International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Software Reverse Engineering Techniques for Evaluating Anti-Forensic Encryption Tools: A Framework Development and Analysis

Zakariyya Hassan Abdullahi[1], Shailendra Kumar Singh[2*], Moin Hasan[3]

Submitted: 24/06/2023 **Revised**: 06/08/2023 **Accepted**: 27/08/2023

Abstract: The digitalization of the world has led to increased cybercrimes, resulting in the growth of digital forensic tools. To counter, anti-forensic tools have also been developed to hinder their effectiveness and digital evidence recovery. Software reverse engineering is assumed to be a potential technique to be able to assess anti-forensic tools. Software reverse engineering is the process of analyzing a computer program or software application in order to understand its functionality, design, and behavior. This article explores software reverse engineering techniques to evaluate anti-forensic encryption tools, specifically focusing on the VeraCrypt. VeraCrypt is free, open-source disk encryption software that provides strong on-the-fly encryption for various storage devices. Moreover, a novel framework is proposed for assessing such tools, emphasizing practical applicability, data driven analysis, and comparative assessment. The proposed framework works in multiple phases to assess/examine the VeraCrypt and is able to take different decisions conditionally. Extensive experiments are performed to evaluate the proposed framework and the obtained results reinforce the proposal.

Keywords: *Digital forensic; anti-forensic; encryption; software reverse engineering; VeraCrypt*

1. Introduction

For the last few decades, the word has seen an exponential growth towards the digitization. Along with that, the rate of cybercrimes has also increased, resulting in the emergence of numerous digital forensic tools/software. As a counter attack, several anti-forensic tools have also been developed to hinder their effectiveness and digital evidence recovery [1]– [4]. Therefore, it is highly important to understand the structure and behavior of such tools [5].

Rather than conventional digital forensic, software reverse engineering is assumed to be a potential technique for the assessment/evaluation/examination of anti-forensic tools [3], [6], [7]. Software reverse engineering is the process of analyzing a software program, binary executable, or codebase to understand its functionality, behavior, and underlying design, often with the intent of extracting valuable information, modifying its behavior, or recreating its source code [8]. This practice involves deconstructing the compiled or executed software into its constituent parts, such as assembly code, data structures, algorithms, and high-level logic, in order to gain insights into how the software operates [9]. Software reverse engineering can be applied to various purposes, including:

[1,2]*School of Computer Science and Engineering, Lovely Professional university, phagwara, India*
[3]*Department of computer Science and Engineering, Jain Deemed-to-be-University, Bengaluru, India*
hassanzakariyya78@gmail.com[1], drsksingh.cse@gmail.com[2], mmoinhhasan@gmail.com[3]
**Correspondence Author: drsksingh.cse@gmail.com*

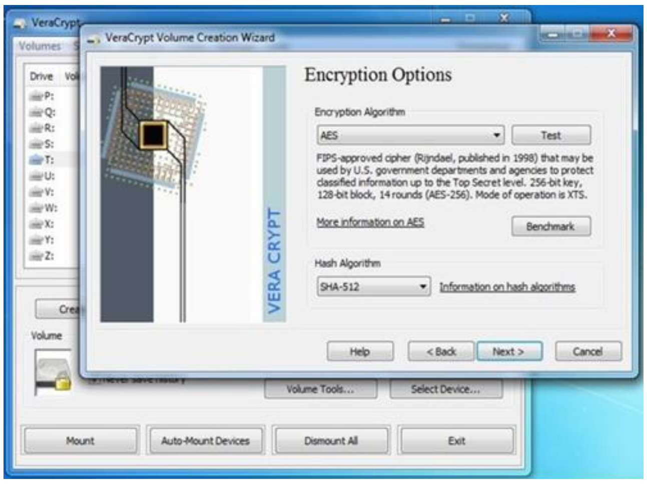
- Understanding: Reverse engineering helps researchers, analysts, and developers comprehend the inner workings of a software application when the original source code is unavailable or insufficiently documented.
- Security Analysis: It is used to identify vulnerabilities, security weaknesses, and potential exploits in software systems. By dissecting the code, security professionals can uncover flaws that could be exploited by malicious actors.
- Interoperability: In cases where documentation is missing or incomplete, reverse engineering can aid in creating compatibility between software systems, allowing different programs to communicate effectively.
- Legacy System Maintenance: Reverse engineering can be used to maintain and update outdated or unsupported software systems by understanding their functionality and making necessary modifications.
- Competitive Analysis: Businesses might engage in reverse engineering to study competitors' software products and gain insights into their features, design, and strategies.
- Copyright and Patent Infringement: In legal contexts, reverse engineering can help determine whether a software product infringes upon copyrights or patents, as it can reveal similarities or code reuse.

The process of software reverse engineering typically involves tasks such as disassembling binary code, analyzing memory structures, identifying function calls and control flow, reconstructing data structures, and ultimately creating a higher-level representation of the software's logic and behavior [10], [11]. It often requires knowledge of low-level programming languages, assembly languages, debugging tools, and familiarity with software architecture and design principles [12].

To this advantage, in this article, we explore software reverse engineering techniques to evaluate antiforensic encryption tools, specifically focusing on the VeraCrypt. VeraCrypt is free, open-source disk encryption software that provides strong on-the-fly encryption for various storage devices, such as hard drives, USB drives, and virtual disk images [13], [14]. It is a successor to the TrueCrypt project, which was a popular disk encryption tool until its development was discontinued in 2014 [15]–[17]. VeraCrypt offers several features and enhancements over TrueCrypt, including:

- Improved Security: VeraCrypt uses a variety of encryption algorithms, including AES, Serpent, and Twofish, to encrypt data on-the-fly [18]–[20]. It also supports various encryption modes, key derivation functions, and hash algorithms [21], allowing users to customize their security settings.
- Hidden Volumes: One of the notable features of VeraCrypt is its ability to create hidden encrypted volumes within a larger encrypted volume. This allows users to have a plausible deniability option, where they can reveal a password that unlocks only the outer encrypted volume, while keeping the existence of the hidden volume confidential.
- Cross-Platform Support: VeraCrypt is available for multiple operating systems, including Windows, macOS, and Linux. This cross-platform compatibility makes it a versatile encryption solution.
- System Partition Encryption: VeraCrypt can encrypt the entire system partition, including the operating system files, boot loader, and user data. This provides a high level of security for protecting sensitive information on a computer.
- Portable Mode: VeraCrypt offers a portable mode, allowing users to run the application directly from a removable storage device without the need for installation on the host system.
- Volume Formats: VeraCrypt supports various volume formats, including standard encrypted volumes, hidden volumes, and even entire operating system partitions.
- Plausible Deniability: As mentioned earlier, VeraCrypt's hidden volume feature provides an element of plausible deniability, where an observer cannot determine if a hidden volume exists within the outer volume [22].

Due to its strong encryption capabilities and the ability to create hidden volumes, VeraCrypt is often used by individuals, businesses, and organizations that require secure data storage and protection against unauthorized access. A snapshot of VeraCrypt software is shown in Fig. 1.


Encryption Options

Encryption Algorithm: AES

FIPS-approved cipher (Rijndael, published in 1998) that may be used by U.S. government departments and agencies to protect classified information up to the Top Secret level. 256-bit key, 128-bit block, 14 rounds (AES-256). Mode of operation is XTS.

Hash Algorithm: SHA-512

Fig.1 Snapshot of VeraCrypt

In this research, a novel framework is proposed for assessing such tools, emphasizing practical applicability, data-driven analysis, and comparative assessment. The proposed framework works in multiple phases; viz., *system structuring*, *system characterization*, *static analysis*, *dynamic analysis*, and *document generation*; to assess/examine the VeraCrypt and is able to take different decisions conditionally. To evaluate the performance, extensive experiments are performed and the obtained results reinforce the proposal.

The article is organized as follows: Section 2 discusses the related works in detail. Section 3 explains the proposed framework followed by the experimental evaluation in Section 4. Section 5 finally concludes the article along with some future research scope.

2. Related Works

The field of reverse engineering in the context of digital forensics has garnered significant attention from researchers, leading to a wide array of studies exploring its potential and effectiveness in various domains. In this section, we present a comprehensive overview of related works, highlighting the significance of reverse engineering in digital investigations, legacy program comprehension, anti-forensics techniques, legal challenges, data security, and encryption.

Software forensics and reverse engineering have been a subject of extensive research in recent years. Ekanem and Meye [23] emphasized the importance of using reverse engineering in software forensics to uncover changes made by infringers, enabling prosecuting teams to present strong evidence in court. Their study underscored the necessity of automating reverse engineering processes to improve efficiency and accuracy. Singh [24] conducted a comprehensive review of reverse engineering theories and tools, highlighting their potential in addressing challenges arising from legacy program code without adequate documentation. The review encompassed various methodologies and tools used in reverse engineering, contributing to the field's overall understanding and application. Müller and Kienle [25] delved into the realm of software reverse engineering, offering insights into analyzing legacy systems to synthesize valuable information about the target system. They discussed the process of reverse engineering, which ranges from code-centric approaches to domain knowledge-centric approaches. The study served as a valuable resource for understanding the diverse aspects of software reverse engineering and its relevance in program comprehension.

Addressing anti-forensics techniques is another critical area of research in digital forensics. Gobel and Baier [26] highlighted data hiding within the filesystem layer as a significant anti-forensic technique employed by cyber-criminals. They explored the implications of data hiding and its impact on digital forensic analysis, emphasizing the need for effective countermeasures to detect hidden data and thwart anti-forensic efforts. Conlan et al. [10] presented an extended, granular taxonomy for anti-forensics, categorizing various techniques used by cyber-criminals to hinder investigations. The taxonomy provided a structured framework for understanding the different anti-forensics methodologies and their potential impact on digital forensic analysis. Bhat et al. [12] examined computer forensic tools' ability to detect file system anti-forensic attacks. Their study employed a six-stage testing methodology to evaluate the effectiveness of these tools in real-world scenarios. The findings revealed that most anti-forensic attacks went unnoticed, highlighting the need for aggressive research to address the pitfalls and improve the efficacy of computer forensic tools in countering anti-forensic challenges. Their work shed light on the significance of continuously evolving computer forensic tools to keep up with the evolving tactics used by cyber-criminals.

Legal challenges and regulation in digital forensics have also been a focal point of research. Stoykova et al. [27] discussed the legal and technical questions surrounding file system reverse engineering for law enforcement. Their study addressed concerns related to the legality of reverse engineering file systems, intellectual property protection, and confidentiality obligations. They emphasized the importance of striking a balance between the need for digital evidence in legal proceedings and respecting individuals' rights to privacy and data protection. The research offered valuable insights into the challenges faced by law enforcement agencies in utilizing reverse engineering techniques while adhering to legal and ethical standards. Authors in [28] explored gaps in digital forensic methodology and raised concerns about trade secrets and vulnerability disclosure. Their study highlighted the need for further development in regulatory frameworks to address these challenges and improve the effectiveness of digital forensic investigations. The paper underscored the importance of standardizing digital forensic practices to ensure consistency and reliability in the investigation process.

Data security and encryption have been areas of interest in the context of digital forensics [29]. Al-Dhaqm et al. [30] examined different areas of digital forensics and identified a lack of standard practices. They proposed a metamodeling approach as a potential solution to address this issue and suggested testing its effectiveness in real-world scenarios. Their work laid the foundation for future research to establish standardized practices in digital forensics, promoting uniformity and credibility in digital forensic investigations. Singh and Singh [31] introduced a novel encryption technique using floating-point numbers to enhance data security. The novel approach involved generating different numbers for repeated words and increasing key length to augment encryption strength. The authors in [32] examines machine learning's impact on software engineering, offering practical insights and resources for researchers applying it to source code analysis tasks. The research offered a promising encryption technique that could find application in real-world scenarios to safeguard digital data. Harahap et al. [33] presented a crypto-system hybrid method using symmetrical algorithms and a neural network for secure file encryption. The method demonstrated high accuracy in generating public keys, and its potential application with AES, Blowfish, and hybrid algorithms could strengthen file encryption and data security. Their work opened avenues for exploring advanced encryption techniques to counter cybersecurity threats and protect digital data from unauthorized access. Ahmad et al. [34] develop optimized image encryption using particle swarm optimization and chaotic map for secure image communication, requiring further research.

Various researchers have explored the applications of reverse engineering in digital forensics beyond software forensics. Muttoo and Sushil [35] discussed steganography as a technique for hiding secret messages in digital audio, images, and video files. Their research addressed the challenges posed by steganography in digital forensics investigations, including hacking, duplication, and malicious usage. The study offered insights into capacity and trade-offs between hiding significant information and ensuring that cover alteration remains undetectable, contributing to the advancement of steganalysis techniques in digital investigations. Muller et al. [36] examined how the Rigi project helps in reverse engineering by creating mental models from abstractions that match the maintainers' understanding, thereby enhancing software comprehension for maintenance, re-engineering, and evolution purposes. D. Binkley [37] provided an in-depth evaluation of source code analysis investigations and presented a strategy for progress in the field. The research examined the current state of source code analysis research, identified potential advancements, applications, techniques, and difficulties in software analysis in the upcoming years. Their work provided valuable knowledge for researchers, practitioners, and developers engaged in software analysis, contributing to the continuous evolution and improvement of source code analysis methodologies. Tonella et al. [38] highlighted the evolution of reverse engineering from legacy systems to software-related problems. Their study advocated for empirical evaluation of reverse engineering methods to understand their effects and demonstrate positive cost-benefit trade-offs. The research provided a roadmap for future investigations, including defining a reference taxonomy, clarifying investigation scope, and adopting a common experiment execution framework. Their work contributed to the establishment of standardized practices in reverse engineering, promoting efficient and effective use of the technique in digital forensic investigations. Chikofsky et al. [39] presents a comprehensive taxonomy of reverse engineering and recovery techniques, categorizing methods used to understand and recover information from software systems. It offers insights into code analysis, program comprehension, and design and architectural information recovery, contributing to the advancement of reverse engineering practices.

The extensive range of studies presented in this section underscores the significant impact of reverse engineering within the realm of digital forensics. Researchers have delved into various dimensions of this field, revealing its potential and effectiveness in addressing critical challenges. From software forensics to ant forensics techniques, legal considerations to data security and encryption, the research contributions have collectively illuminated the multifaceted nature of reverse engineering's role in digital investigations. As evidenced by the diverse methodologies, tools, and strategies explored by scholars, it is evident that reverse engineering stands as a crucial pillar in the arsenal of digital forensic practitioners. These studies collectively lay the groundwork for future developments, emphasizing the importance of ongoing research and innovation to navigate the evolving landscape of cybersecurity threats and ensure the integrity, privacy, and security of digital data. Table 1 gives a comparative analysis of the related works discussed above.

Table 1: Comparative analysis of the related works

Authors/References	**Year**	**Contribution**	**Key features**	**Limitation**
Chikofsky et al. [39]	1990	Presents a comprehensive taxonomy of reverse engineering and recovery techniques	Provides a structured framework for understanding reverse engineering techniques	The taxonomy is outdated, and some of the techniques are no longer used
D. Binkley, [37]	2007	Provided an in-depth evaluation of source code analysis investigations and presented a strategy for progress in the field	Provides a roadmap for future research in source code analysis	The study focused on source code analysis, and the findings may not be generalizable to other types of reverse engineering
Tonella et al. [38]	2007	Highlighted the evolution of reverse engineering from legacy systems to	Provides an overview of the history of reverse engineering	The study focused on the evolution of reverse engineering, and the findings may not be

		software related problems		relevant to the current state of the field
Muttoo and Sushil [35]	2008	Discussed steganography as a technique for hiding secret messages in digital audio, images, and video files	Provides an overview of steganography and its applications	The study focused on steganography in digital forensics, and the findings may not be generalizable to other applications of steganography
H. A. Müller [25]	2009	Investigate how the Rigi project helps in reverse engineering by creating mental models from abstractions that match the maintainers' understanding	Offers insights into how reverse engineering can be used to improve software comprehension	The study focused on the Rigi project, and the findings may not be generalizable to other reverse engineering tools and techniques
Singh and Singh [31]	2010	Introduced a novel encryption technique using floating-point numbers to enhance data security	Offers a promising encryption technique that could be used to protect digital data	The technique has not been evaluated in real-world scenarios, and its effectiveness may be limited
Ramandeep Singh [24]	2013	Reviewed reverse engineering theories and tools	Provides a comprehensive overview of reverse engineering techniques	Some of the tools and techniques are outdated
Shao and Balogun [28]	2013	Explored gaps in digital forensic methodology and raised concerns about trade secrets and vulnerability disclosure	Provides insights into the challenges of digital forensic investigations	The study focused on gaps in digital forensic methodology, and the findings may not be generalizable to other challenges faced by digital forensic investigators
Conlan et al. [10]	2016	Presented an extended, granular taxonomy for anti-forensics	Provides a structured framework for understanding antiforensics techniques	Can be complex and difficult to use
Global and Baier [26]	2018	Highlighted data hiding within the filesystem layer as a significant anti- forensics' technique	Can be used to detect hidden data	Requires specialized knowledge and tools
Ahmad et al. [34]	2018	Develop optimized image encryption using particle swarm optimization and chaotic map for secure image communication	Offers a promising image encryption technique that could be used to protect digital images	The technique has not been evaluated in real-world scenarios, and its effectiveness may be limited
M Dayalan [36]	2019	Discussed the process of reverse engineering,	Offers a holistic view of reverse engineering	Can be complex and timeconsuming

		which ranges from code-centric approaches to domain knowledge-centric approaches		
Harahap et al. [33]	2019	Presented a crypto-system hybrid method using symmetrical algorithms and a neural network for secure file encryption	Offers a secure file encryption method that could be used to protect digital data	The method has not been evaluated in real-world scenarios, and its effectiveness may be limited
Bhat et al. [12]	2020	Examined computer forensic tools' ability to detect file system anti-forensic attacks	Provides insights into the effectiveness of computer forensic tools	The study was conducted in a lab environment, and the results may not be generalizable to real-world scenarios
Ekane and Meye [23]	2021	Used reverse engineering to uncover changes made by infringers in software	Can be used to identify copyright infringement	Requires manual analysis, which can be time-consuming and error-prone
Al-Dhaqm et al. [30]	2021	Proposed a metamodeling approach as a potential solution to address the lack of standard practices in digital forensics	Provides a roadmap for future research in digital forensics	The study was conceptual in nature, and the effectiveness of the metamodeling approach has not been empirically evaluated
Stoykova et al [27]	2022	Discussed the legal and technical questions surrounding file system reverse engineering for law enforcement	Provides an overview of the legal and ethical issues surrounding reverse engineering	The study focused on file system reverse engineering for law enforcement, and the findings may not be generalizable to other types of reverse engineering

3. Proposed Framework

The methodology employed in this research aims to provide a comprehensive framework (Fig. 2) for reverse engineering to a target software system, specifically focusing on VeraCrypt. Reverse engineering involves the systematic analysis of the software's design, functionality, and inner workings to gain insights into its components, behavior, vulnerabilities, and interactions. The detailed steps of the reverse engineering process employed in the framework are given as follows:

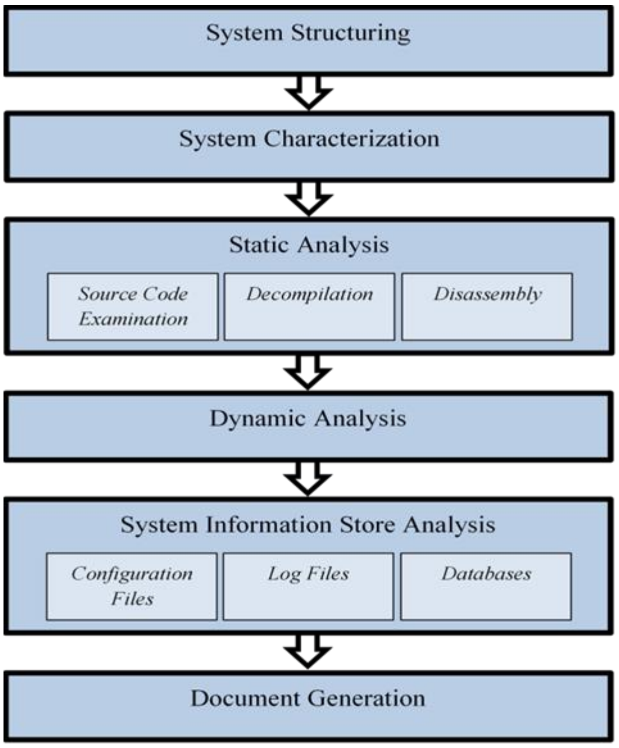


Fig. 2. Proposed Software Reverse Engineering Framework

- **System Structuring:** The first step, System Structuring, involves a meticulous process of preparing the target system, VeraCrypt, for in-depth analysis. This is achieved by utilizing IDA Pro, a powerful disassembler and debugger tool. VeraCrypt software is uploaded into IDA Pro to create a controlled environment where every aspect of the software can be observed and analyzed. Placing the software in such an environment enables to dissect its architecture, components, and execution flow systematically. This structured approach provides a foundation for subsequent analyses and facilitates a comprehensive understanding of the software's inner workings.

- **System characterization:** The next step is to characterize the system by identifying its components, such as modules, functions, and data structures. It also includes identifying the system's entry points and exit points, as well as its communication channels. This information is used to create a high-level overview of the system's architecture and behavior.

- **Static analysis:** In this step, a static analysis of the system to identify potential vulnerabilities is performed. This analysis involves examining the system's code without executing it. The purpose is to

look for common security vulnerabilities, such as buffer overflows, format strings, and SQL injection. Suspicious code patterns, such as code obfuscation and anti-debugging techniques are also looked for. Static analysis is a fundamental technique in reverse engineering. It involves an exhaustive examination of the software's binaries without executing them. This technique unveils critical insights into the software's code structure, logic, and potential vulnerabilities. Static analysis encompasses various sub-steps, including source code examination, decompilation, and disassembly.

Source Code Examination **-** If the source code for the software is available, the framework examines it to gain a better understanding of the software's design and functionality. This information can be used to guide the static analysis of the binaries.

Decompilation **-** If the source code is not available, the framework tries to decompile the binaries to create a human-readable version of the code. This can be helpful for identifying potential vulnerabilities and understanding the software's inner workings.

Disassembly **-** If both the source code and decompiled code are not available, the framework disassembles the binaries to create a low-level representation of the code. This can be a challenging task, but it is often necessary to identify potential vulnerabilities.

- **Dynamic analysis:** Once the static analysis is complete, the framework performs a dynamic analysis of the system. This involves executing the system and observing its behavior. For this purpose, a debugger to step through the code line by line and inspect the values of variables and registers is used. It also uses a memory analysis tool to track the flow of data in memory. This information is used to identify potential vulnerabilities that were not detected during the static analysis.

- **System Information Store Analysis:** The fifth step, System Information Store Analysis, involves exploring and analyzing a repository or database within the VeraCrypt software. This repository contains crucial information about the software's configuration, settings, and interactions with the underlying system. Reverse engineers delve into this repository to decipher how the software interacts with hardware components, system resources, and external dependencies. This analysis aids in comprehending the software's runtime environment, system-level interactions, and potential points of vulnerability. Following entities are analyzed during this step:

***Configuration files* -** Framework examines the software's configuration files to learn about its default settings and how it can be configured. This information can be used to identify potential vulnerabilities, such as default passwords or weak encryption keys.

***Log files* -** Framework examines the software's log files to learn about its activities and interactions with the user. This information can be used to identify potential vulnerabilities, such as errors or unexpected behavior.

***Databases* -** Framework examines the software's databases to learn about its data storage and retrieval mechanisms. This information can be used to identify potential vulnerabilities, such as unauthorized access to sensitive data.

▪ **Document Generation:** Document Generation, the final step, plays a pivotal role in capturing and organizing the insights obtained during the reverse engineering process. The framework creates a comprehensive documentation that outlines the findings, analyses, and observations from each preceding step. This documentation serves as a reference for traceability, allowing future researchers or developers to understand the rationale behind decisions, vulnerabilities detected, and enhancements proposed. The documentation also aids in knowledge transfer, enabling efficient collaboration and informed decision-making throughout the reverse engineering process.

The above steps are algorithmically explained in Algorithm 1 as follows. The methodology presented in this research offers a structured and thorough approach to reverse engineering a target software system, with a specific focus on VeraCrypt. The outlined framework incorporates a sequence of meticulously designed steps that progressively delve into the intricacies of the software's architecture and functionality. The methodology also serves as a valuable guide for reverse engineering practitioners seeking to comprehensively dissect and understand complex software systems like VeraCrypt. By blending a structured approach with a range of analytical techniques, the framework equips researchers, developers, and security professionals with a robust toolkit to uncover vulnerabilities, enhance comprehension, and promote the overall security and reliability of software systems.

3.1. Complexity Analysis

The first step, System Structuring, is essentially a data gathering step. The complexity of this step can be expressed as O(n), where n is the size of the VeraCrypt file. The second step, System Characterization, is also a

data gathering step. The complexity of this step can also be expressed as O(n), where n is the size of the VeraCrypt file. The third step, Static Analysis, is a more computationally intensive step. The complexity of this step depends on the size and complexity of the VeraCrypt file, as well as the tools that are used for static analysis. For a small and simple VeraCrypt file, the complexity of this step may be O(n). However, for a large and complex VeraCrypt file, the complexity of this step may be $O(n^2)$. The fourth step, Dynamic Analysis, is also a computationally intensive step. The complexity of this step depends on the size and complexity of the VeraCrypt file, as well as the tools that are used for dynamic analysis. For a small and simple VeraCrypt file, the complexity of this step may be O(n). However, for a large and complex VeraCrypt file, the complexity of this step may be $O(n^2)$. The fifth step, System Information Store Analysis, is essentially a data gathering step. The complexity of this step can be expressed as O(n), where n is the size of the VeraCrypt file. The sixth step, Document Generation, is a relatively straightforward step. The complexity of this step can be expressed as O(n), where n is the amount of data that needs to be documented.

Hence, the overall complexity of the above algorithm can be expressed as $O(n^2)$, where n is the size of the VeraCrypt file. This is because the most computationally intensive steps, Static Analysis and Dynamic Analysis, both have a complexity of $O(n^2)$. **Algorithm 1**

Algorithm 1

def reverse_engineer_veracrypt(veracrypt_file):

Args: veracrypt_file - The VeraCrypt file to be reverse engineered.

Returns: A dictionary of findings, including identified vulnerabilities and suspicious code patterns.

Remaining steps of Algorithm 1 is shown in Table 2 along with the different functions.

Table 2: Shows the steps and functions of Algorithm 1

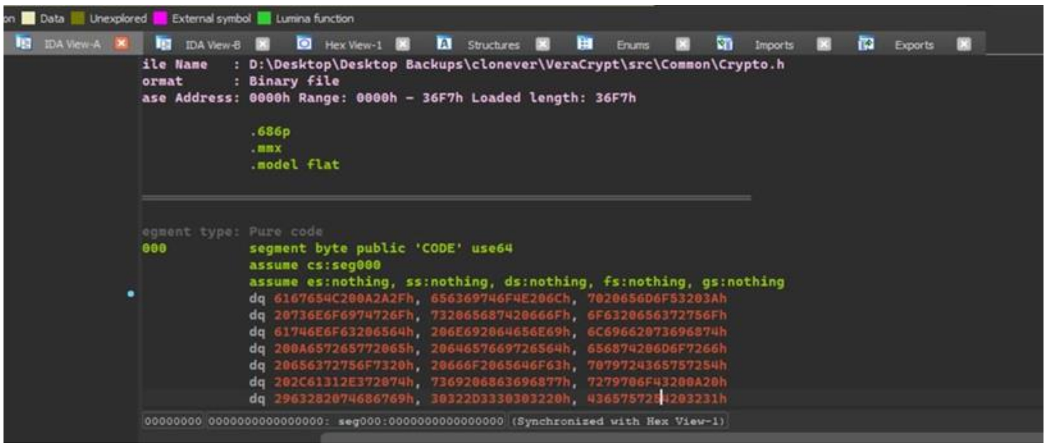
Step 1: System Structuring	ida_pro.open(veracrypt_file) identify_components() identify_entry_points_and_exit_points() identify_communication_channels()
Step 2: System Characterization	identify_modules() identify_functions() identify_data_structures()
Step 3: Static Analysis	identify_security_vulnerabilities() identify_suspicious_code_patterns()
Step 4: Dynamic Analysis	debug_veracrypt() track_data_flow()
Step 5: System Information Store Analysis	examine_configuration_files() examine_log_files() examine_databases()
Step 6: Document Generation	generate_documentation() return findings

4. Experimental Evaluation

The proposed framework is implemented in C++ on Visual Studio Platform. System configuration is given as follows:

- Processor: Intel Core i5 3217U
- Frequency: 1.8 GHz
- Memory (RAM): 8 GB
- Operating System: Windows 10 Home Edition

For system structuring, IDA pro is used. IDA Pro is considered a fundamental tool in the field of reverse engineering due to its advanced disassembly capabilities and a wide range of features that aid in understanding and analyzing complex binary code [40]. Fig. 3 and Fig 4 show the snapshots of IDA View and Hex View generated by IDA Pro.

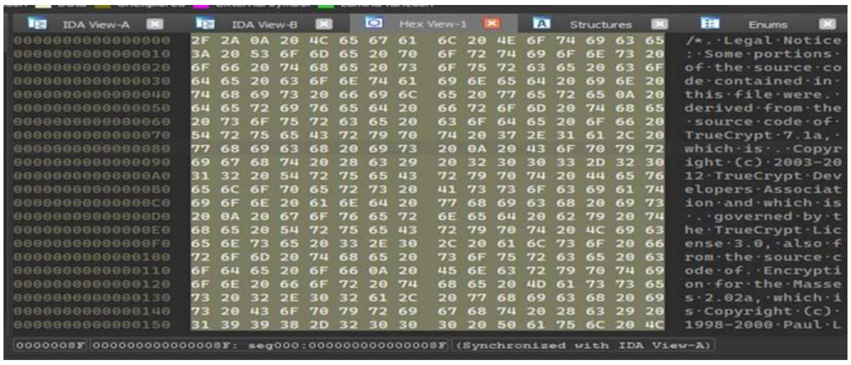


```
ile Name    : D:\Desktop\Desktop Backups\clonever\VeraCrypt\src\Common\Crypto.h
ormat       : Binary file
ase Address: 0000h Range: 0000h - 36F7h Loaded length: 36F7h

                .686p
                .mmx
                .model flat

egment type:  Pure code
000             segment byte public 'CODE' use64
                assume cs:seg000
                assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
                dq 6167654C200A2A2Fh, 656369746F4E206Ch, 7020656D6F53203Ah
                dq 20736E6F6974726Fh, 732065687420666Fh, 6F6320656372756Fh
                dq 61746E6F63206564h, 206E69206465E69h, 6C69662073696874h
                dq 200A657265772065h, 2064657669726564h, 656874206D6F7266h
                dq 20656372756F7320h, 20666F2065646F63h, 7079724365757254h
                dq 202C61312E372074h, 7369206863696877h, 7279706F43200A20h
                dq 2963282074686769h, 30322D3330303220h, 436575728420323lh
00000000 0000000000000000: seg000:0000000000000000 (Synchronized with Hex View-1)
```

Fig. 3. IDA View of VeraCrypt structured by IDA Pro



```
0000000000000000  2F 2A 0A 20 4C 65 67 61  6C 20 4E 6F 74 69 63 65  /*.·Legal·Notice
0000000000000010  3A 20 53 6F 6D 65 20 70  6F 72 74 69 6F 6E 73 20  :·Some·portions·
0000000000000020  6F 66 20 74 68 65 20 73  6F 75 72 63 65 20 63 6F  of·the·source·co
0000000000000030  64 65 20 63 6F 6E 74 61  69 6E 65 64 20 69 6E 20  de·contained·in·
0000000000000040  74 68 69 73 20 66 69 6C  65 20 77 65 72 65 0A 20  this·file·were.·
0000000000000050  64 65 72 69 76 65 64 20  66 72 6F 6D 20 74 68 65  derived·from·the
0000000000000060  20 73 6F 75 72 63 65 20  63 6F 64 65 20 6F 66 20  ·source·code·of·
0000000000000070  54 72 75 65 43 72 79 70  74 20 37 2E 31 61 2C 20  TrueCrypt·7.1a,·
0000000000000080  77 68 69 63 68 20 69 73  20 0A 20 43 6F 70 79 72  which·is·.·Copyr
0000000000000090  69 67 68 74 20 28 63 29  20 32 30 30 33 2D 32 30  ight·(c)·2003-20
00000000000000A0  31 32 20 54 72 75 65 43  72 79 70 74 20 44 65 76  12·TrueCrypt·Dev
00000000000000B0  65 6C 6F 70 65 72 73 20  41 73 73 6F 63 69 61 74  elopers·Associat
00000000000000C0  69 6F 6E 20 61 6E 64 20  77 68 69 63 68 20 69 73  ion·and·which·is
00000000000000D0  20 0A 20 67 6F 76 65 72  6E 65 64 20 62 79 20 74  ·.·governed·by·t
00000000000000E0  68 65 20 54 72 75 65 43  72 79 70 74 20 4C 69 63  he·TrueCrypt·Lic
00000000000000F0  65 6E 73 65 20 33 2E 30  2C 20 61 6C 73 6F 20 66  ense·3.0,·also·f
0000000000000100  72 6F 6D 20 74 68 65 20  73 6F 75 72 63 65 20 63  rom·the·source·c
0000000000000110  6F 64 65 20 6F 66 0A 20  45 6E 63 72 79 70 74 69  ode·of.·Encrypti
0000000000000120  6F 6E 20 66 6F 72 20 74  68 65 20 4D 61 73 73 65  on·for·the·Masse
0000000000000130  73 20 32 2E 30 32 61 2C  20 77 68 69 63 68 20 69  s·2.02a,·which·i
0000000000000140  73 20 43 6F 70 79 72 69  67 68 74 20 28 63 29 20  s·Copyright·(c)·
0000000000000150  31 39 39 38 2D 32 30 30  30 20 50 61 75 6C 20 4C  1998-2000·Paul·L
0000008F 000000000000008F: seg000:000000000000008F (Synchronized with IDA View-A)
```

Fig. 4 Hex View of VeraCrypt structured by IDA Pro

After system structuring, rest of the steps are performed. The performance of the framework is evaluated in terms of accuracy and time consumption. Accuracy is defined as a percentage of modules/functions identified in

VeraCrypt's architecture during the course of framework execution. Fig 5 shows the accuracy of framework with respect to the file size. File size is varied from 1000 to 5000 and the obtained results show proportional increase in the accuracy. As the file size increases, the framework gets more information to explore and examine. Therefore, the accuracy is increasing correspondingly.

On the other hand, time consumed by the framework to evaluate the structured VeraCrypt by IDA is evaluated with respect to the file size varied similar as above. Results (Fig. 6) show that the time consumption is acquiring stability as the file size increases. It is because as the file size increases, the framework already got adapted to the similar instances which consequently speed up the process.

5. Conclusions and Future Scope of Research

The advancement of digitalization has led to a surge in cybercrimes, prompting the development of digital forensic tools. In response, the emergence of anti-forensic tools aims to thwart these tools' effectiveness and hinder digital evidence recovery. To counter these techniques, software reverse engineering stands as a promising avenue for evaluation. This article explores the use of software reverse engineering techniques to assess anti-forensic encryption tools, with a specific focus on VeraCrypt—a free, open-source disk encryption software renowned for its robust on-the-fly encryption capabilities across various storage devices.

A novel framework is proposed herein, emphasizing practical applicability, data-driven analysis, and comparative assessment. This framework operates through a multi-phase process, namely system structuring, system characterization, static analysis, dynamic analysis, system information store analysis, and document generation. It leverages the potency of IDA Pro, a powerful disassembler and debugger tool, for meticulous examination. By systematically dissecting the architecture, components, and execution flow, the framework establishes a strong foundation for subsequent analyses.

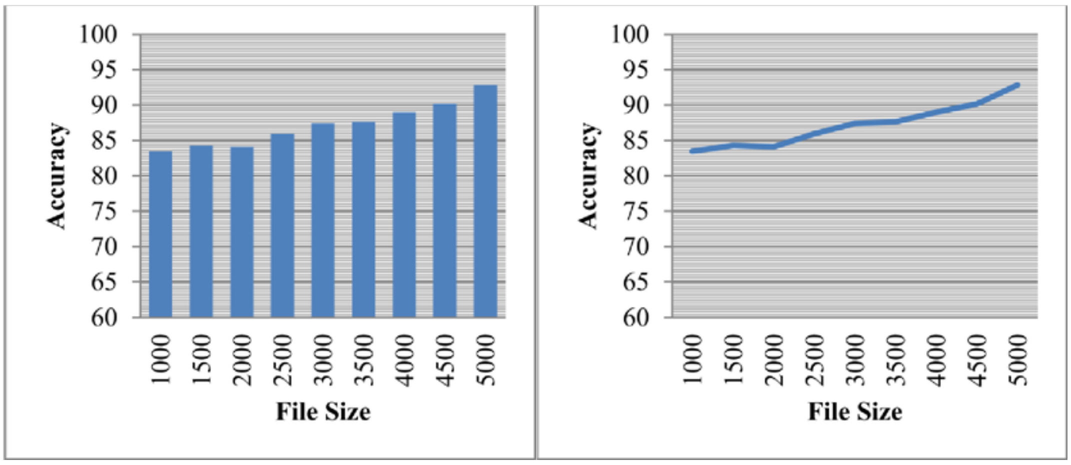


Fig. 5. Accuracy Vs File Size

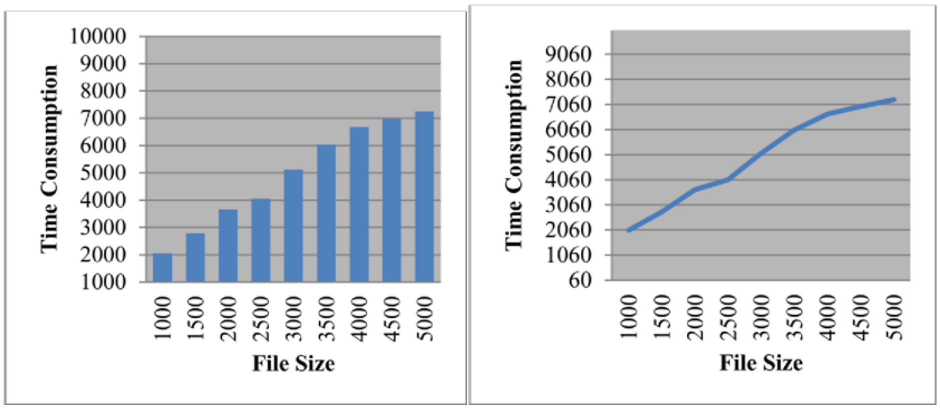


Fig. 6. Time Consumption Vs File Size

Extensive experiments are conducted to evaluate the framework's performance. Findings reveal that accuracy increases proportionally with the size of the analyzed files. Moreover, the framework exhibits a stabilization of time consumption as file size grows, indicating its efficiency in handling larger-scale analyses.

In conclusion, the proposed framework presents a comprehensive approach for evaluating anti-forensic encryption tools, particularly exemplified through the analysis of VeraCrypt. By fusing the power of software reverse engineering and sophisticated disassembly tools, the framework offers a promising solution to the evolving challenges posed by anti-forensic techniques. As digital forensics continues to grapple with cybersecurity threats, this framework offers a valuable contribution to the arsenal of techniques available for safeguarding digital evidence and ensuring a secure digital landscape. Further research and development in this area hold the potential to fortify cybersecurity measures and empower digital investigators in their pursuit of truth and justice.

The presented research lays the foundation for a multitude of promising future research directions. Firstly, exploring the integration of machine learning and artificial intelligence techniques into the proposed framework could enhance the accuracy and efficiency of anti-forensic tool evaluation. Leveraging these technologies could automate pattern recognition, anomaly detection, and decision-making processes, thereby reducing manual efforts and expanding the framework's capabilities. Additionally, extending the framework's applicability to other encryption tools and software systems would provide a broader understanding of its effectiveness and versatility. Further research could delve into real-world case studies, simulating various threat scenarios to validate the framework's practical applicability and resilience against sophisticated anti-forensic techniques. Moreover, investigations into emerging encryption methodologies, quantum-resistant encryption, and blockchain technology could offer insights into adapting the framework for future security challenges. Collaboration with legal experts to address ethical and legal considerations associated with software reverse engineering in anti-forensic evaluations is also a crucial avenue for exploration. As the digital landscape evolves, this research paves the way for ongoing innovation and adaptation, aligning with the ever-evolving dynamics of cyber threats and the imperative to safeguard digital evidence and security.

Conflict of Interest Statement

We, the authors, want to emphasize that we have no conflicts of interest related to publishing this article. Our research was conducted impartially, and there are no financial or personal connections that could affect our results.

References

[1] M. K. Rogers and K. Seigfried, "The future of computer forensics: A needs analysis survey," *Comput. Secur.*, vol. 23, no. 1, pp. 12–16, 2004, doi: 10.1016/j.cose.2004.01.003.

[2] M. A. Wani, A. AlZahrani, and W. A. Bhat, "File system anti-forensics – types, techniques and tools," *Comput. Fraud Secur.*, vol. 2020, no. 3, pp. 14–19, Mar. 2020, doi: 10.1016/S1361-3723(20)30030-0.

[3] J. P. A. Yaacoub, H. N. Noura, O. Salman, and A. Chehab, "Advanced digital forensics and anti-digital forensics for IoT systems: Techniques, limitations and recommendations," *Internet of Things*, vol. 19, p. 100544, Aug. 2022, doi: 10.1016/J.IOT.2022.100544.

[4] N. Goel and D. Ganotra, "An approach for anti-forensic contrast enhancement detection using grey level co-occurrence matrix and Zernike moments," *Int. J. Inf. Technol.*, vol. 15, no. 3, pp. 1625–1636, 2023, doi: 10.1007/s41870-023-01191-0.

[5] P. Nerurkar, M. Chandane, and S. Bhirud, "Understanding structure and behavior of systems: a network perspective," *Int. J. Inf. Technol.*, vol. 14, no. 2, pp. 1145–1159, 2019, doi: 10.1007/s41870-019-003542.

[6] H. Majed, H. N. Noura, and A. Chehab, "Overview of digital forensics and anti-forensics Techniques," in *8th International Symposium on Digital Forensics and Security, ISDFS 2020*, 2020, no. June. doi: 10.1109/ISDFS49300.2020.9116399.

[7] P. Minetola, L. Iuliano, and F. Calignano, "A customer-oriented methodology for reverse engineering software selection in the computer aided inspection scenario," *Comput. Ind.*, vol. 67, pp. 54–71, Feb. 2015, doi: 10.1016/J.COMPIND.2014.11.002.

[8] M. Gül and E. Kugu, "A survey on anti-forensics techniques," in *IDAP 2017 - International Artificial Intelligence and Data Processing Symposium*, 2017, no. September 2017. doi: 10.1109/IDAP.2017.8090341.

[9] M. A. Qureshi and E. S. M. El-Alfy, "Bibliography of digital image anti-forensics and anti-antiforensics techniques," *IET Image Process.*, vol. 13, no. 11, pp. 1811–1823, 2019, doi: 10.1049/ietipr.2018.6587.

[10] K. Conlan, I. Baggili, and F. Breitinger, "Anti-forensics: Furthering digital forensic science through a new extended, granular taxonomy," *Digit. Investig.*, vol. 18, pp. S66–S75, 2016, doi: 10.1016/j.diin.2016.04.006.

[11] K. Saleh and A. Boujarwah, "Communications software reverse engineering: a semi-automatic approach," *Inf. Softw. Technol.*, vol. 38, no. 6, pp. 379–390, Jun. 1996, doi: 10.1016/09505849(95)01061-0.

[12] W. A. Bhat, A. AlZahrani, and M. A. Wani, “Can computer forensic tools be trusted in digital investigations,” *Sci. &Justice*, vol. 61, no. 2, pp. 198–203, 2020, doi: 10.1016/j.scijus.2020.10.002.

[13] G. C. Kessler, “Anti-forensics and the digital investigator,” in *Proceedings of the 5th Australian Digital Forensics Conference*, 2007, pp. 1–7.

[14] “GitHub - veracrypt/VeraCrypt: Disk encryption with strong security based on TrueCrypt.” https://github.com/veracrypt/VeraCrypt (accessed Aug. 18, 2023).

[15] “TrueCrypt.” https://truecrypt.sourceforge.net/ (accessed Aug. 18, 2023).

[16] 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.

[17] A. R. Mothukur, A. Balla, D. H. Taylor, S. Teja Sirimalla, and K. Elleithy, “Investigation of countermeasures to anti-forensic methods,” in *2019 IEEE Long Island Systems, Applications and Technology Conference, LISAT 2019*, 2019, pp. 1–6. doi: 10.1109/LISAT.2019.8816826.

[18] A. K. Mandal, C. Parakash, and A. Tiwari, “Performance evaluation of cryptographic algorithms: DES and AES,” in *2012 IEEE Students’ Conference on Electrical, Electronics and Computer Science: Innovation for Humanity, SCEECS 2012*, 2012. doi: 10.1109/SCEECS.2012.6184991.

[19] M. Panda, “Performance Evaluation of Symmetric Encryption Algorithms for Information Security,” *Int. J. Adv. Res. Trends Eng. Technol.*, vol. 4, no. 11, pp. 37–41, 2017.

[20] K. Patel, “Performance analysis of AES, DES and Blowfish cryptographic algorithms on small and large data files,” *Int. J. Inf. Technol.*, vol. 11, no. 4, pp. 813–819, 2019, doi: 10.1007/s41870-018-0271-4.

[21] D. Gligoroski, “Cryptographic hash functions,” *A Multidiscip. Introd. to Inf. Secur.*, no. May, pp. 49–72, 2011, doi: 10.1587/essfr.4.57.

[22] H. Evkan *et al.*, “Security evaluation of VeraCrypt,” 2018.

[23] B. A. Ekanem and J. Meye, “Application of reverse engineering technique in software forensic analysis to detect infringements,” in *World Congress on Engineering*, 2021, pp. 191–194.

[24] Ramandeep Singh, “A Review of Reverse Engineering Theories and Tools,” *Int. J. Eng. Sci. Invent.*, vol. 2, no. 1, pp. 1–4, 2013.

[25] H. A. Müller and H. M. Kienle, “A Small Primer on Software Reverse Engineering A Small Primer on Software Reverse Engineering,” *Reverse Eng.*, no. March, 2009.

[26] T. Göbel and H. Baier, “Anti-forensic capacity and detection rating of hidden data in the ext4 filesystem,” *IFIP Adv. Inf. Commun. Technol.*, vol. 532, no. August 2018, pp. 87–110, 2018, doi: 10.1007/978-3-319-99277-8_6.

[27] 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.

[28] Adedayo M and Shoa Ying, “Privacy Impacts of Data Encryption on the Efficiency of Digital Forensics Technology,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 5, pp. 36–40, 2013, doi:

10.14569/ijacsa.2013.040506.

[29] M. Tajammul and R. Parveen, “Auto encryption algorithm for uploading data on cloud storage,” *Int. J. Inf. Technol.*, vol. 12, no. 3, pp. 831–837, 2020, doi: 10.1007/s41870-020-00441-9.

[30] A. Al-Dhaqm *et al.*, “Digital forensics subdomains: the state of the art and future directions,” *IEEE Access*, vol. 9, pp. 152476–152502, 2021, doi: 10.1109/ACCESS.2021.3124262.

[31] Dilbag Singh and Alit Singh “An Effective Technique for Data Security in Modern Cryptosystem,” *Int. J. Inf. Technol.*, vol. 2, no. 1, pp. 189–194, 2010.

[32] T. Sharma *et al.*, “A Survey on Machine Learning Techniques for Source Code Analysis,” vol. 0, no. 0, 2021.

[33] Christnatalis, A. M. Husein, M. Harahap, A. Dharma, and A. M. Simarmata, “Hybrid-AES-Blowfish algorithm: key exchange using neural network,” in *International Conference of Computer Science and Information Technology, ICoSNIKOM 2019*, 2019, pp. 4–7. doi: 10.1109/ICoSNIKOM48755.2019.9111500.

[34] M. Ahmad, M. Z. Alam, Z. Umayya, S. Khan, and F. Ahmad, “An image encryption approach using particle swarm optimization and chaotic map,” *Int. J. Inf. Technol.*, vol. 10, no. 3, pp. 247–255, 2018, doi: 10.1007/s41870-018-0099-y.

[35] Kumar, S.K.Muttoo and Sushil, “Data Hiding in JPEG Images,” *Int. J. Inf. Technol.*, vol. 1, no. 1, pp. 13–16, 2009.

[36] H. A. MÜLLER, K. WONG, and S. R. TILLEY, “Understanding Software Systems Using Reverse Engineering Technology,” *Object-Oriented Technol. Database Softw. Syst.*, pp. 240–252, 2000, doi:10.1142/9789812831163_0016.

[37] D. Binkley, “Source code analysis: A road map,” *FoSE 2007 Futur. Softw. Eng.*, pp. 104–119, 2007, doi: 10.1109/FOSE.2007.27.

[38] P. Tonella, M. Torchiano, B. Du Bois, and T. Systä, “Empirical studies in reverse engineering: state of the art and future trends,” *Empir. Softw. Eng.*, vol. 12, no. 5, pp. 551–571, 2007, doi: 10.1007/s10664007-9037-5.

[39] E. J. Chikofsky and J. H. Cross, “Reverse engineering and design recovery: a taxonomy,” *IEEE Softw.*, vol. 7, no. 1, pp. 13–17, 1990.

[40] “Hex Rays - State-of-the-art binary code analysis solutions.” https://hex-rays.com/ida-pro/ (accessed Aug. 19, 2023).

[41] Paul Garcia, Ian Martin, Laura López, Sigurðsson Ólafur, Matti Virtanen. Deep Learning Models for Intelligent Tutoring Systems. Kuwait Journal of Machine Learning, 2(1). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/167

[42] Khem, D. ., Panchal, S. ., & Bhatt, C. . (2023). An Overview of Context Capturing Techniques in NLP. International Journal on Recent and Innovation Trends in Computing and Communication, 11(4s), 193–198. https://doi.org/10.17762/ijritcc.v11i4s.6440

[43] Sherje, N. P., Agrawal, S. A., Umbarkar, A. M., Dharme, A. M., & Dhabliya, D. (2021). Experimental evaluation of mechatronics based cushioning performance in hydraulic cylinder. Materials Today: Proceedings, doi:10.1016/j.matpr.2020.12.1021