designed and reviewed, presenting no vulnerabilities (at least not known by most people). However, depending on the type of encryption applied, some methods can get users' cryptographic keys or obtain some useful information. Also, there are other ways to get users data without attacking the algorithms.

When **File Encryption (FE)** and **Virtual Disk Encryption (VDE)** tools are employed, there is a broader scope of attacks that can succeed in some cases, allowing forensic examiners recover useful information despite the use of the strongest algorithms. In such cases, Casey E. and Stellatos G.J. [Casey and Stellatos 2008] point out that plaintext versions of some encrypted files may be obtained in temporary files, in the process of creating or editing documents, or even in a spool file created while printing data. Besides, deleted files that have not been overwritten on hard drivers are also a possible source of useful information. The authors highlight that keyword searching could be an effective mechanism for finding relevant information in plaintext, looking for dates, phrases, and other known characteristics. As a matter of fact, forensic examiners may also be able to recover encryption keys from passphrases stored on hard drives or removable media (thumb drivers, external hard drivers etc), since many users prefer to write them down to avoid forgetting them.

Mrdovic S. and Huseinovic [Mrdovic and Huseinovic 2011] indicates that encryption keys can also be recovered from hibernation files, responsible for storing the system memory content when the computer hibernates (or enters in a power-saving mode). The RAM may hold sensitive information from a running computer, such as decryption keys, passphrases or other relevant information, which will be written in these files. Analyzing them allows forensic examiners getting the keys to decrypt data or important information to a case. The same concept applies to the memory dump method, analyzed by Balogh S. and Pondelík M. [Balogh and Pondelik 2011], where the examiners obtain a dump of the memory content to work with in the search of useful data.

Another possible way to look for traces of sensitive data are through paging files or bad sectors on disks [IDRIX 2015]. The former, also called swap file, is related to data that do not fit in memory and are stored on disk unencrypted. The latter refers to bad sectors of a disk that once may have stored useful information and now are marked as bad sectors due to problems in the writing/reading processes. In this case, a new sector is allocated to store the content of the bad one. This means that data in these sectors can not be overwritten, causing security implications due the old data that still remains.

In cases where **FDE** tools are employed, these approaches described are not suitable for recovering useful information. In such cases, the hard drive is fully encrypted preventing forensic examiners to access any operating system files. This way, new methods for overcoming this technology are necessary. However, it is important to highlight that as long as the computer is on and the passphrase was supplied, forensic examiners can access everything, such as personal data, memory content, operating system files etc, since all data will be decrypted. FDE tools usually decrypt the cryptography keys once the right credentials are supplied and keep them on memory to decrypt other content on the hard drives as necessary. On the other hand, if the computer is off, recovering useful information becomes harder and, sometimes, an impossible task. One way is through a brute-force or dictionary attack in order to guess users' passphrase. In this scenario, the adversary needs to locate the sector(s) storing the decryption keys on the hard drive, extract them and perform the attack, trying lots of combinations until it matches. For poorly chosen passphrases, this task should not take a long time. But if users choose a strong passphrase, the attack fails since it would require a large amount of time. Zhang L. et al. [Zhang et al. 2014] show how to proceed in such attack, where they studied the TrueCrypt system (version 7.1a), a cryptographic tool precursor of VeraCrypt and CipherShed. They also cite the possibility of using a GPU (Graphics Processing Unit) to accelerate the process. A similar work is done by Kornblum J. D. [Kornblum 2009], but the author explore the BitLocker system, providing useful information for forensic examiners while dealing with this encryption tool. Other way of attacking some FDE systems is by the recovery password feature available in some softwares [Zhang et al. 2014], [Kornblum 2009]. In case of BitLocker, users can save the recovery key in their Microsoft account. If so, this could be a way of attacking the system. Adversaries first steal user's credentials and then have access to the recovery key.

Another vulnerability to be exploited independent of the encryption subject, is the users behaviour. In general, users do not take the necessary precautions regarding the right management of the cryptography keys neither create a strong passphrase to protect their keys. They usually chose easy to remember passwords, which are not large enough and encompasses common words from the dictionary, names of places, celebrities or family, last names, dates etc, vulnerable to brute-force or dictionary attacks. This may be a consequence of the fear and impact of losing their passwords, since it prevents users from accessing their data or using their computers (specially when FDE tools are used). Social engineering is another kind of attack involving users that most tools are vulnerable.

Weak implementations of the cryptographic algorithms are also a vulnerability to be exploited. With the source code released to the community, this risk tends to be mitigated since there will be many eyes evaluating the code. However, developers are always subject to making mistakes, as the ones in the Drown Attack [Aviram et al. 2016] and the Heartbleed Bug [Codenomicon 2014]. For the non-open source applications, the risks are even greater with addition of doubts about backdoors implanted to compromise the whole security by those who know them.

In some contexts, the forensic experts may have support from a server administrator or the I.T. team from a company in which some of its employees are under investigation. Such support could be supplying a recovery password, for example, allowing the decryption of the whole data.

Finally, Halderman J. A. et al. [Halderman et al. 2009] describe another alternative to attack encrypted systems, using a technique called Cold-Boot Attack. It is similar to the memory dump technique but with the difference that it uses cooling techniques to retain information on memory after the computer is turned off. In some cases, this technique could be a solution to recover the decryption keys of FDE systems.

Table 1 summarizes the vulnerabilities according to encryption subject, where the mark (✓) represents the presence of the vulnerability in that mode of encryption.

2.3. Particular features of FDE tools

FDE tools present some particular features due to their requirements that need to be understood, to allow users making a good choice of a tool that fits their needs. In the following subsections, we will present and explain these features.

Table 1. Vulnerabilities according to the encryption subject

Encryption Subject	Vulnerabilities									
	Plaintext version of old files	Hibernation files	Memory Dump	Paging files	Bad sectors on disks	Brute-force or Dictionary Attacks	Social Engineering	Weak implementation	Support from others	Cold Boot Attack
FE	✓	✓	✓	✓	✓	✓	✓	✓		✓
VDE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FDE					✓	✓	✓	✓	✓	✓

2.3.1. The XTS mode for disk Encryption

When it comes to data encryption on storage devices, the XTS (**X**EX **T**weakable Block Cipher with Ciphertext **S**tealing) mode is the one recommended by NIST [Dworkin 2010]. It was designed specifically for encryption on storage devices with fixed length data units, and it is a modified version of the XEX (XOR Encrypt XOR) tweakable block cipher, with the difference of using two independent keys instead of one.

This mode of operation takes advantage of the hard drives structure, where the disks are partitioned into circular paths (the tracks) and these are divided into fixed-sized (typically 512 bytes) logical sectors, where information is stored. The encrypted data is placed in a sector including the cryptographic key and any necessary metadata, which usually correspondes to the sector number and block number in that sector. This is how the XTS stores the encrypted data, protecting better than other modes against ciphertext manipulation and cut-and-paste attacks [Martin 2010].

Martin [Martin 2010] explains how this mode of operation works. Let us consider a block cipher E encrypting a message M with a key K, producing a ciphertext C ($C = E_K(M)$). A tweakable block cipher operates on M, K, and a tweak T using E to result in C ($C = E_K(T, M)$). The tweak is similar to an initialization vector (IV) used in standard operation modes as CBC (Cipher Block Chaining), but it does not require to be random or to remain secret. Its purpose is to provide variability in the ciphertext. Also, a tweak block cipher has to remain secure even if the adversary knows the tweak.

In the XTS mode there are two tweaks, one corresponding to the sector number (j) and the other to the block number in that sector (i). It calculates a ciphertext C from a message M and the two keys K_1 (primary key) and K_2 (secondary key) as follows:

$$E_{K_1,K_2}(i, j, M) = E_{K1}(M \oplus \Delta) \oplus \Delta,$$

where $\Delta = \alpha^{\mathrm{i}} \bigotimes E_{K2}(j)$, $\bigotimes$ is polynomial multiplication modulo $x^{128} + x^7 + x^2 + 1$ and α a constant primitive element in $GF(2^{128})$. This process is illustrated in Fig. 1, where a message M is broken in n equal size blocks and encrypted with K_1 and K_2, resulting in the ciphertexts C_1 to C_n. The value of i goes through 1 to m, where m is the number of blocks that fits in a single sector. The decryption process can be seen in Fig. 2.

The use of XTS as an operation mode for disk encryption has some advantages besides being secure against common attacks in this context. The first one is regarding error propagation when a ciphertext block gets damaged, affecting the decryption of that particular block only. Another advantage is the use of parallelism in the processes of encryption and decryption to accelerate the process.

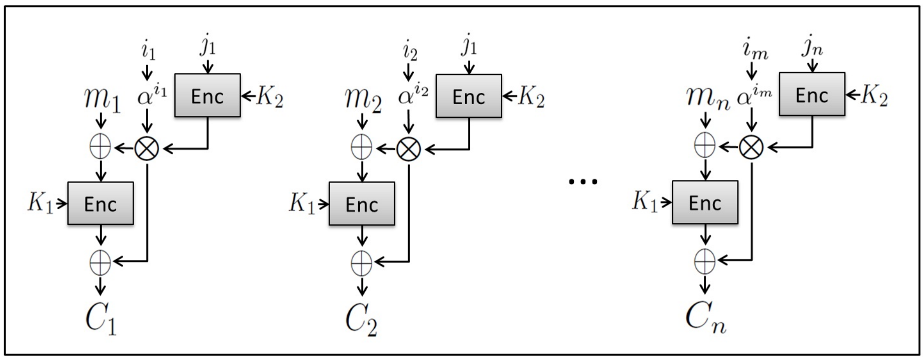

Figure 1. Encryption process in XTS mode

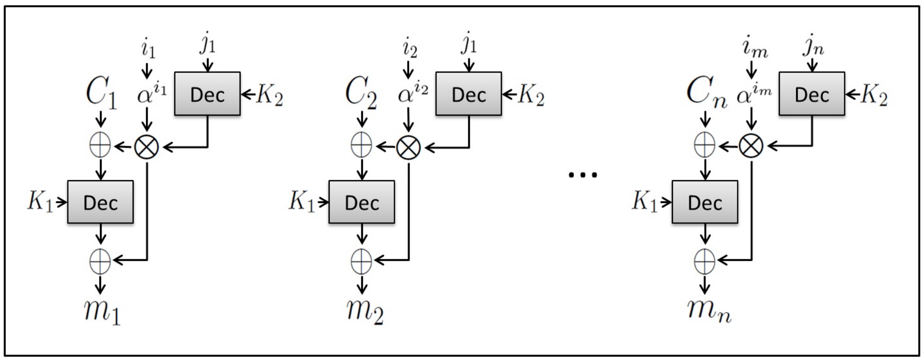

Figure 2. Decryption process in XTS mode

2.3.2. Plausible Deniability

As highlighted by Lowman S. [Lowman 2010], plausible deniability is a feature available in some encryption tools which allows users to deny the use of cryptography. They may blame the presence of ciphertext on others, saying that a malicious software infected their machines and encrypted everything or that no encrypted data exists at all. Without the proper encryption key, it is difficult for an adversary to prove that plaintext data exists. Lowman S. says that "it is difficult to determine whether a disk contains encrypted versus random data".

When plausible deniability is employed, users create an encrypted message consisting of two keys and two messages, one representing the real information and the other a fake one. In case of coercion, users can decrypt the latter message with the second key, revealing the fake data. That way, the real information is protected and the adversary convinced that users have no futher information. Without the knowledge of the right key, one cannot prove that more data exist [Canetti et al. 1997].

2.3.3. Use of Trusted Platform Module

A Trusted Platform Module (TPM) is a secure cryptoprocessor embedded in the motherboard with the purpose of authenticating hardware devices. Encryption tools can store cryptographic keys inside the TPM. When the system is booting, there is a process that verifies the integrity of the hardware and operating system in that machine. In case the verification succeeds, the encryption keys are used by the cryptoprocessor and the whole disk is decrypted, allowing the booting process to continue. However, using this technology implies tying the volume to that particular device, making decryption difficult, if not impossible, to suceed if the volume is removed and placed in another device [Kornblum 2009]. In case the TPM or motherboard have problems, users will not be able to access their data, unless they have a recovery key. This is one of the disadvantages of using the TPM, since it creates a single point of failure.

3. Encryption Tools comparison

In this section we present a comparison between some encryption tools available nowadays. We selected a small group of well known tools that implement the three cryptography usages in a disk storage system (FE, VDE and FDE) and analyzed them according to some criteria in order to show users the better tools according to their needs. We will also present an analysis about the level of expertise required from users in the tools used, evaluating the minimum amount one needs in order to make good use of each processes analyzed, according to our expertise and experience with these tools.

3.1. Security aspects

In the following items we will present the criteria used to evaluate the encryption tools and a brief explanation of each one.

- **Algorithms:** Represents the algorithm(s) available for encrypting data. More than one algorithm may be available for users' choice, and some tools allow users to use them in cascade, for example, where data is encrypted more than once.
- **Key recovery:** This feature allows users to recover their keys in case they lose or forget their password. This could be a good way to avoid the burden of losing all data in such case. However, adversaries can use this artifice to break the encryption. Also, some tools allow system administrators to have access to the recovery keys, which in some cases could represent as a threat to users.
- **Algorithm operation mode:** It is already known that some operation modes are more secure than others. In our analysis, we are considering tools that are used for encrypting data on hard drives, which require a different treatment, for example, the reuse of IV, forbidden for some ciphers. For this reason, some encryption modes should be avoided, as the most common cipher CBC and other well known stream ciphers (CTR, CFB, OFB), due to problems pointed out by Fruhwirth C. [Fruhwirth 2005].
- **Open source code:** According to Kerckhoff's principle, only the key should constitute the secret information. This stands in contrast with the concept of 'security by obscurity', where the security is achieved by keeping the algorithm secret. Having a code open to the community can bring several advantages to the software. For example, the public scrutiny, where the code will receive an extensive study by the community and will be tested against several attacks. Also, if the security relies on the algorithm secrecy, methods as reverse engineering represent a big threat and can compromise the software. Thus, this characteristic is desired in a secure tool and helps eliminating vulnerabilities.
- **Encryption subject:** This feature refers to the classification of the tool regarding its encryption subject, as discussed previously: File Encryption (FE), Virtual Disk

Encryption (VDE) and Full Disk Encryption (FDE). It is important to highlight that some tools can operate in more than one mode.

- **Multi-factor authentication:** This feature is related to the use of other factors in addition to the passphrase to authenticate users and decrypt their data. For example, VeraCrypt uses the concept of keyfiles, which, besides user's password, demands the presence of a certain file to derive a key and decrypt users data.
- **Operating system:** This characteristic represents all the available choices of operating systems for using the encryption tool.
- **Portable software:** This option allows users to execute a software without installing it on the computer. Gupta D. and Mehtre B. M [Gupta and Mehtre 2013] highlight that usually this kind of applications does not leave files or any data on host systems. Also, they do not use the registry or configuration files, and can be stored on removable storage devices. This means that the traces left behind by their use will be limited, making forensics work harder.
- **Plausible deniability:** Users can use this technique to avoid adversaries know about the use of cryptography to protect information.
- **File Attributes Encryption:** Encrypting data attributes, such as filename, creation and last modifications dates, ownership, among others, is also important feature for preserving users' privacy. Usually, a file is named according to its content in order to make easier its identification later. However, this could help an adversary in several ways, for example, identifying the most valuable files that worth stealing and focusing the attack on them.
- **Security level:** This criteria evaluates the size of the keys available in the algorithms. The larger the keys, the bigger its search space will be, making brute-force or dictionary attacks harder to succeed.
- **Hidden containers:** Some tools offer a similar feature to plausible deniability, with the difference of allowing users to create a hidden area in the hard drive (instead of a fake content) which will be revealed only with a specific key.
- **TPM:** Use of a Trusted Platform Module to manage the cryptographic keys and validate the hardware. Ties the volume to the device.
- **Key Stretching:** Use of an algorithm for key stretching and number of iterations performed in the process. The purpose of this feature is to slow down attacks such as brute-force or dictionary, in which the adversary tries to figure out the right password and steal users' keys. A small number of iterations make no effect on adversaries, and a big one can impact on the application usability and annoy users. It is important to find a balance between these two options.

3.2. Levels of expertise

We also made an analysis of the tools according to the level of expertise required from users to accomplish certain tasks while using them. We made an estimation based on our point of view and experience using them. We created a categorization level to classify users based on the work of the National Institutes of Health [OHR 2009], presented next.

- **Level 0** - No knowledge on data security field at all.
- **Level 1 - Beginner:** Someone with a common knowledge or an understanding of basic techniques and concepts of data security. Able to perform simple tasks on computers, as managing files and folders, installing basic programs etc. However, they often need some help to deal with more complex issues.

- **Level 2 - Intermediate:** Users with a better and broader understanding of cryptography and computer systems. They have knowledge on some security algorithms, related tools, and are able to complete simple tasks by themselves. Some assistance is needed from time to time.
- **Level 3 - Advanced:** Someone with a degree in areas related to computers, with a solid knowledge about security and operating systems. He/She can perform tasks associated to the fields without assistance and will be recognized as the one to ask questions to. He/She is able to implement tools and algorithms on the field.
- **Level 4 - Expert:** People with a degree or deep knowledge on security, able to implement, develop and evaluate algorithms and tools to find out vulnerabilities.

We considered the following items for evaluating the tools during their use:

- **Installation process:** We installed each tool with default setting options (when possible), which is usually the simplest way. We evaluate how hard is for users to install them based on their knowledge, choosing between the options, and so on.
- **Configuration:** Once the tools are installed, some of them need to be configured in order to encrypt a disk partition or the whole disk. We evaluate how hard is for users perform this task.
- **Encryption/ Decryption:** Some tools require an interface to allow users to encrypt/decrypt their data, while others do it automatically. In the latter case, this operation is not considered at all. We evaluated these processes considering how intuitive and simple they are.
- **Data manipulation:** In this item, we evaluated the process of authentication and data access, considering how intuitive and simple it is.
- **Changing default settings:** We also evaluated the process of changing the default configuration. For example, some tools allow users to use multiple algorithms (AES, 3DES, Twofish, Blowfish etc), select different key sizes (128, 192, 256 bits), and modes of operation (ECB, CBC, XTS). Users are required to change the configuration manually. We evaluate how hard is for them to perform this task, or, for example, change their passphrase.
- **Enabling additional features:** This item evaluated how hard is for users to enable certain features available in the tools, such as the second factor of authentication, use of hidden containers etc.
- **Uninstallation process:** This is the last item evaluated. We took into consideration the removal process, considering how hard is to decrypted all users' data before removing the tool.

Table 2. Comparison of encryption tools regarding to security aspects

Tools	Algorithms	Key recovery	Alg. operation modes	Open source	Encryption subjects	Multi factor authentica-tion	Operating systems	Portable software	Plausible deniability	File Attributes Encryption	Security Levels	Hidden containers	TPM use	Key Stretching
AxCrypt	AES / RSA	X	CTR	✓	FE	X	Windows	✓	X	X	128 or 256 / 4096 bits	X	X	PBKDF2 - 10000 iterations or more
GnuPG	AES*, Cast5, 3DES, IDEA, Blowfish, TwoFish, Camellia / RSA*, DSA	X	CFB	✓	FE	X	Windows, Linux and Mac OS	✓	X	X	128 or 256 / 1024-4096 bits	X	X	X
7-Zip	AES	X	CBC	✓	FE	X	Windows, Linux and Mac OS	✓	X	Only filenames	256 bits	X	X	SHA-256 - 262144 iterations
R-Crypto	AES, DES, 3DES	X	?	X	VDE	X	Windows	X	X	✓	56-256 bits	X	X	X
DiskCryptor	AES, Serpent, Twofish (and combinations)	X	XTS	✓	VDE and FDE	✓	Windows	X	✓***	✓	256 bits	X	X	PBKDF2 - 1000 iterations
LUKS	AES*, Twofish, Serpent, Cast5, Cast6	✓**	*XTS, ECB, CBC (plain or ESSIV)	✓	VDE and FDE	✓	Linux	X	X	✓	256 bits	X	X	PBKDF2 - 1000 iterations
VeraCrypt	AES, Serpent, Twofish (and combinations)	X	XTS	✓	VDE and FDE	✓	Windows, Linux and Mac OS	✓	✓	✓	256 bit	✓	X	PBKDF2 - X iterations****
CipherShed	AES, Serpent, Twofish (and combinations)	X	XTS	✓	VDE and FDE	✓	Windows, Linux and Mac OS	✓	✓	✓	256 bits	✓	X	PBKDF2 - (1000 - 2000) iterations
BitLocker	AES	✓	CBC modified	X	FDE	✓	Windows	X	X	✓	128 or 256 bits	X	✓	X
FileVault 2	AES	✓	XTS	X	FDE	X	Mac OS	X	X	✓	128 bits	X	X	PBKDF2 - 41000 iterations

Observations: The mark (✓) represent the tools that implement the feature, while the X the ones that do not. The interrogation point (?) is used to mark the features that we could not find information related to their use or not. All options marked with the special character * are default choices. The options marked with ** allows users to add additional passphrases or keyfiles for the authentication. The ones with *** do not support plausible deniability but allow users to install the boot-loader on an external device, being possible to claim that the hard drive was wiped clean since there is no boot-loader neither identifiable information proving that it has been encrypted. The mark **** represents the number of iterations of VeraCrypt, consisting in two ways: system partition, with 200000, 327661 or PIM x 2048 iterations, and containers, with 655331, 500000 or 15000+(PIMx1000) iterations, where PIM (Personal Iterations Multiplier) is a field for users having more control over the number of iterations.

3.3. Comparisons

We choose a set of tools according to works that points out the best ones to encrypt data [Henry 2015] [Manes 2015] [Sharma 2016], considering the operating system (Macintosh, Windows and Linux) and the subject of encryption (FE, VDE and FDE). The tools chosen are still maintained and available for free use. Table 2 presents the comparison, evaluating security aspects. Another comparison evaluated the level of expertise required to use the tools' different features (Table 3).

Table 3. Minimum level of expertise required for different features of the tools

Tools	Installation	Data manipulation	Configuration	Encryption/ Decryption	Changing default setting	Enabling additional features	Uninstallation
AxCrypt	1	2	-	2	1	2	1
GnuPG (Windows version)	1	2	-	2	1	-	1
GnuPG (Linux version)	-	2	2	3	3	-	2
7-Zip (Windows version)	1	2	-	2	-	-	1
R-Crypto	1	1	2	-	2	-	1
DiskCryptor	1	1	2	-	1	1	1
LUKS (VDE)	2	1	1	-	2	3	2
LUKS (FDE)	1	1	1	-	2	3	2
VeraCrypt (Windows version)	1	1	2	-	1	1	1
CipherShed (Windows version)	1	1	2	-	1	1	1
BitLocker	-	1	2	-	1	1	1
FileVault 2	1	1	1	-	1	-	1

4. Discussion

Our security analysis shows users how to choose the tools that best fit their needs. FDE method is the least vulnerable encryption mode, but this is true only when computers are off. Once users turn on their devices and enter their passwords, all data will be decrypted.

The analysis allowed us to observe some good trends. First, most VDE and FDE tools adopt the XTS encryption mode, recommended for disk data storage with no vulnerabilities known so far. Besides that, we can see that 70% of the tools are open source, a tendency we hope will increase since it is a necessary requirement with respect to secure software. The file attributes encryption is also a matter taken into consideration and available in all VDE and FDE tools. The maximum security level is also available for all tools except FileVault 2, which works with AES 128 bits keys.

However, there are some security aspects to improve. The first one is related to the multi-factor authentication, offered by only 50% of the tools. As it can minimize the impacts of choosing weak passwords, it adds an extra protection layer to the system, and should be more explored. Also, only half of tools have a portable version, which besides being easier to run (does not require installation), does not leave traces in the machine. This and some other features as plausible deniability (30%), hidden containers (20%) and TPM (10%), also need to be more explored, since they offer additional security services that should at least be available for users. Also, we point out that more FE tools should adopt the file attributes encryption, because only one tool of this kind use it (but protecting only filenames). The key stretching algorithm is another point for improvement. Although 70% of the tools adopt this feature, 30% use it ineffectively (small number of iterations). Besides, there are new algorithms such as Argon2, Lyra2 etc., that could better protect users since the PBKDF is becoming obsolete by these password hashing algorithms.

The key recovery feature is a point to be explored by adversaries, being addressed by most tools (70%). We noticed that BitLocker enables users to store the recovery key in their Microsoft account. This can reduce the security, since users tend to create shorter and easier to remember passwords for this kind of account. This way, adversaries can focus on breaking users' accounts to get the recovery keys. However, for some users this could release them from the burden of losing their keys.

From our analysis and criteria, the most secure tool is VeraCrypt (satisfies all criteria except the use of TPM). CipherShed has a similar result, but its key stretching algorithm has a smaller number of iterations. On the other hand, the least secure one is R-Cript. However, we do emphasize that even when the best tools are used, it is necessary that users take some precautions, such as using strong passwords (upper and lower case letters, numbers, special characters etc, with a long length - 20 characters or more are recommended) and managing them appropriately. Another precaution is related to malicious applications on the device. None of the tools analyzed was designed to protect users data in a hostile environment, and in such cases, the whole security could be compromised. This threat is not in the scope of this paper and could be addressed in a future work.

In the analysis of the minimum level of expertise needed to use the tools, we see that they are becoming easier to use. What once was a restricted tool for military and experts only, has now reached average users, allowing them to enjoy the benefits of privacy with little effort and no costs. We have observed that tools designed for Linux systems are more complex and demand more effort or knowledge from users, as the case of GnuPG and LUKS. For Macintosh and Windows, we have easier options, as FileVault 2, 7-Zip, and BitLocker, with low level of expertise required. This statement is sustained by the comparison of GnuPG in Windows and Linux version, where in the former we have a lower required level of expertise in most activities. In our analysis, FileVault 2 is the easiest tool compared to the others, while GnuPG (Linux) is the hardest to use.

It is important to emphasize that some tools, once configured, do not require extra effort to encrypt data. Most VDE or FDE tools work in this way, requiring only that users store their files in a specific place (container or partition) or anywhere in the current operating system to encrypt them. Cryptography has little impact on users, working in the background. On the other hand, most FE tools require the use of an interface to perform the encryption. Also, some tools are already installed in the operating system (BitLocker in specific versions of Windows and GnuPG for Linux), making them easier to use.

5. Conclusion

With so many encryption tools available today, due to the wide spread of simple and easy to use cryptographic tools, users face a challenge of choosing the ones that best protect their data. An analysis of these tools from a forensics point of view was done with the purpose of pointing out strengths and weakness to help users differentiate them. We presented different ways of using cryptography to protect data on the disk, pointing out some vulnerabilities. We showed that FE and VDE methods have a broader scope of attacks in which one can get useful information about users. On the other hand, the FDE mode is the safest one as long as the computer is off. Only those who know the password can decrypt data and access the system. We also presented particular features of disk encryption tools and evaluated a set of them regarding security aspects, showing important features that can help users to choose the tools that best fit their needs. The other evaluation was related to the minimum level of expertise required to use different features during tool operation. We concluded that there are free and easy to use tools that can protect users data when basic precautions are taken. As future work, we intend to extend the number of tools and items analyzed, evaluating criteria as the whole key management process, the PRNG (Pseudo Random Number Generator) algorithms and the performance in each one of the tools.

Acknowledgment

The authors would like to thank the FORTE project and CAPES (23038.007604/2014-69) for the partial financial support.

References

Apple (2011). Use FileVault to encrypt the startup disk on your Mac. `https://support.apple.com/en-us/HT204837`. Accessed 2016 jun 07.

Aviram, N., Schinzel, et al. (2016). Drown: Breaking TLS using SSLv2. `https://drownattack.com/drown-attack-paper.pdf`. Accessed 2016 jun 07.

AxCrypt (2001). AxCrypt. `http://www.axcrypt.net/`. Accessed 2016 jun 07.

Balogh, Š. and Pondelik, M. (2011). Capturing encryption keys for digital analysis. In *IDAACS, 2011 IEEE 6th International Conf. on*, volume 2, pages 759–763. IEEE.

C., S. (2010). Laptop/table encryption. `https://answers.uchicago.edu/page.php?id=15736`. Accessed 2016 May 09.

Canetti, R., Dwork, C., Naor, M., and Ostrovsky, R. (1997). Deniable encryption. In *Advances in Cryptology-CRYPTO'97*, pages 90–104. Springer.

Casey, E. and Stellatos, G. J. (2008). The impact of full disk encryption on digital forensics. *ACM SIGOPS Operating Systems Review*, 42(3):93–98.

Codenomicon (2014). Drown: Breaking tls using sslv2. `http://heartbleed.com/`. Accessed 2016 jun 07.

Dworkin, M. (2010). Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices. In *NIST Special Publication*.

Fruhwirth, C. (2005). New methods in hard disk encryption. `http://clemens.endorphin.org/nmihde/nmihde-letter-os.pdf`. Accessed 2016 Jun 07.

GNU, P. (1999). The GNU Privacy Guard. `https://www.gnupg.org/`. Accessed 2016 jun 07.

Guardian, P. (2014). Luks: Linux Unified key Setup. `https://gitlab.com/cryptsetup/cryptsetup`. Accessed 2016 jun 07.

Gupta, D. and Mehtre, B. M. (2013). Recent trends in collection of software forensics artifacts: Issues and challenges. In *Security in Computing and Communications*, pages 303–312. Springer.

Halderman, J. A., Schoen, S. D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J. A., Feldman, A. J., Appelbaum, J., and Felten, E. W. (2009). Lest we remember: cold-boot attacks on encryption keys. *Communications of the ACM*, 52(5):91–98.

Henry, A. (2015). Five best file encryption tools. `http://lifehacker.com/five-best-file-encryption-tools-5677725`. Accessed 2016 jun 08.

IDRIX (2014). Veracrypt. `https://veracrypt.codeplex.com/`. Accessed 2016 jun 07.

IDRIX (2015). Veracrypt user's guide. `http://cyberside.net.ee/veracrypt/VeraCrypt%20User%20Guide.pdf`. Accessed 2016 Jun 01.

Kessler, G. C. (2007). Anti-forensics and the digital investigator. In *Australian Digital Forensics Conference*, page 7. Accessed 2016 Apr 22.

Kornblum, J. D. (2009). Implementing bitlocker drive encryption for forensic analysis. *digital investigation*, 5(3):75–84.

Lowman, S. (2010). The effect of file and disk encryption on computer forensics. Accessed 2016 Apr 22.

Manes, C. (2015). The top 24 free tools for data encryption. `http://www.gfi.com/blog/the-top-24-free-tools-for-data-encryption/`. Accessed 2016 jun 08.

Martin, L. (2010). Xts: A mode of aes for encrypting hard disks. *IEEE Security & Privacy*, (3):68–69.

Microsoft (2006). Bitlocker. `https://technet.microsoft.com/en-us/windows/bitlocker-and-bitlocker-to-go`. Accessed 2016 jun 07.

Mrdovic, S. and Huseinovic, A. (2011). Forensic analysis of encrypted volumes using hibernation file. In *TELFOR, 2011 19th*, pages 1277–1280. IEEE.

NTLDR (2007). Diskcryptor: Open source partition encryption solution. `https://diskcryptor.net/wiki/Main_Page`. Accessed 2016 jun 07.

OHR (2009). Proficiency scale. `https://hr.od.nih.gov/workingatnih/competencies/proficiencyscale`. Accessed 2016 Jun 01.

Pavlov, I. (1999). 7-zip. `http://www.7-zip.org/`. Accessed 2016 jun 07.

PCM, P. M. C. (2014). Ciphershed: Secure Encryption Software. `https://www.ciphershed.org/`. Accessed 2016 jun 07.

R-Tools, T. I. (2007). R-crypto: Data security for windows. `http://www.r-tt.com/data_security_software/`. Accessed 2016 jun 07.

Sharma, R. (2016). 7 Best Encryption Software for Windows. `http://beebom.com/best-encryption-software-windows/`. Accessed 2016 jun 08.

Zhang, L., Zhou, Y., and Fan, J. (2014). The forensic analysis of encrypted truecrypt volumes. In *PIC, 2014 International Conference on*, pages 405–409. IEEE.