ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

Examining VeraCrypt Security: Finding Hidden Volumes Using Reverser Engineering Techniques

Zakariyya Hassan Abdullahi[1]; Kabiru Bashir[2]; Yunusa Ibrahim[3]

[1,3]Department of Computer Engineering, Hussaini Adamu Federal Polytechnic Kazaure Jigawa Nigeria
[2]Department of Mechanical Engineering, Kano State Polytechnic Kano Nigeria

Publication Date: 2026/08/18

Abstract: Data software that encrypts information is being used by more and more people to protect data secrecy. The freely downloadable and open-source programme known as VeraCrypt is one tool that supports data encryption. It can be difficult for a digital forensic investigator to even find encrypted data. The paper explores the worst-case situation when a memory seizure or access to the data in a decrypted state is not conceivable and the forensic detective only has access to the suspicious computer's hard disc after the device has been off for a significant amount of time. It starts by assessing the effectiveness of current statistical tests in distinguishing between encrypted VeraCrypt data and other non-encrypted data. By analysing the raw byte data content of the suspicious hard disc, a forensic investigator could utilise a specific process model to locate the encrypted data. The secret operating system and volume, however, continue to be invisible. The research paper comes to the final conclusion that it is still difficult for forensic investigators to detect the concealed volume system only from an investigation of the hard disc of the suspect machine.

Keyword: *Vera Crypt, Encryption, Forensic Investigation, Reverse Engineering.*

How to Cite: Zakariyya Hassan Abdullahi; Kabiru Bashir; Yunusa Ibrahim (2026) Examining VeraCrypt Security: Finding Hidden Volumes Using Reverser Engineering Techniques. *International Journal of Innovative Science and Research Technology*, 11(8), 813-826. https://doi.org/10.38124/ijisrt/26aug189

I. INTRODUCTION

Information technology and digital gadgets have been an integral part of our daily life ever since the first computers were invented. Digital technologies and electronic devices are also being abused by cybercriminals and attackers [1]. Due to the growing illicit use of computers, digital forensic analysis has become a crucial component of criminal investigations [2]. Digital forensics has become increasingly difficult due to Increased storage, bandwidth, communication speed, device networks, and mobile devices [3]. Cryptography is an important method for protecting data from hackers, involving encryption and decryption. Encryption involves converting encrypted material to plain text while leaving all of the original text's words intact [4].

A growing number of firms are using whole-disc encryption software due to concerns over the security of data held on lost or stolen laptops. The forensic investigation of such encrypted material is then complicated by a variety of challenges as a result. The first concern is that the analyst cannot access this evidence if the keys cannot be found, which is obviously a problem. This issue has been studied in the past, and solutions include using live forensics [3], [5].

Some encryption methods allow the creation of two keys for a volume[1], one unlocks a hidden volume containing sensitive content, while the other decrypts a separate set of innocent data (cover volume). This research focuses on the difficulties faced in digital forensic investigations when dealing with hidden encrypted volumes. It explores the functioning of concealed encrypted volumes, which involve the use of two keys to decrypt the hidden and cover volumes respectively. The existence of these volumes can complicate digital examinations and pose challenges for digital investigations [6],[7]. This article examines at the VeraCrypt[2] software and offers an effective solution for the problem with identifying hidden encrypted volumes.

In today's digital era, encryption software plays a crucial role in ensuring data security. There are various encryption programs available publicly and license-based along with unique features and capabilities, all are aimed at effectively protecting sensitive information.

[1] In a Vera crypt, volume refers to a logical container that can hold files and other data. This container is created by encrypting a portion of a physical disc or a file with an encryption algorithm such as AES, Serpent, or Twofish.

[2] VeraCrypt is an encryption software which uses some strong encryption algorithms such as AES, Serpent, and Two fish, which are considered to be secure.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

BitLocker is a native encryption tool available in Windows, but is not an open-source tool. TrueCrypt was a popular open-source encryption software, but its discontinuation raised concerns about security. GNU Privacy Guard (GnuPG) is a free and open-source encryption tool, but its command-line interface may not be user-friendly for beginners[8]. AxCrypt is a free encryption tool that offers only file encryption and password management features [9],[10],[11],[12].

Table 1 Comparison Among Different Types of Encryption Software

Encryption Software	Type	Platform
TrueCrypt, 2014 [9]	Discontinued	Windows, Linux, Mac
GNU Privacy Guard, 2015 [10]	File encryption and email encryption	Windows, Linux, Mac
AxCrypt, 2017 [10]	File encryption	Windows
BitLocker, 2020 [13]	Full-disk encryption	Windows
ProtonMail, 2021 [12]	Email encryption	Web, iOS, Android

As per table 1, Compares various encryption software types: TrueCrypt, GNU Privacy Guard, AxCrypt, BitLocker, and ProtonMail for Windows, Linux, Mac, and email encryption. [13]. GnuPG has a command-line interface, a steep learning curve, and may have compatibility issues with other encryption tools. AxCrypt has limited encryption options, is not open source, and uses a subscription model. TrueCrypt has been abandoned, is difficult to use, and has limited platform support. Proton Mail has limited functionality, relies on third-party providers, and is closed-source [8],[14]. Therefore, to deal with these kinds of issues in this article VeraCrypt software is discussed in detailed along with-it working process.

High data security is provided by VeraCrypt's safe and user-friendly encryption mechanism. Strong encryption algorithms that provide high data security by allowing users to construct hidden volumes inside encrypted volumes, also an effective and secure encryption tool that offers a wide range of functions not found in other encryption software [14].

In digital investigations, there are different ways to try to access locked information. Following ways are explained in articles [15] [16] [17]. and can be simplified as:

- Getting the suspect to give the key
- Finding unencrypted copies of the data
- Finding the keys or passwords
- Smart attempts to guess passwords
- Trying all possible keys
- Using weaknesses in how it's made
- Watching the hardware or software.

➢ *Various Kinds of Attacks*

- *Brute-Force Attack:*

The most important idea is to use strong passwords, two-factor authentication, and physical security measures to protect against brute-force attacks on VeraCrypt. [39]. Most of these attacks are caused by malicious software, as observed in past incidents. To ensure the security of virtual spaces, it is crucial to promptly detect and address such security breaches. By employing methods that distinguish between desirable and undesirable software behaviours, we can effectively identify problematic software and detect malware before it leads to harmful outcomes. To ensure the accuracy of the detection process, it is important to evaluate both types of software [18].

- *Dictionary Attack:*

In a dictionary attack, an attacker uses a precomputed list of commonly used passwords, known as a dictionary, to try and crack the encryption. This attack is effective against weak or easily guessable passwords. To protect against dictionary attacks, it is crucial to use strong, unique passwords that are not easily found in dictionaries [19].

- *Known-Plaintext Attack:*

In a known-plaintext attack, the attacker has access to both the encrypted data and the corresponding unencrypted plaintext. The goal is to discover the encryption key or vulnerabilities in the encryption algorithm. VeraCrypt is designed to be resistant to known-plaintext attacks and uses strong encryption algorithms that are widely trusted [20].

- *Timing and Side-Channel Attacks:*

It can be exploited to gain information about the encryption key, but VeraCrypt has countermeasures to mitigate them [21].

- *Rubber Hose Cryptanalysis:*

This attack involves physically coercing the user into revealing the password or encryption key. VeraCrypt cannot protect against this type of attack directly, as it relies on human behaviour. It is crucial to keep your password or key confidential and not disclose it to anyone under duress [22].

Figure 1, Shows an Increase in Malware Volume Between 2016 and 2022. It Could See an Increase in Assault Activity in the Next Years.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

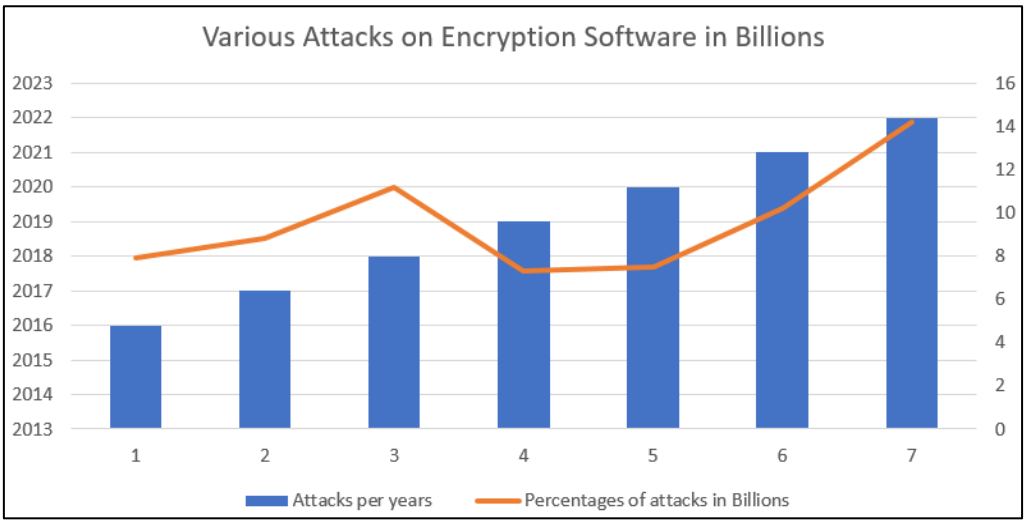


Fig 1 Various Attacks on Encryption Software

Figure 1.Number of Encryption attacks source, published by Ani Petrosyan, May 3, 2023(During 2022, the worldwide number of malware attacks reached 5.5 billion, an increase of two percent compared to the preceding year. In recent years, the highest number of malware attacks was detected in 2018, when 10.5 billion such attacks were reported across the globe.

➢ *Novel Contribution of this Article:*

This research paper explores hidden encrypted volumes in digital investigations, addressing challenges and features. It highlights the need for reverse engineering techniques and forensic operations to verify VeraCrypt's effectiveness in real-world situations.

This examination explores the use of VeraCrypt, a software designed to protect user data from unauthorized access, and its hidden volume and Examine feature. It emphasizes the need for forensic operations to verify VeraCrypt's effectiveness in data protection. The paper also highlights the importance of utilizing reverse engineering techniques to enhance software security and ensure user data protection. Despite its benefits, rigorous testing and documentation are crucial for its effectiveness.

II. RELATED WORK

The article focuses on how criminal justice agencies, are facing an increased need to investigate crimes committed over the Internet or other electronic media, requiring resources and procedures to search for, locate, and preserve electronic evidence[23].The study discovered, that TrueCrypt makes it challenging to use built-in utilities, by not supporting encrypting full system drives, dynamic CDs, or the Windows Mount Manager under Windows 2000 [24]. This paper discusses how full disk encryption (FDE) can hinder digital investigations, and how circumventing FDE has benefited certain cases. It also provides guidance for gathering items at the crime scene and performing on-scene forensic acquisitions of live computer systems to increase the chances of acquiring digital evidence in an unencrypted state [25].

Digital evidence can help prosecutors win more convictions, but there are challenges such as lack of personnel resources, lack of training, and civil liberty concerns, effective use requires understanding the importance of devices and what information is captured [26],[27]. Exploring TrueCrypt's flaws at multiple levels failed to identify a decryption technique[28]. The article outlines two methods attackers might use VeraCrypt to increase their privileges, however neither method successfully decrypts encrypted data [8].

Tvrdik et al., (2017) focused on analyzing the security and reverse engineering of the disc encryption used by the Best Crypt Volume Encryption application. Additionally, it provides a thorough explanation of a previously unidentified binary file format used for rescue operations. Several flaws and issues were found after conducting a thorough security analysis [29]. Researchers examined the full-disc encryption features of numerous self-encrypting drives (SEDs) made by various vendors by reverse engineering the firmware. About 50% of the current SSD market is made up of these vendors. The research team found that the drives under examination have major security flaws. It is sometimes possible to entirely get around the encryption and retrieve the drive's data without knowing any secret codes or passcodes [30]. The article finds that reverse engineering is an important process for understanding and enhancing older systems. It involves using various methods, such as transformational approaches, program comprehension theories, and reverse engineering technologies [31].

Reverse engineering (RE) is the process of duplicating an existing product to create a product knowledge library for engineering design [32],[33],[34]. The article outlines a plan for exploring the area of data and code reverse engineering over the first decade of the 21st century. It also discusses approaches for creating and assessing tools to aid in this field. The subject of reverse engineering is a stimulating topic that could be included in academic programs for computer science, computer engineering, and software engineering [34].The importance of techniques to hide data cannot be overstated in the field of computer forensics. As technology

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

advances, new methods for concealing data will inevitably emerge. The appeal of hiding data lies in the creative or technical skills of the person attempting to hide it [35].

Balducci et al., (2015) have concentrated on the efforts of data forensics to discover a means of decrypting TrueCrypt by scrutinizing its vulnerabilities. However, despite the extensive investigation, only four weaknesses have been identified thus far [36].The article discusses a crucial distinction where investigators must maintain their motivation and curiosity to access coded disc partitions and volumes [37]. Brinkmann et al., (2021) Iinvestigated TrueCrypt volume search techniques by examining the primary boot record of the hard drive and the registry of the operating system. Meanwhile, the Ubuntu Privacy Remix Team addressed matters related to privacy [7]. The articles disclosed a vulnerability in the derivation of the header key. Zhang provided a technique for recovering passwords based on the type of encrypted file volume. These initiatives represent TrueCrypt's earliest commercialization efforts in the field of forensics [38].

Zhang et al. ,(2019) have evaluated TrueCrypt's encryption and data recovery capabilities from the standpoint of data forensics, with a focus on techniques such as key file obfuscation, data extraction, and password cracking [39]. The article covers several reverse engineering methods and tools for conducting effective malware analysis. A productive patch was employed to develop a valuable Yara rule that would assist analysts in constructing a robust defence system against harmful binaries. Machine learning or artificial intelligence techniques will be utilized to detect emerging threats [40]. The research paper explains how a disassembly technique that employs machine learning divides an x86 binary into smaller byte units and categorizes each cluster as either code or data [41].

Stroulia and Systa.(2002) have examined the dynamic behavior, of a system to understand its operations and applications [42].The articles elucidate that Cross PDE is a system designed for mobile devices that operates across multiple layers to protect confidential data from coercion attacks. It intervenes at the primary levels of a mobile storage system to prevent the loss of sensitive information and allows users to access a concealed mode. Empirical research indicates that Cross PDE can maintain deniability while incurring only a minor decrease in throughput [43]. The paper emphasizes the importance of utilizing dependable and thorough forensic tools to successfully counter anti-forensics techniques [44].

Meijer and Gastel, (2019) have found flaws in SSD and BitLocker encryption, requiring users to take extra precautions[45].The article investigates the technology utilized by self-encrypting drives (SEDs) and the BitLocker tool for full disc encryption (FDE) in Windows. It also explores threat models and attacks on disc encryption in the context of BitLocker drives' SEDs that support the Opal standard with Windows. Furthermore, it assesses the open-source software that enables access to the BitLocker drive format in Windows or Linux [45]. The paper explains that reverse engineering of file systems is important for verifying the effectiveness of tools, collecting accurate evidence, and interpreting data structures in criminal investigations. It presents a position on the technological and legal challenges that arise from the use of reverse engineering for law enforcement purposes [46]. The process of gaining access to encrypted evidence during a digital investigation involves several techniques, which are discussed in the article [15],[17].

Table 2 Summary of Related Work

Author/References	Approaches/ Methods	Limitations	Remarks
Wong et al., 2000 [11]	focuses on software reverse engineering and redocumentation.	involves re-documenting existing software architectures.	facilitating ongoing software understanding
Stroulia et al., 2002 [16]	Dynamic reverse engineering techniques methods	Focusing on program understanding on modelling the structurer of a program by examining its code	Complexity of the software systems being developed increases the complexity of the systems that need to understood also increases
Steven et al., 2011 [20]	Models based on a critical heuristic for identifying code	Discovering Feature Groupings, Creating the Feature Hierarchy,	In the field of feature location, there are a lot of unresolved issues.
Antunes et al., 2012 [17]	Methodology Was Implemented FTP Protocol	Does not require any access to a protocol implementation or its source code	To extending the methods to, support binary files
Gonzalez et al., 2015 [16]	String Offset Order	Works on Android Malware Genome Project Apps	Works on Android Apps
Debray et al., 20015 [18]	Cloning dynamic analyses Methods	Regarding techniques for understanding obfuscated code and the strengths and weaknesses of sophisticated obfuscation algorithms	Malware suspicious

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

David et al., 2017 [11]	static and dynamic detection techniques	Focus on a blind detection on a collection of suspicious software	Malware suspicious
Michael et al., 2018 [34]	Graph analysis methods.	Software ranks code snippets in the top k results using learned schema.	Utilizing Cryptographic Architecture.
Kaizheng et al., 2020 [32]	Manual Reverse Engineering Framework	device employs secure boot and the firmware image verification key is in secure storage such as e-fuse,	Some software programmes have relatively constrained methodologies.
Gorecki et al., 2020 [32]	anti-forensics techniques	Anti-forensics tools needed to combat evolving techniques.	lacks explanation of how cybercriminals use the tools
Bajracharya et al., 2021 [22]	Code Rank approach	Utilises the Fundamental Coder-Ank Notation, Which Exclusively Retrieves Structural Data.	Seeing strong power-law behavior
Ahmed et al., 2022 [31]	NTFS file system	insufficiently addresses issues with methodology disclosure	undermines the scientific validity of the digital forensic process.
Maqbool et al., 2023 [21]	Aapproach uses a machine learning approach to train the schema	software ranks code snippets in the top k results using learned schema.	Enhancing tool performance requires the use of new capabilities.

III. METHODOLOGY

Exploiting the security of VeraCrypt can pose difficulties due to its robust encryption and defending mechanisms against different attack types. Reverse engineering is a method for the scrutinizing software systems by analysing their source codes, structures, and functionality, which can be used to reveal concealed (encrypted) volumes or potential security flaws in VeraCrypt [33]. Analysing the security of the VeraCrypt can be a challenging task, as it is designed to provide strong encryption and protection against various types of attacks. In this article, it is suggested that reverse engineering can be employed to reveal hidden volumes and identify potential security weaknesses. The methodology for examining the VeraCrypt security using reverse engineering techniques could involve the following steps.

➢ *Acquiring the VeraCrypt Ssoftware*

The first step is to acquire the VeraCrypt software from the official website or a trusted source. It is important to ensure that the software version being analysed is the latest and legitimate one. As Fig. 2 shows the legitimate software and process to enter the license key and many more.

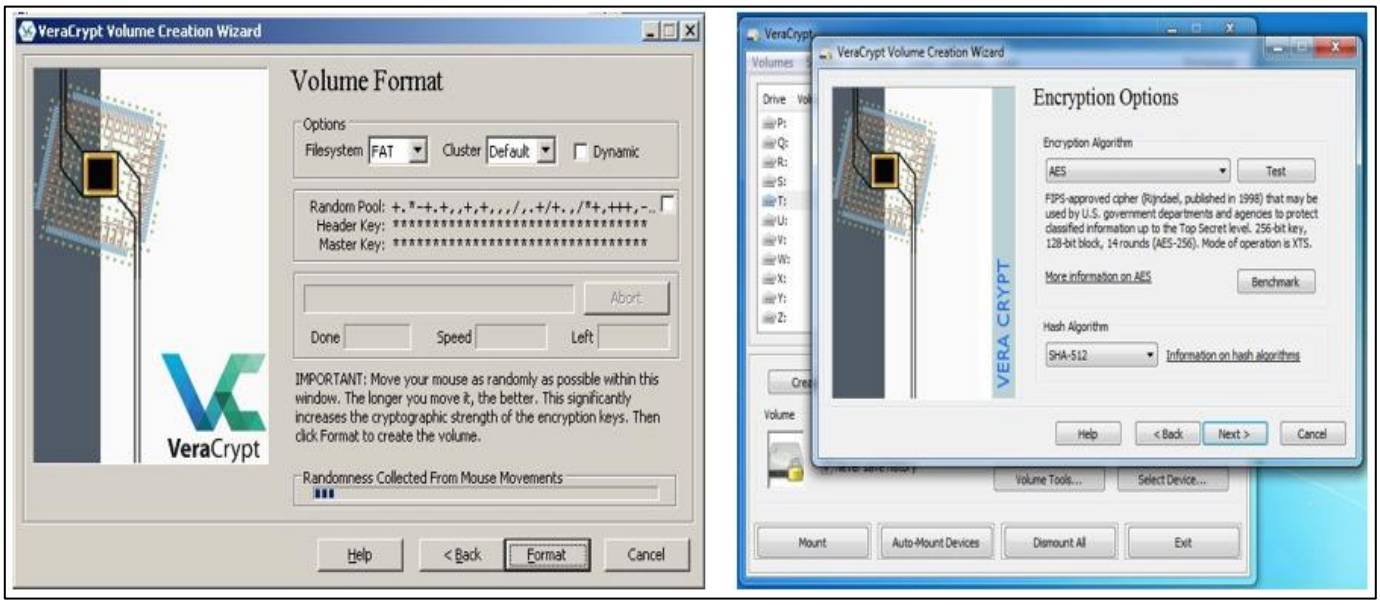


Fig 2 Shows the VeraCrypt Software and it's Use, a) Making code Based on the Movement of the Mouse b) VeraCrypt Software (Sources https;//www.idrix.fr)

➢ *Reverse Engineering the VeraCrypt Code*

Reverse engineering involves analysing the binary code of the VeraCrypt software to identify the algorithms are used for encryption, key generation, and other security features. Tools such as disassemblers and decompiles can be used to convert the binary code into a more readable format.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

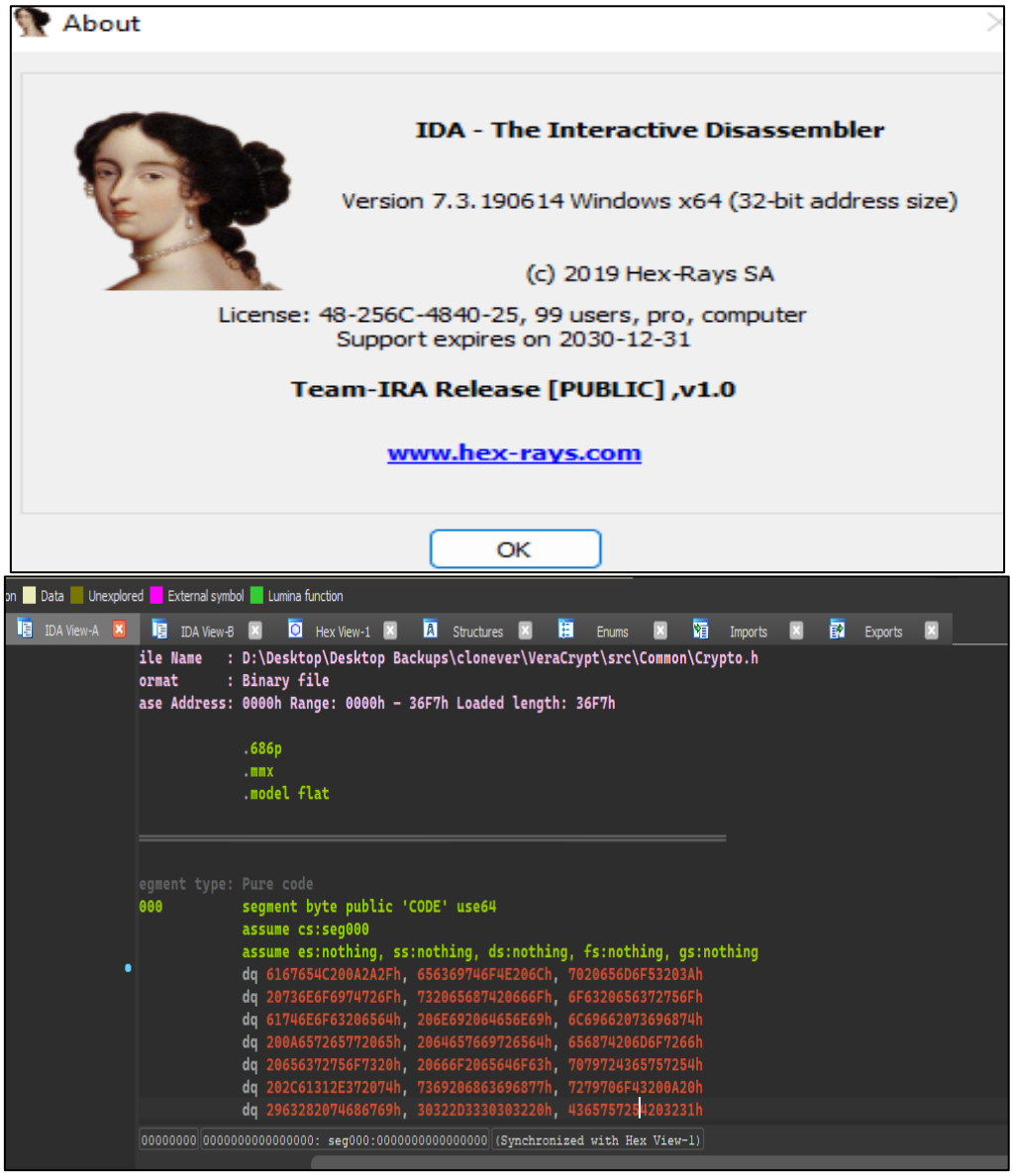
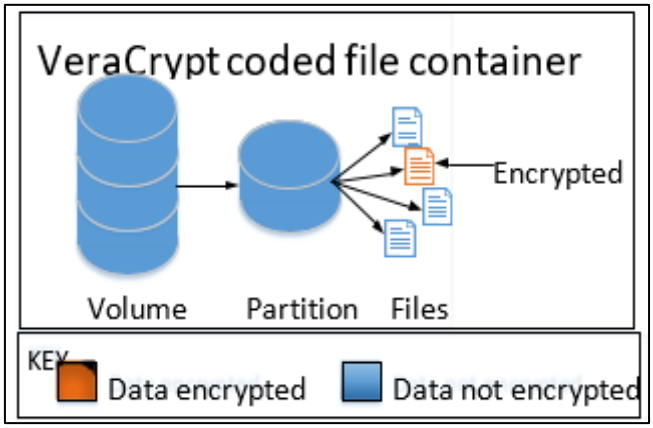


Fig 3 a) IDA Pro Reverser Engineering Tools, b) IDA Pro Disassemble VeraCrypt Binary Hex View

➢ *Identifying Hidden Volumes:*

Hidden volumes are a key security feature of VeraCrypt, and their detection can be a challenge. Reverse engineering methods can help detect hidden volumes by examining certain indicators, like unused storage areas or sections of code referencing them.



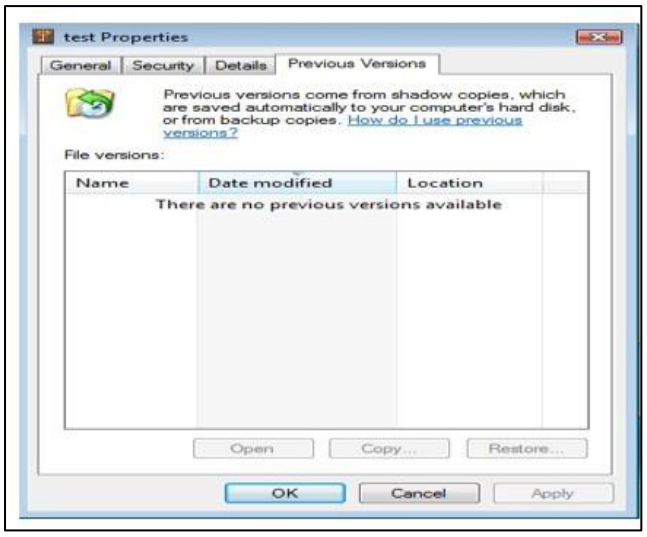


Fig 4 a) VeraCrypt Encrypted File Container b) Versions Visible for a VeraCrypt Container

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

In the context of VeraCrypt, "Versions Visible" refers to a feature that allows you to specify which versions of the VeraCrypt software are capable of mounting and accessing a particular encrypted container. When creating a VeraCrypt container, you have the option to select the "Versions Visible" parameter, which determines the compatibility of the container with different versions of VeraCrypt. This parameter specifies the minimum and maximum versions of VeraCrypt that can access the container.

➢ *Obtaining Multiple Copies of an Encrypted Container*

The Volume Shadow Copy feature in Windows version, enables creating multiple encrypted containers, extending the Restore Point feature. This process involves booting a virtual machine, listing Restore Points, mounting them, and copying them to blank media. VeraCrypt encrypted containers are included in automatic backups, but not accessible through the GUI [5][47].

- *Analysing Encryption Algorithms:*

VeraCrypt uses various encryption algorithms, such as AES, two fish, and Serpent. But with the help rreverse engineering anyone can be used to examine the implementation of these algorithms and identify any weaknesses or vulnerabilities.

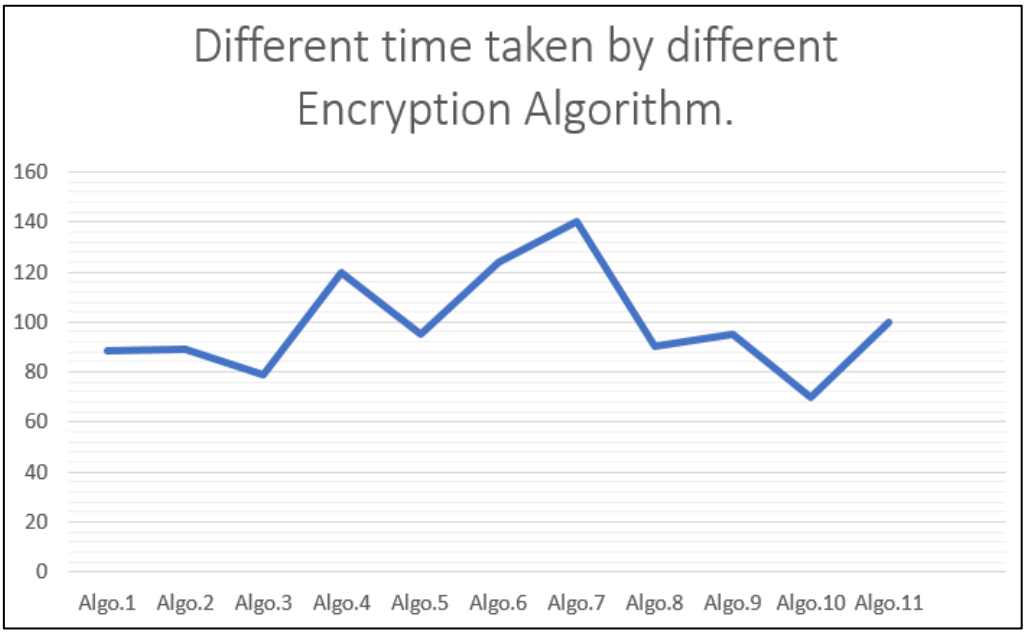


Fig 5 VeraCrypt's Encryption Algorithms, Present Different Kinds of Encryption Algorithms.

The figures 2–5 present a comprehensive process for assessing VeraCrypt's security and uncovering hidden volumes. It starts by obtaining the software and utilising reverse engineering techniques to access the source code and binary information. This iterative process continues until the crucial components, such as encryption algorithms and other sensitive functions, are successfully identified and thoroughly analysed. Where Algo1 to Algo11, Stand as follows.

- Algo.1 AES (Advanced Encryption Standard)
- Algo. 2 Serpent
- Alog. 3 Twofish
- Algo. 4 Camellia
- Algo. 5 AES-Twofish,
- Algo. 6 AES-Twofish-Serpent,
- Algo. 7 Camellia-Kuznyechik,
- Algo. 8 Camellia-Serpent, K
- Algo. 9 uznyechik-AES, Kuznyechik-
- Algo. 10 Serpent-Camellia,
- Algo. 11 Kuznyechik-Twofish

IV. RESULTS AND DISCUSSIONS

VeraCrypt, a free and open-source software, allows users to encrypt entire drives or specific volumes. It is widely used, reliable, and easily accessible. The research suggests that encrypted volumes that are not hidden can be securely concealed within VeraCrypt [23]. VeraCrypt offers several encryption options for data security. It utilizes encryption schemes such as AES, Serpent, Twofish, Camellia, Kuznyechik, and combinations of these algorithms in cascade mode. Some of the combinations include AES-Twofish, AES-Twofish-Serpent, Camellia-Kuznyechik, Camellia-Serpent, Kuznyechik-AES, Kuznyechik-Serpent-Camellia, and Kuznyechik-Twofish as shown in Fig. [12].

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

```
68
69      // Encryption algorithm configuration
70      static EncryptionAlgorithm EncryptionAlgorithms[] =
71      {
72          //  Cipher(s)                       Modes                       FormatEnabled
73
74      #ifndef TC_WINDOWS_BOOT
75
76          { { 0,                          0 }, { 0, 0},       0, 0 }, // Must be all-zero
77          { { AES,                          0 }, { XTS, 0 },    1, 1 },
78          { { SERPENT,                    0 }, { XTS, 0 },    1, 1 },
79          { { TWOFISH,                    0 }, { XTS, 0 },    1, 1 },
80          { { CAMELLIA,                   0 }, { XTS, 0 },    1, 1 },
81      #if defined(CIPHER_GOST89)
82          { { GOST89,                     0 }, { XTS, 0 },    0, 0 },
83      #endif  // defined(CIPHER_GOST89)
84          { { KUZNYECHIK,             0 }, { XTS, 0 },    0, 1 },
85          { { TWOFISH, AES,               0 }, { XTS, 0 },    1, 1 },
86          { { SERPENT, TWOFISH, AES,  0 }, { XTS, 0 },    1, 1 },
87          { { AES, SERPENT,               0 }, { XTS, 0 },    1, 1 },
88          { { AES, TWOFISH, SERPENT,  0 }, { XTS, 0 },    1, 1 },
89          { { SERPENT, TWOFISH,       0 }, { XTS, 0 },    1, 1 },
90          { { KUZNYECHIK, CAMELLIA,       0 }, { XTS, 0 },    0, 1 },
91          { { TWOFISH, KUZNYECHIK,        0 }, { XTS, 0 },    0, 1 },
92          { { SERPENT, CAMELLIA,      0 }, { XTS, 0 },    0, 1 },
93          { { AES, KUZNYECHIK,        0 }, { XTS, 0 },    0, 1 },
94          { { CAMELLIA, SERPENT, KUZNYECHIK,  0 }, { XTS, 0 },    0, 1 },
95          { { 0,                          0 }, { 0,    0},    0, 0 }      // Must be all-zero
96
97      #else // TC_WINDOWS_BOOT
98
99          // Encryption algorithms available for boot drive encryption
100         { { 0,                      0 }, { 0, 0 },      0 },    // Must be all-zero
101         { { AES,                    0 }, { XTS, 0 },    1 },
102         { { SERPENT,                0 }, { XTS, 0 },    1 },
```

Fig 6 Encryption Algorithms

This code snippet, which appears in Fig. 6, is part of a software implementation related to encryption and cryptography. It includes various header files and defines several structures, functions, and configurations for different cyphers and encryption algorithms. The code segment displayed above seems to be a part of the VeraCrypt disk encryption program [35]. It consists of predefined data structures that define settings for various encryption methods and cyphers, such as AES, Serpent, Twofish, Camellia, GOST89 (if defined), and Kuznyechik. These codes are utilised in XTS[3] mode and combined differently to form the encryption algorithms. The data structures specify the cypher, mode, block size, key size, and key schedule size for each encryption technique. Additionally, the code includes references to header files that contain encryption-related functions, including block functions for encryption and decryption [35]. Let's break down the code and understand its components:

- The code includes several header files, such as Tcdefs.h, Crypto.h, Xts.h, Crc.h, Common/Endian.h, string.h, and possibly EncryptionThreadPool.h. These headers provide necessary definitions, function prototypes, and declarations for the code.
- The code contains compiler directives, such as #if, #ifndef, and #pragma, which conditionally include or disable certain code based on specific conditions. For example, some code blocks are specific to non-UEFI environments or non-Windows boot scenarios.
- There are definitions for different ciphers and their properties, including block size, key size, and key schedule size. The cipher configurations are stored in an array called Ciphers, which lists different ciphers like AES, Serpent, Twofish, Camellia, GOST89, and Kuznyechik.
- The code defines encryption algorithm configurations in an array called EncryptionAlgorithms. Each entry specifies the combination of ciphers and modes to be used for encryption. It also indicates whether the format is enabled or not.
- In non-Windows boot scenarios, the code defines hash algorithms in the Hashes array. Each entry includes an ID, name, and flags to indicate if it is deprecated or used in system encryption.
- The code includes a function called CipherInit that initializes the specified cipher with the provided key. It supports ciphers like AES, Serpent, Twofish, Camellia,

[3] In VeraCrypt, XTS (XEX-based Tweaked Codebook Mode with Ciphertext Stealing) is an encryption mode used for disk encryption. It is specifically designed for encrypting data stored on disk or other block-oriented storage devices. XTS mode is commonly used in conjunction with block ciphers like AES (Advanced Encryption Standard).

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

GOST89, and Kuznyechik. The function returns success or failure codes.

- Another function called EncipherBlock performs encryption on a single block of data using the specified cipher. It supports ciphers like AES, Twofish, Serpent, and Camellia.
- There is a EncipherBlocks function that performs encryption on multiple blocks of data using the specified cipher. The number of blocks to encrypt is determined by the blockCount parameter.

After examining this VeraCrypt software using IDA Pro software, it analysed that this software contains a large number of modules. The summary of these modules is given below in Table 3. Table 3 displays VeraCrypt modules, including KDF, encryption, decryption, random number generator, hashing, boot loader, and driver, ensuring data integrity[14]

Table 3 VeraCrypt Modules

Sensitive Module	Description
Key derivation function (KDF)	Generates a strong and unique key based on the user's input password
Encryption and decryption modules	Encrypts and decrypts the data on the fly using algorithms like AES, Serpent, and Two fish
Random number generator	Generates random numbers used in the encryption process, using a combination of entropy sources
Hashing modules	Uses various hashing algorithms like SHA-256, SHA-512, and Whirlpool for integrity checking and password verification
Boot loader	Loads the VeraCrypt driver during the boot process to access encrypted data
Driver modules	Kernel-level driver modules intercept read and write requests to encrypted volume, ensuring data integrity

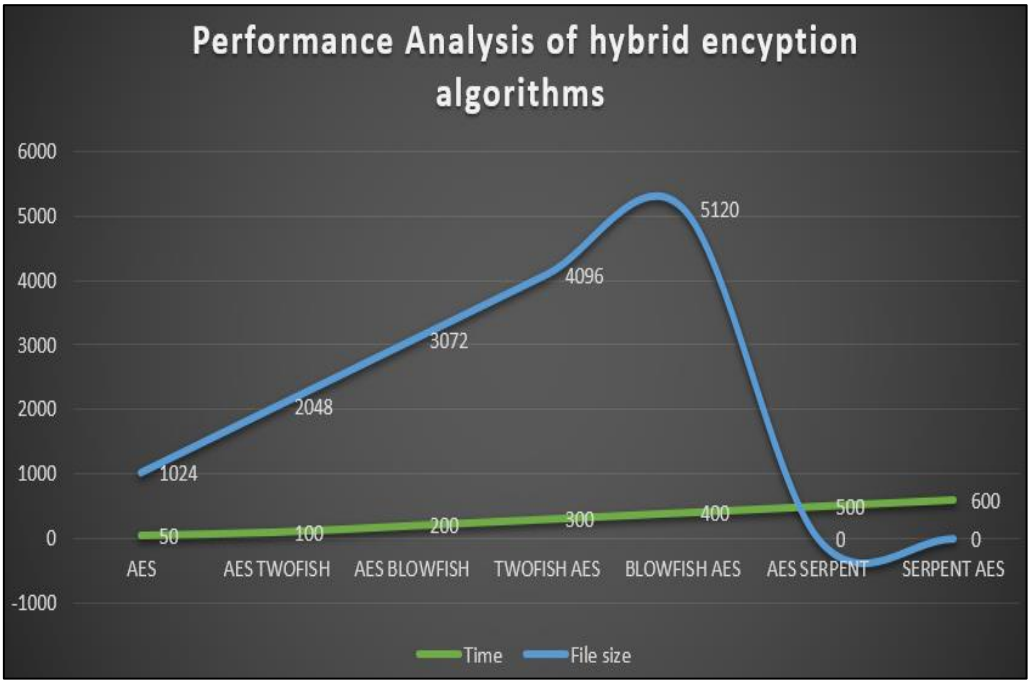


Fig 7 Real Time Taken by Different Hybrid Encryption Algorithms

Figure 7 examines a Hybrid Encryption Algorithm's performance. It looks at key-related tasks, encryption speed, security factors, computing resources, and more. The goal is to find the right balance between performance and security by considering various aspects like key handling, encryption speed, security strength, scalability, and specific needs for different applications [48].

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

➢ *HEADER _Common_ Volumes*

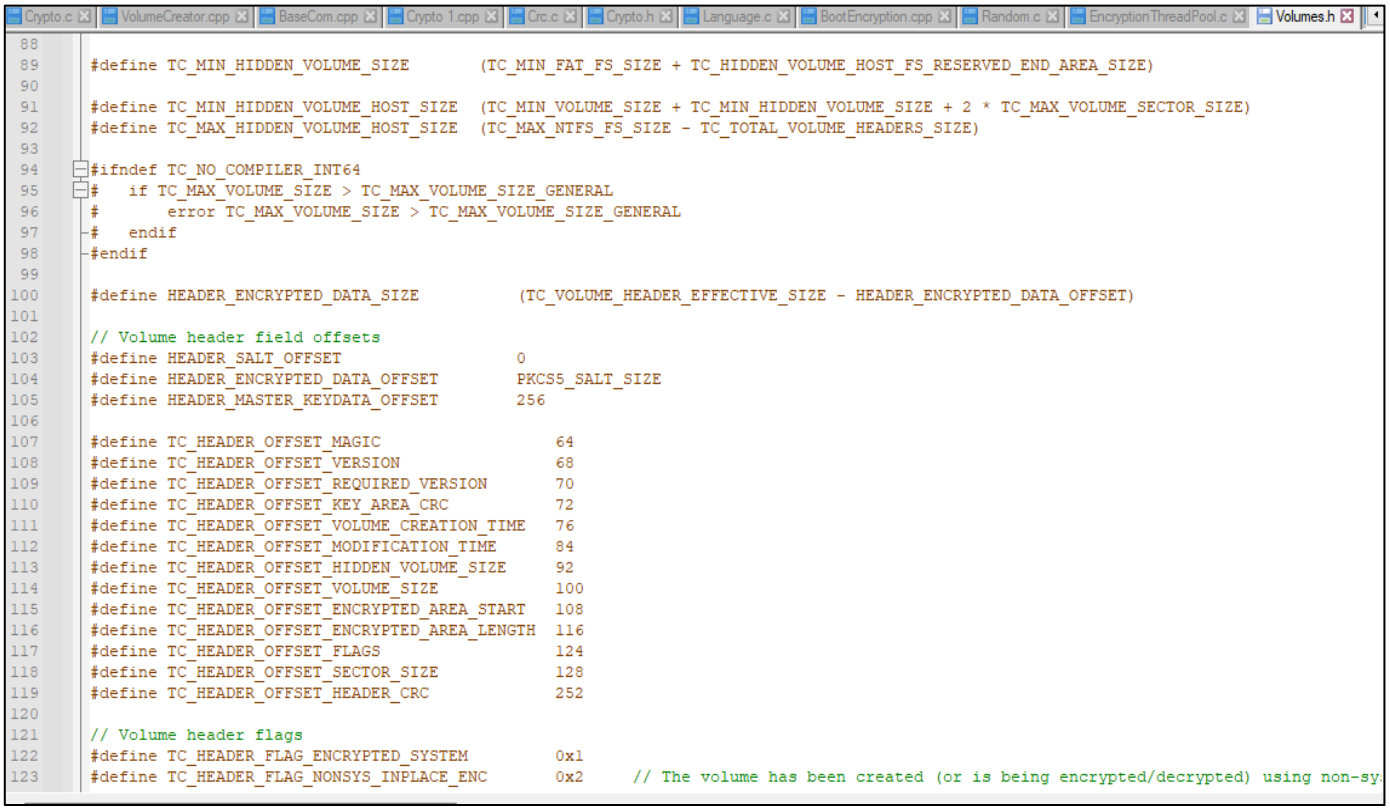


Fig 8 HEADER Common_ Volumes

In Fig. 8, the code includes several header files and defines functions and methods within the VeraCrypt namespace. Here is a breakdown of its components and their meanings:

➢ *Header Files are Included*

- Crc32.h: Contains the declaration and implementation of the CRC32 class for calculating CRC32 checksums.
- EncryptionModeXTS.h: Includes the necessary definitions for the EncryptionModeXTS class, which represents the XTS encryption mode.
- Pkcs5Kdf.h: Provides the definition of the Pkcs5Kdf class, which is responsible for key derivation using the PKCS5 standard.
- VolumeHeader.h: Contains the definition of the VolumeHeader class, which represents the header of a VeraCrypt volume.
- VolumeException.h: Defines the VolumeException class, which represents exceptions related to volume operations.
- Common/Crypto.h: Includes common cryptographic functions and data structures used within the VeraCrypt namespace.
- VolumeHeader constructor: Initializes a VolumeHeader object with a specified size and calls the Init() method to set default values.
- VolumeHeader destructor: Calls the Init() method to reset the VolumeHeader object to its default state.
- Init() method: Resets the VolumeHeader object by setting all member variables to their default values.
- Create() method: Creates a volume header using the provided options and encrypts it with the specified parameters.
- Decrypt() method: Decrypts the encrypted volume header using the provided password, key derivation functions, and encryption algorithms.
- Deserialize () method: Deserializes the decrypted volume header and extracts relevant information.
- Various member variables are used to store information such as the volume key, creation time, data sizes, encryption area details, flags, and sector size.
- The code uses iterations, conditionals, and function calls to derive the header key, decrypt the header, and deserialize its contents.
- The code performs checks and validations, throwing exceptions, when necessary, to ensure the header's integrity and compatibility with the specified parameters.

➢ *Header Structure and Encryption Operations*

Encrypted volumes are secure methods for storing sensitive information, especially in disk encryption. The header contains essential information about the algorithm, key size, and metadata. Encrypted volumes use a user's passphrase or key file for encryption, and the mounting process initiates decryption. Encrypted volumes maintain data privacy and security, especially in portable storage devices or situations with physical access concerns.

Figure 8 shows a special file called the VeraCrypt volume header. This file is like a container that holds

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

important information about a VeraCrypt secure storage area. Inside this file, there are different types of parts, kind of like building blocks, called structures and classes. Some of these parts are about the type of volume, how the header was made, and the main header itself. The type of volume can be one of three things: unknown, normal (regular), or hidden. This helps tell the computer what kind of storage area it's dealing with.

Volume Header Creation Options are setup decisions for creating a volume header, including data encryption, key methods, and storage size. The volume header represents a VeraCrypt storage area's main information, including building, locking, and unlocking. It also contains storage details like size, size, and encryption.

Table 4 Format Specification of Header Part:

Offset	Size	Description
0	64	Salt
64	4	ASCII string "TRUE"
68	2	Volume header format version: 5
70	2	Minimum version required to open volume
72	4	CRC checksum of decrypted bytes 256-511
76	16	Reserved values
92	8	Size of hidden volume, zero for non-hidden
100	8	Size of volume
108	8	Byte offset of start of master key scope
116	8	Size of encrypted area in master key scope
124	4	Flag bits of encryption type
128	4	Sector size in bytes
132	120	Reserved values
252	4	CRC checksum of decrypted bytes 64-251
256	256	Concatenated master keys,

Table 4 details header format specifications, including offset, size, description, ASCII string, volume, minimum version, CRC checksum, reserved values, and concatenated keys.

The provided document describes the process of establishing an encrypted storage space within a file. It highlights the presence of both regular and concealed encrypted storage compartments. When it comes to encrypting a system partition, the starting point is determined as 31744. Copies of header files are positioned at 0 and 65536, each sized at 131072 bytes (which is twice the size of 65536). On the recovery disc, the files have a size of 1835008 bytes and begin at an offset of 84992. A detailed compilation of all the mentioned positions is available in Table 5.

Table 5 Volume Type and Data Extraction Offset:

Type	Offset	Size
Normal encrypted volume	31744	variable
Hidden encrypted volume	31744	variable
Header backup file	0	131072
Header backup file	65536	131072
Rescue disc file	84992	1835008

Table 5 details volume types, data extraction offsets, header backup files, and rescue disc files, highlighting starting offsets and sizes for data extraction.

➢ *A Description of Data Storage*

Hard disc drives are made up of stacked platters, connected by a spindle, read and written by a head attached to a movable arm, and can hold 512 bytes of data. The original 504 MB disc restriction imposed by older machines has been exceeded by newer hard disc drive capacity.

➢ *An Essential Requirement for VeraCrypt's Data Forensics*

Preserving the initial data is extremely important in VeraCrypt data forensics. Investigators must ensure that the original VeraCrypt storage remains unchanged during the investigation. Altering the original storage can harm the data's reliability and make it inadmissible in legal situations. To sum up, two key needs for VeraCrypt data forensics are keeping the original data intact and having the ability to retrieve encryption keys. By protecting the authenticity of the original disk and acquiring encryption keys, forensic investigators can extract pertinent details from VeraCrypt storage without altering the original data.

➢ *Data Hiding with Encryption*

Encryption is a common technique to hide information, with various tools available for this purpose. Our study focused on using a specific tool called VeraCrypt to make encrypted storage spaces. During our tests, we put three text files into both the normal and hidden volumes created by VeraCrypt. To analyze the volumes and the VeraCrypt tool,

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

we transformed a VMDK[4] file into a binary file and studied the hexadecimal values in the two volume files. We discovered that neither of these volume files had sequences of consecutive zeros. While VeraCrypt has a feature to fill empty parts, Windows 10 should use zero bytes to fill those areas. The absence of such consecutive zero bytes hinted at the use of VeraCrypt. Furthermore, Figure 5 shows that the TSK-Autopsy forensic analysis tool identifies volume files as likely encrypted files.

➢ *The Operating Mode and Algorithms of VeraCrypt*

VeraCrypt used AES, Serpent, and Two fish encryption algorithms, which have been verified by cryptographers. Only algorithms reserved in the final version of VeraCrypt.

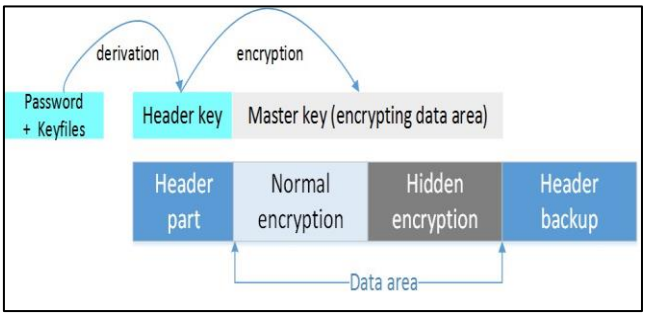


Fig 9 Shows the Container Structure and Encryption Key Relationship.

Figure 9, discover hidden encryption data areas using VeraCrypt, follow setup, use password key files, store metadata, analyse header, use forensic tools, and decrypt using generated header key. Decryption should adhere to ethical standards. VeraCrypt applies two rounds of encryption to data blocks using five concatenations and 128-bit and 256-bit keys [37]. However, it deviates from XEX by employing two independent keys instead of a single key. The XTS mode, acknowledged by NIST and IEEE, is specifically beneficial for safeguarding data on storage devices [38].

➢ *Finding the Size of the Hidden Container Encrypted Volumes*

As of 2009, over 12 million individuals had downloaded the data encryption software VeraCrypt. VeraCrypt offers the capability to encrypt entire volumes or disks and create encrypted containers. Additionally, it provides disguised volume functionality, allowing one password to reveal predetermined harmless information while another password unlocks the actual content [42].

In file systems that utilize the FAT structure, the cluster serves as the smallest unit of storage on a volume. The actual file content is stored elsewhere, while the directory entry only contains information about the file. The file system maintains and tracks "cluster chains" in a File Allocation Table (FAT), which is stored near the beginning of the volume. Two copies of the FAT are maintained for data reading purposes. When a file is written to the disk, the FAT entries are updated to indicate that the allocated clusters are no longer available. The size of each FAT entry varies depending on the file system in use [49],[50].

To prevent potential harm, users can choose to enable the option of "protecting the hidden volume from damage caused by writing to the outer volume" and provide the password for the concealed volume. Hidden volumes in VeraCrypt are stored within the unused space of a normal VeraCrypt disk. If unauthorized individuals manage to access a dismounted Vera Crypt volume at different points in time, they may be able to detect the sectors within the volume that undergo changes. Hargreaves et al. showcased this technique by using multiple volume copies to deduce the whereabouts of the hidden volume. [31], [2].

Figure 10. A standard VeraCrypt volume's structure appears on the left, while a version that has a concealed volume is displayed on the right.

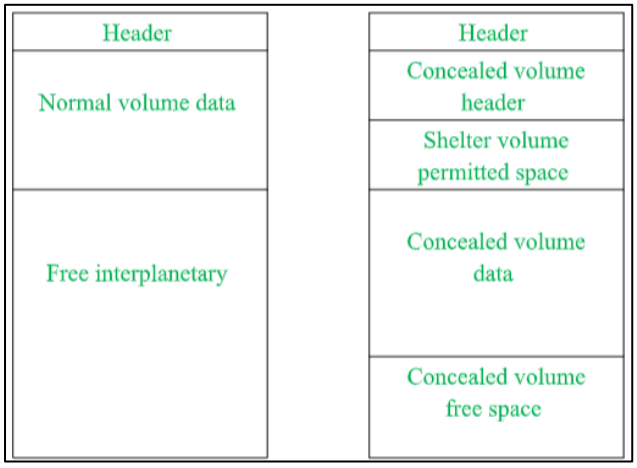


Fig 10 VeraCrypt Encrypted File Container

To decrypt a VeraCrypt volume with a hidden volume, the following steps are taken: The user inputs a password, which is used to generate a key. This key is then used to attempt decryption of the hidden volume header. If decryption succeeds, the volume key is extracted from the header and used to decrypt the hidden volume. If decryption fails, the key is utilized to decrypt the header and subsequently decrypt the visible (cover) volume [9].

V. CONCLUSION

The research conducted in this study focused on analysing VeraCrypt's encryption and data recovery capabilities from a forensic standpoint. It provided insights on how to effectively hide selected key files and password hashes, as well as a helpful method for locating encrypted volumes. The research also explored techniques for extracting data by determining the offset values required for password authentication in protected VeraCrypt containers. Additionally, the study delved into the source code module and interoperability of VeraCrypt, and developed targeted

[4] VMDK file refers to a file format used by virtualization software,

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

password guessing methods for forensic software. The overall goal of the research was to enhance the forensic analysis of VeraCrypt encryption by offering practical solutions and valuable insights into its implementation.

REFERENCES

[1]. D. Lillis, B. Becker, T. O'Sullivan, and M. Scanlon, "Current Challenges and Future Research Areas for Digital Forensic Investigation," DFRWS 2016 USA - Proc. 16th Annu. USA Digit. Forensics Res. Conf. 11th ADFSL Conf. Digit. Forensics, Secur. Law (CDFSL 2016)At Daytona Beach, FL, USA., vol. 5, no. 2, 2016, [Online]. Available: http://arxiv.org/abs/1604.03850

[2]. K. Conlan, I. Baggili, and F. Breitinger, "Anti-forensics: Furthering digital forensic science through a new extended, granular taxonomy," DFRWS 2016 USA - Proc. 16th Annu. USA Digit. Forensics Res. Conf., vol. 18, no. December 2015, pp. S66–S75, 2016, doi: 10.1016/j.diin.2016.04.006.

[3]. S. Raghavan, "Digital forensic research: current state of the art," CSI Trans. ICT, vol. 1, no. 1, pp. 91–114, 2013, doi: 10.1007/s40012-012-0008-7.

[4]. M. A. Ako, "Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data," Cryptogr. Netw. Secur., no. June, 2017, [Online]. Available: https://www.researchgate.net/publication/317615794

[5]. G. G. Richard and V. Roussev, "Next-generation digital forensics," Commun. ACM, vol. 49, no. 2, pp. 76–80, 2006, doi: 10.1145/1113034.1113074.

[6]. D. Denning and W. Baugh, "Encryption and Evolving Technologies: Tools of Organized Crime and Terrorism," Edpacs, vol. 25, no. 10, pp. 17–17, 1998, doi: 10.1201/1079/43232.25.10.19980401/30163.7.

[7]. A. Davies, "Detecting hidden volumes and operating systems," 2014.

[8]. D. Nn. H. W. Jackson, an Advanced Introduction To Alloy. Springer International Publishing, 2018.

[9]. A. Tomlinson, "TrueCrypt_Tests." 2014.

[10]. Werner Koch, "GNU Privacy Guard – Wikipedia." p. Retrieved 27.April 2023. [Online]. Available: https://de.wikipedia.org/wiki/GNU_Privacy_Guard

[11]. M. Bursać, R. Vulović, and M. Milosavljević, "Comparative Analysis of the Open Source Tools Intended for Data Encryption," Int. Conf. Inf. Technol. Dev. Educ. Zrenjanin, Repub. Serbia, no. June, 2017.

[12]. K. Saxena, D. Rajdev, D. Bhatia, and M. Bahl, "ProtonMail: Advance Encryption and Security," Proc. - Int. Conf. Commun. Inf. Comput. Technol. ICCICT 2021, 2021, doi: 10.1109/ICCICT50803.2021.9510041.

[13]. C. Tan, L. Zhang, and L. Bao, "A Deep Exploration of BitLocker Encryption and Security Analysis," Int. Conf. Commun. Technol. Proceedings, ICCT, vol. 2020-Octob, pp. 1070–1074, 2020, doi: 10.1109/ICCT50939.2020.9295908.

[14]. J. Lee, "Security Evaluation of Romulus," Fraunhofer Inst. Secur. Inf. Technol. Andreas Poller Rheinstrasse 75, D-64295 Darmstadt, Ger. E-Mail, 2018.

[15]. E. Casey, "Practical approaches to recovering encrypted digital evidence," Proc. Digit. Forensic Res. Conf. DFRWS 2002 USA, vol. 1, no. 3, 2002.

[16]. Henry B Wolfe, "Encountering Encrypted Evidence (potential)," Proc. 2002 InSITE Conf., no. June, 2002, doi: 10.28945/2590.

[17]. H. Wolfe, "Penetrating encrypted evidence," Digit. Investig., vol. 1, no. 2, pp. 102–105, 2004, doi: 10.1016/j.diin.2004.04.002.

[18]. A. Van Deursen and E. Burd, "Software reverse engineering," J. Syst. Softw., vol. 77, no. 3, pp. 209–211, 2005, doi: 10.1016/j.jss.2004.03.031.

[19]. Y. Huang, T. Y. Zhuo, Q. Xu, H. Hu, X. Yuan, and C. Chen, "Training-free Lexical Backdoor Attacks on Language Models," 2023, doi: 10.1145/3543507.3583348.

[20]. V. Cambareri, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "On Known-Plaintext Attacks to a Compressed Sensing-Based Encryption: A Quantitative Analysis," IEEE Trans. Inf. Forensics Secur., vol. 10, no. 10, pp. 2182–2195, 2015, doi: 10.1109/TIFS.2015.2450676.

[21]. N. Zaidenberg and A. Resh, "Timing and side channel attacks," Intell. Syst. Control Autom. Sci. Eng., vol. 78, no. May, pp. 183–194, 2015, doi: 10.1007/978-3-319-18302-2_11.

[22]. H. Bojinov, D. Sanchez, P. Reber, D. Boneh, and P. Lincoln, "Neuroscience meets cryptography," Commun. ACM, vol. 57, no. 5, pp. 110–118, 2014, doi: 10.1145/2594445.

[23]. S. Yadav, K. Ahmad, and J. Shekhar, "Analysis of digital forensic tools and investigation process," Commun. Comput. Inf. Sci., vol. 169 CCIS, pp. 435–441, 2011, doi: 10.1007/978-3-642-22577-2_59.

[24]. Q. X. Miao, "Research and analysis on Encryption Principle of TrueCrypt software system," 2nd Int. Conf. Inf. Sci. Eng. ICISE2010 - Proc., pp. 1409–1412, 2010, doi: 10.1109/ICISE.2010.5691392.

[25]. E. Casey, G. Fellows, M. Geiger, and G. Stellatos, "The growing impact of full disk encryption on digital forensics," Digit. Investig., vol. 8, no. 2, pp. 129–134, 2011, doi: 10.1016/j.diin.2011.09.005.

[26]. S. E, R. C. Davis, and B. A. Jackson, "Digital Evidence and the US Criminal justice system," Crimanal justices J., vol. 5, 2019.

[27]. S. Rekhis and N. Boudriga, "A system for formal digital forensic investigation aware of anti-forensic attacks," in IEEE Transactions on Information Forensics and Security, 2012, vol. 7, no. 2, pp. 635–650. doi: 10.1109/TIFS.2011.2176117.

[28]. A. Balducci, S. Devlin, and T. Ritter, "Open Crypto Audit Project TrueCrypt Cryptographic Review Cryptography Services Final Report," Cryptogr. Netw. Secur., vol. 5, 2015.

[29]. I. P. R. L. Tvrdík, "Analysis of the Rescue File of BestCrypt Volume Encryption," 2017.

[30]. C. Meijer and B. Van Gastel, "Advisory on Solid State Disks (SSDs) / Digital Security , Radboud University / 5-11-2018 Research results," pp. 1–2, 2018.

[31]. Z. H. Abdullahi and S. K. Singh, "A Conceptual and Technical Perspective of Reverse Engineering In

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug189

Digital forensic," in 2nd International Conference on Recent Development in Engineering ,Sciences, and Management Goverment Engineering College Bharatpur, Rjasthan, 2023, no. April. [Online]. Available: ISBN; 978-9391535-43-8

[32]. H. Payal and R. Tomer, "Review on reverse engineering," J. Crit. Rev., vol. 7, no. 7, pp. 1408–1412, 2020, doi: 10.31838/jcr.07.07.255.

[33]. A. Czeskis, D. J. St Hilaire, K. Koscher, S. D. Gribble, T. Kohno, and B. Schneier, "Defeating encrypted and deniable file systems: TrueCrypt v5.1a and the case of the tattling OS and applications," in HotSec 2008 - 3rd USENIX Workshop on Hot Topics in Security, 2008, pp. 1–7.

[34]. H. A. Miiller, J. H. Jahnke, D. B. Smith, M. A. Storey, S. R. Tilley, and K. Wong, "Reverse engineering: A roadmap," Proc. Conf. Futur. Softw. Eng. ICSE 2000, pp. 47–60, 2000, doi: 10.1145/336512.336526.

[35]. J. Leng and T. Li, "Research on Computer System Information Hiding Anti-Forensic Technology," vol. 83, no. Snce, pp. 55–60, 2018.

[36]. A. Balducci, S. Devlin, and T. Ritter, "Open Crypto Audit Project - TrueCrypt Cryptographic Review," iSECpartners, no. Security Assesment, 2015.

[37]. A. Tomlinson and R. Holloway, "Detecting the use of TrueCrypt Forensic investigations : Detecting evidence of the use of TrueCrypt," R. Hollow. Inf. Secur. Thesis Ser. | Detect. use TrueCrypt Forensic, 2018.

[38]. L. Zhang, Y. Zhou, and J. Fan, "The forensic analysis of encrypted Truecrypt volumes," PIC 2014 - Proc. 2014 IEEE Int. Conf. Prog. Informatics Comput., pp. 405–409, 2014, doi: 10.1109/PIC.2014.6972366.

[39]. C. T. Lijun Zhang, Xiaoyan Deng, "An Extensive Analysis of truecrypt Encryption Forensics," pp. 1–6, 2019, doi: 10.1145/3331453.3361328.

[40]. H. Agarwal, F. Husain, and P. Saini, Advances in Computing and Data Sciences, vol. 1046, no. July. Springer Singapore, 2019. doi: 10.1007/978-981-13-9942-8.

[41]. R. Wartell, Y. Zhou, K. W. Hamlen, M. Kantarcioglu, and B. Thuraisingham, "Differentiating code from data in x86 binaries," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6913 LNAI, no. PART 3, pp. 522–536, 2011, doi: 10.1007/978-3-642-23808-6_34.

[42]. E. Stroulia and T. Systä, "Dynamic analysis for reverse engineering and program understanding," ACM SIGAPP Appl. Comput. Rev., vol. 10, no. 1, pp. 8–17, 2002, doi: 10.1145/568235.568237.

[43]. N. Chen, B. Chen, and W. Shi, "MobiWear: A Plausibly Deniable Encryption System for Wearable Mobile Devices," Springer, pp. 138–154, 2021, doi: 10.1007/978-3-030-80851-8_10.

[44]. E. C. Hosgor, "Detection and Mitigation of," Computer (Long. Beach. Calif)., no. December 2020, pp. 1–8, 2020, doi: 10.5281/zenodo.4425257.

[45]. S. G. Lewis and T. Palumbo, "BitLocker Full-Disk Encryption," 2018. doi: 10.1145/3235715.3241363.

[46]. R. Stoykova, R. Nordvik, M. Ahmed, K. Franke, S. Axelsson, and F. Toolan, "Legal and technical questions of file system reverse engineering," Comput. Law Secur. Rev., vol. 46, 2022, doi: 10.1016/j.clsr.2022.105725.

[47]. E. Casey, G. Fellows, M. Geiger, and G. Stellatos, "The growing impact of full disk encryption on digital forensics," Digit. Investig., vol. 8, no. 2, pp. 129–134, 2011, doi: 10.1016/j.diin.2011.09.005.

[48]. Et. al., Pravin Soni, "Performance Analysis of Cascaded Hybrid Symmetric Encryption Models," Turkish J. Comput. Math. Educ., vol. 12, no. 2, pp. 1699–1708, 2021, doi: 10.17762/turcomat.v12i2.1506.

[49]. Benjamin Wah, "Encyclopedia of Computer Science and Engineering," Wiley Encycl. Comput. Sci. Eng., vol. 1–170, 2008.

[50]. S. Arif and M. M. Kumar, "Cyber Safety Analysis Using Reverse Engineering Syed," Int. J. Res. Publ. Rev. J. homepage www.ijrpr.com ISSN 2582-7421 Cyber, vol. 4, no. 1, pp. 399–403, 2023.