ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

# An Evaluation Framework for Anti-Forensic Encryption Tools Through Software Reverse Engineering Methods

Zakariyya Hassan Abdullahi[1]; Zainab Suleiman Abdullahi[2]; Kabiru Bashir[3]

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

Publication Date: 2025/08/25

**Abstract: The widespread adoption of encryption technologies has raised concerns about the protection and vulnerability of digital data. Protocol reverse engineering (PRE) is a critical methodology for evaluating and validating encryption implementations. It involves analyzing network traffic, message logging, and model checking processes. The key security properties of encryption include confidentiality, integrity, availability, and non-repudiation. However, the dual-use nature of encryption presents challenges for digital forensics and law enforcement investigations. Malicious actors can exploit encryption to conceal criminal activities and obstruct justice. Digital investigators and forensic specialists must develop expertise in specialized decryption tools, steganographic detection methods, and advanced analytical techniques to uncover hidden or obfuscated data. Cryptographic service implementations vary significantly in performance characteristics and security effectiveness, with key size, algorithm type, encryption rounds, algorithm complexity, and data size influencing performance.**

***Keywords:*** *Reverse Engineering, Encryption, Protocol.*

**How to Cite**: Zakariyya Hassan Abdullahi; Zainab Suleiman Abdullahi; Kabiru Bashir(2025), An Evaluation Framework for Anti-Forensic Encryption Tools Through Software Reverse Engineering Methods. *International Journal of Innovative Science and Research Technology*, 10(8), 1095-1106. https://doi.org/10.38124/ijisrt/25aug411

## I. INTRODUCTION

In today's digital age, nearly every aspect of our lives is influenced by digital technology and supported by digital data. Numerical data plays a significant role across various fields, highlighting the widespread integration of technology. Individuals, businesses, and governments rely on digital data for many purposes, including investigating cybercrime, terrorism, and common criminal activities, where such data can offer critical insights. The growth of digital forensics has played a key role in enabling this expanded use of digital information [1][2]. The presence of digital forensics tools has also acted as a deterrent to hackers, threat actors, and privacy-focused individuals, prompting the development of anti-forensics tools aimed at undermining the effectiveness of forensic tools (FT) in retrieving valuable and relevant information. Computer Forensics Tools (CFT) and Mobile Forensics Tools (MFT) support forensic examiners in extracting evidence from digital devices.

Anti-forensics tools (AFT) and techniques are employed to compromise the accessibility and usefulness of digital evidence by altering, disrupting, or eliminating its scientific reliability [3] [4] Based on their purpose and application, anti-forensics tools (AFT) can take various forms, such as artifact wiping. With technological advancements, forensic professionals now use sophisticated methods to conduct investigations more efficiently, accurately, and decisively. However, cybercriminals are also leveraging these same technological advancements to develop advanced, customized techniques designed to obstruct and mislead forensic investigations [5] While the concept of "anti-forensics" is not new, it lacks a universally accepted definition within engineering or academic communities. In this context, the authors have proposed a definition, suggesting that anti-forensics refers to efforts aimed at negatively impacting the presence, amount, or integrity of evidence from a crime scene, or at making the examination and interpretation of that evidence difficult or impossible [6] A digital forensics expert and investigator defines anti-forensics (AF) as efforts to negatively influence the nature, quantity, or quality of evidence from a crime scene. However, those skilled in anti-forensic techniques and tools often adopt a more critical perspective, describing it as the use of scientific methods on digital media specifically to eliminate or obscure information, preventing it from being examined in a legal or judicial contex.[7].

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

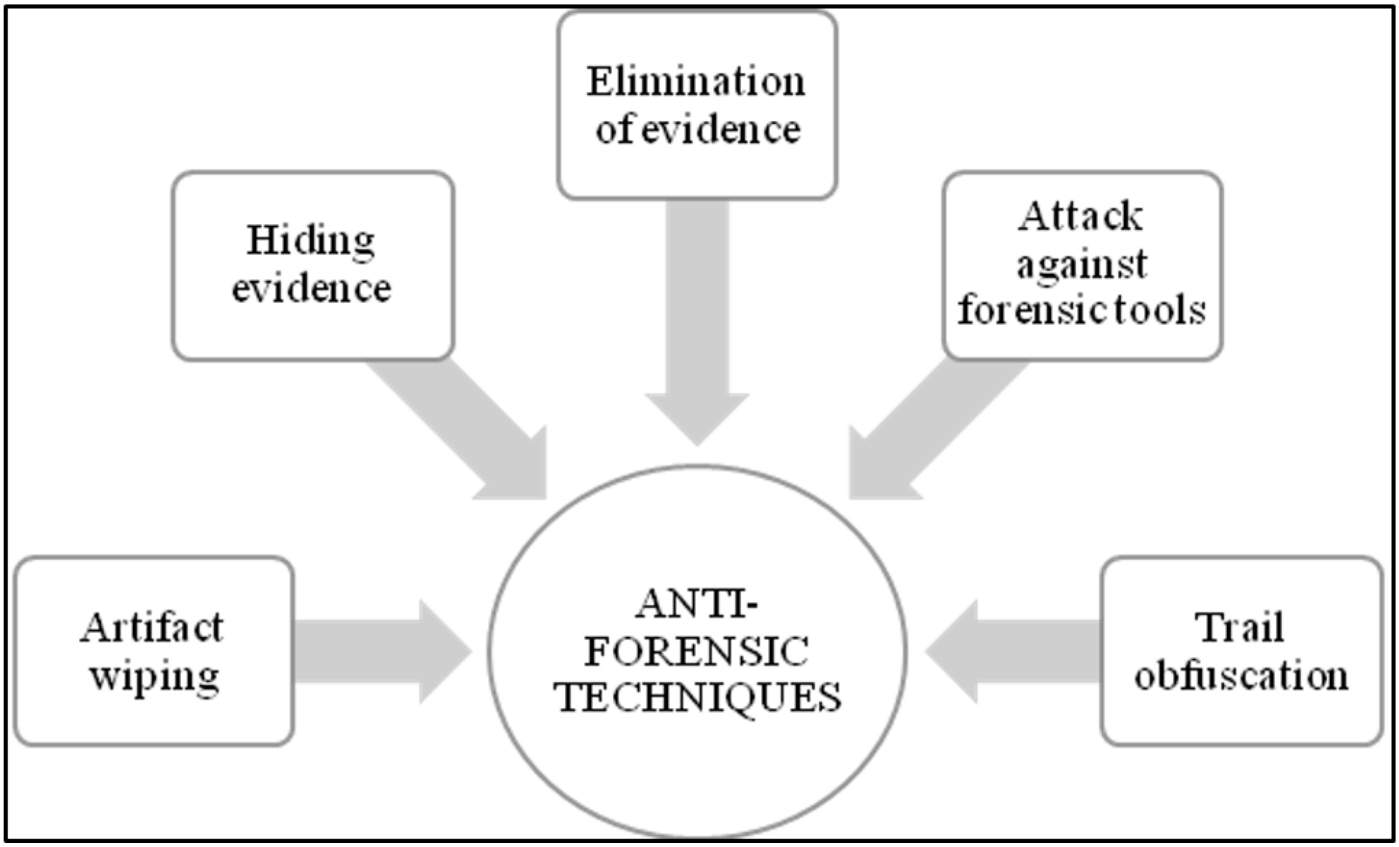


Fig1 Classification of Anti-Forensics Techniques

## II. ENCRYPTION

Data encryption functions as a security barrier that blocks unauthorized individuals from accessing stored information. This protective measure can be implemented at various levels - securing single files, protecting databases, safeguarding email communications, or encrypting complete storage drives through the use of multiple cryptographic algorithms.

In contemporary digital environments, encrypting data has become a fundamental security requirement. Electronic devices such as laptops, personal digital assistants, USB flash drives, smartphones, and external hard drives are prime targets for security breaches, as they store valuable user information that attracts malicious actors. According to research conducted by Intel Corporation in 2017, approximately half of all stolen laptop computers lacked any form of encryption protection, leaving their stored data completely exposed and accessible to thieves [8]. In the modern era, protecting data information has become a vital concern for individuals across all sectors. People invest substantial amounts of money—often reaching hundreds of dollars—to safeguard their digital information as a necessity for maintaining their competitive edge and business viability. The unauthorized disclosure or breach of critical data can result in devastating and irreversible consequences that may be impossible to recover from [9]. Data Information security is the most critical type of security-more important than network security-since only securely encrypted data can be safely transmitted.

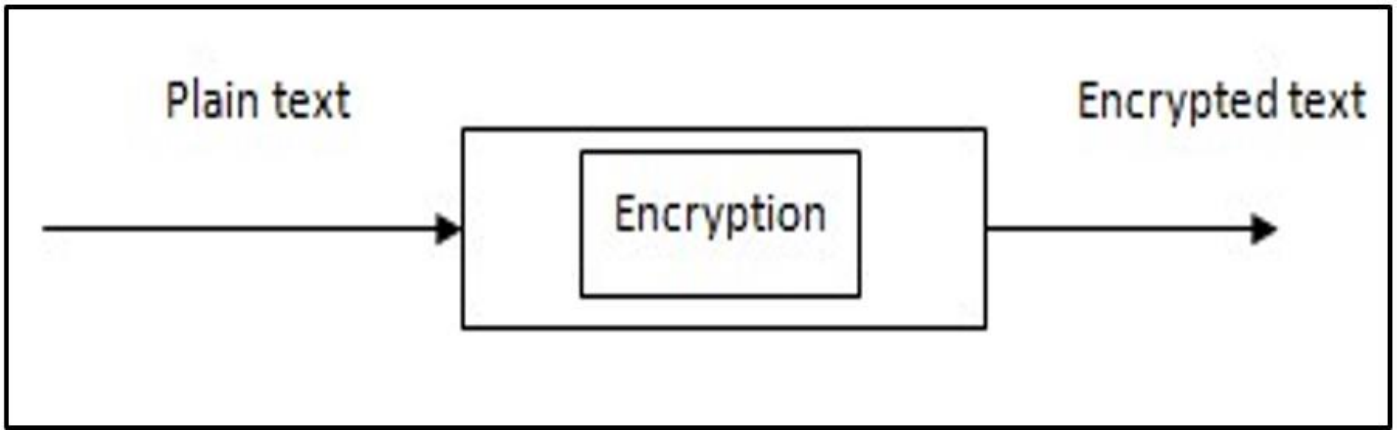


Fig 2 Encryption

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

VeraCrypt is an open-source on-the-fly encryption (OTFE) software that offers full support for Windows XP and includes features for plausible deniability. TrueCrypt version 1.0 was compatible with Windows 98/ME and Windows 2000/XP, but starting with version 2.0a, support for Windows 98 and ME was discontinued [10]

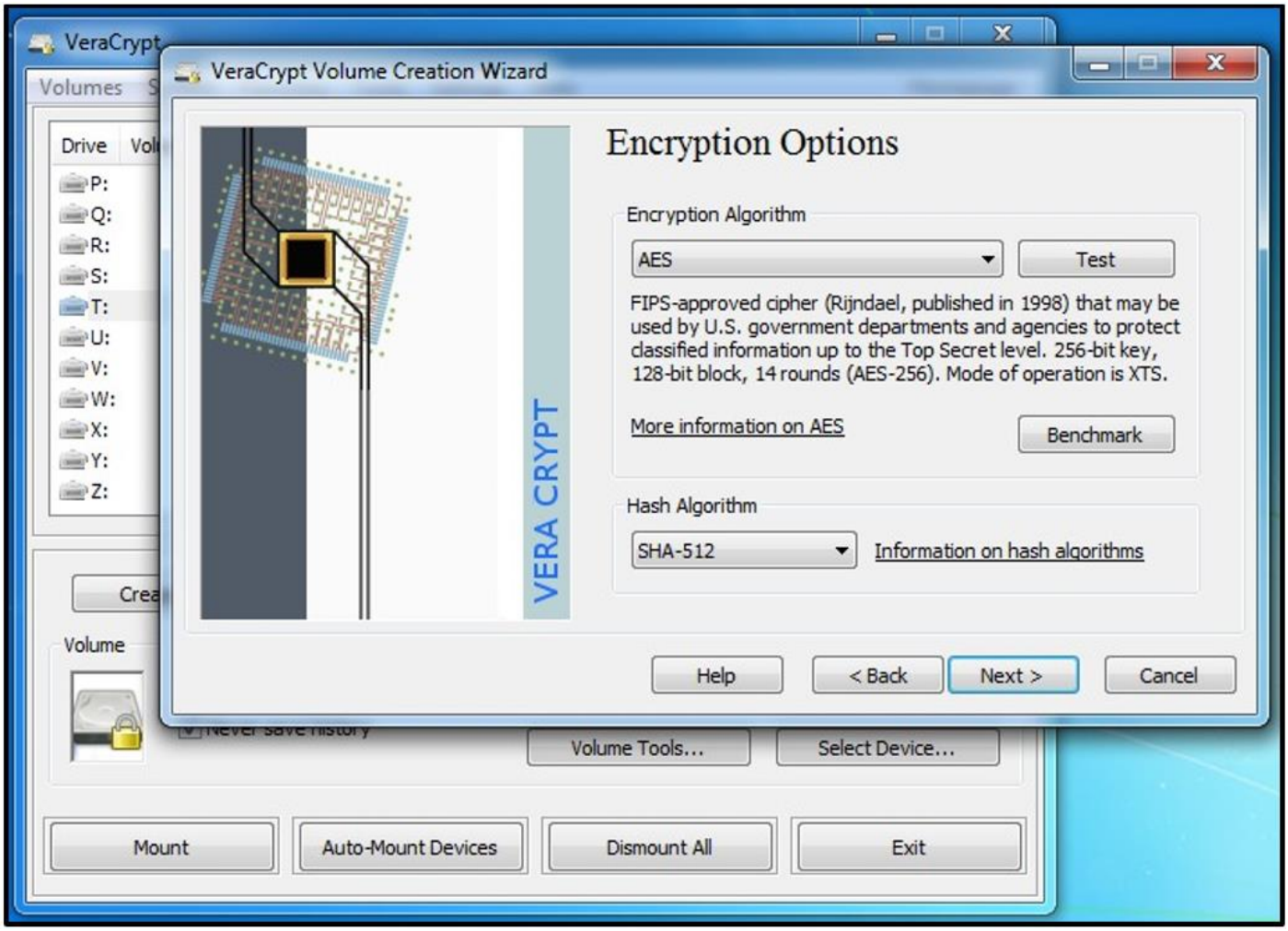


Fig 3 VeraCrypt Software (sources https;//www.idrix.fr)

## III. VALUE ADDED TO KNOWLEDGE

The article contributes to the strategy used by the author to develop higher-level abstract representations of a system. This involves accurately identifying subsystem interfaces and pinpointing key components and their interconnections within complex, multi-layered subsystem structures. The method, rooted in software reverse engineering principles, includes steps such as offline analysis and the identification of modules and components. Compared to automated approaches commonly used in earlier research, this system-level technique typically achieves a more effective decomposition of the system. Offline code analysis proves to be a powerful tool, as it provides a clear understanding of the program and facilitates the targeted search for specific functions of interest.

## IV. RELATED WORK

Software reverse engineering has long been employed to analyze and comprehend the logic, architecture, and design of code. Through reverse engineering an application, its code structure can be uncovered and partially reconstructed. Malware creators exploit this process to clone apps by reusing code and repackaging them. These malicious actors focus on such apps due to the open nature of the Machine App Market and the insufficient security testing carried out by developers, enabling the creation and spread of mobile malware. This perspective is supported by Gonzalez and colleagues [11]. Examine several offline and online techniques for detecting repackaging by identifying software similarities through different attributes and metrics in detail . Including Lim et al.[12]. Protocol reverse engineering (PRE) uses communication analysis to approximate the definition of a protocol. Unlike reverse engineering of executable program binaries, which focuses on recovering source code or understanding a program's implementation, PRE aims to infer the program's behavior. However, PRE is not limited to communication analysis alone and can also incorporate software reverse engineering to gather communication details. Techniques for software reverse engineering differ fundamentally from those used in traffic analysis. Software reverse engineering requires specialized tools, methods, and analytical processes, as well

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

as a certain level of expertise from the analyst. Additionally, beyond PRE, the techniques used for entity analysis are also well-established and widely applied[13]. By reverse engineering a node's software implementation, entity analysis is able to deduce the underlying protocol (e.g., [14]). To apply software control flow analysis and memory introspection techniques, entity analysis requires access to both the program and its execution environment. However, acquiring the program or a suitable environment for its execution is often not feasible, rendering entity analysis methods unusable in such cases. In contrast, traffic trace analysis remains a viable alternative when reverse engineering the executable is not possible, as it focuses solely on the observable communication between entities. This form of analysis is non-intrusive and does not require control over any of the entities involved, although it is limited to the information that can be seen on the communication channel. Rauch [12] In 2006, efforts to automate protocol reverse engineering (PRE) were discussed, highlighting steps to reduce the repetitive manual work typically involved in the PRE process. However, the tool showcased at Black Hat was never released publicly, which hindered ongoing progress in the PRE field. As a result, even more than a decade later, the standard PRE process largely remains manual, despite the introduction of various techniques offering partial automation. Stroulia, Eleni and colleagues concentrated on program comprehension, aiming to model the structure of software by analyzing its COBOL and C source code. [15] Reverse engineering generates higher-level abstractions and offers architectural perspectives on the overall structure of complex software systems [16]. Various methods have been created to automate the evaluation of sources and code snippets; however, many algorithms tend to overlook or underestimate important characteristics of the source code. To address this issue Diamantopoulos et al. [17]

Reverse engineering is not a recent concept. The idea of reversing occurs anytime someone analyzes another person's source code. It can also take place when a developer revisits and examines their own code days after originally writing it. Using reverse engineering is a method of discovery. When we look at code again, whether it was written by us or someone else, we investigate, we learn, and we may see things that weren't there before.[18] David et al. [19] exposed the structure of source or executable code and found six significant static features. Based on these six different features, authors categorised files as benign or malicious. The six most crucial factors are Section Alignment, Compilation Time, Size of Image, File Alignment, File info, and Size of Header. The feature models from large-scale projects can be reverse-engineered with the help of this work. There are still numerous remaining issues in the fields of feature location and dependency mining that must be resolved for these techniques to be implemented in useful tools.[20], One of the hardest tasks is retrieving the pertinent component from the repository. Tasks. The assessment and evaluation of software components are based on a number of factors. Due to different technological needs, goals, and business considerations.[21] For the feature descriptions of software components, the CBSE process also makes use of ontologies. Components that ultimately aid in the matching process. Several methods have been put forth for component retrievals that use modified versions of keyword-based and signature-matching techniques [22][23]. Several tools in addition to ours have been created in light of the findings. software development to be modelled. These tools' disadvantage is that they only use a few basic capabilities.[24]. Neural Networks are proposed to compose key public for secure files, with 99.98% accuracy for AES, Blowfish, and Hybrid algorithms.[25] This study will combine two Cryptographic Algorithms, AES (Advanced Encryption Standard) and Two fish (256 bits key generated by HASH function SHA 256), to give more security to data uploaded or downloaded in the Cloud system.[26]

Table 1 Literature Survey

| Author/References | Approach, Methods | Software Used | Limitations | Remarks |
|---|---|---|---|---|
| Gonzalez, A et. al, 2015 [16] | String Offset Order | DexGuard and HoseDex2Jar | Works on Android Malware Genome Project Apps | Works on Android Apps |
| Antunes, J et.al 2012 [16] | Methodology Was Implemented FTP Protocol D | ReverX Tools | Does not require any access to a protocol implementation or its source code | To extending the methods to, support binary files |
| David, Baptiste Filiol, Eric Gallienne, Kévin,2017 [11] | static and dynamic detection techniques | DAVFI/OpenDAVFi AV software | Focus on a blind detection on a collection of suspicious software | Malware suspicious |
| Debray, S, et. al, 20015 [18] | Cloning dynamic analyses Methods | SPECint-2000 benchmark suite | Regarding techniques for understanding obfuscated code and the strengths and weaknesses of | Code obfuscation has been proposed make it difficult to reverse engineer software. |

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

| | | | | |
|---|---|---|---|---|
| | | | sophisticated obfuscation algorithms | |
| Kaizheng Liu et.al 2020 | Manual Reverse Engineering Framework | Eembedded Linux based IoT systems using either read-only or writable filesystems. | 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. |
| P. Swierczynski et al.2021 | | | | |
| Michael Werner et.al, 2018 | graph analysis methods | AES, DES, ECC and RSA | Reverse Engineering process for integrated circuits image | Utilizing Cryptographic Architecture. |
| Stroulia, Eleni et.al,2002 [16] | Dynamic reverse engineering techniques methods | COBOL and C CODE GRAS, a database | Concentrating on software comprehension through the creation of models that represent a program's architecture by analyzing its underlying code structure. | Complexity of the software systems being developed increases the complexity of the systems that need to understood also increases |
| Wong, Kenny,2000 [11] | The proposed strategy focuses on approaches for software reverse engineering and redocumentation. | lexical, syntactic, and semantic analysis | redocument current software architectures | facilitating ongoing software understanding |
| S, Steven et.al 2011 [20] | Models based on a critical heuristic for identifying code | Linux, e Cos and FreeBSD, kernels. | Discovering Feature Groupings, Creating the Feature Hierarchy, | In the field of feature location, there are a lot of unresolved issues. |
| A Magbool et al.2023. [21] | Approach uses a machine learning approach to train the schema | Crawling Software Repositories | software ranks code snippets in the top k results using learned schema. | Enhancing tool performance requires the use of new capabilities. |
| S.Bajracharya et al.[22] | Code Rank approach | machine learning techniques to | Utilises The Fundamental Coder- Ank Notation, Which Exclusively Retrieves Structural Data. | Seeing strong power-law behavior across several code entities |
| Diamantopoulos et al.[17] | QualBoa, Methods | Code Search Engines ( | This instrument combines useful and excellent qualities. | Using the functional score to rank the various components |
| Christnatalis et.al 2019 [2] | AES Blowfish | 2Tier | Dual encryptions | Used public keyalgorithms |
| K. I. Santoso 2020 | AES Blowfish | 2 Tier | Increase computational cost | Model is the best |

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

## V. REVERSER ENGINEERING

Reverse engineering is an essential set of techniques and tools used to uncover the inner workings of software. More precisely, it involves analyzing an existing system to identify its components, understand how they interact, and create alternate representations of the system—often at a higher level of abstraction.[27]. Through reverse engineering, we gain insight into a software system's architecture, operational methods, and the features that drive its behavior. This approach, supported by automated analysis tools and evaluation techniques, provides a practical way to understand the complexity of software and uncover its underlying structure. While reverse engineering is not new, it occurs more often than we might think—whenever someone studies another's code, they are essentially engaging in a form of reverse engineering. Even revisiting one's own code after some time can become a reverse engineering exercise. Ultimately, it is a process of discovery, often revealing unexpected elements as we explore and learn from existing code[18]. Protocol reverse engineering (PRE) focuses on understanding communication protocols by carefully analyzing interactions, distinguishing it from other reverse engineering goals like software binary analysis, which typically aims to recover source code or gain insight into how an application functions. While PRE often involves these objectives, its scope is not limited to them. This paper presents a reverse engineering methodology and a corresponding tool used to study how network-based intrusion detection systems match signatures. The findings from this analysis can be applied to create modified attacks that avoid detection or generate benign traffic that overwhelms the detection system [28]. Multiple analytical approaches are employed to evaluate communication systems. As a result, there are two fundamental methodologies for comprehending these processes: entity analysis and trace analysis, which serve as the two distinct protocols for investigation [19]. as shown in Fig. 4 below.

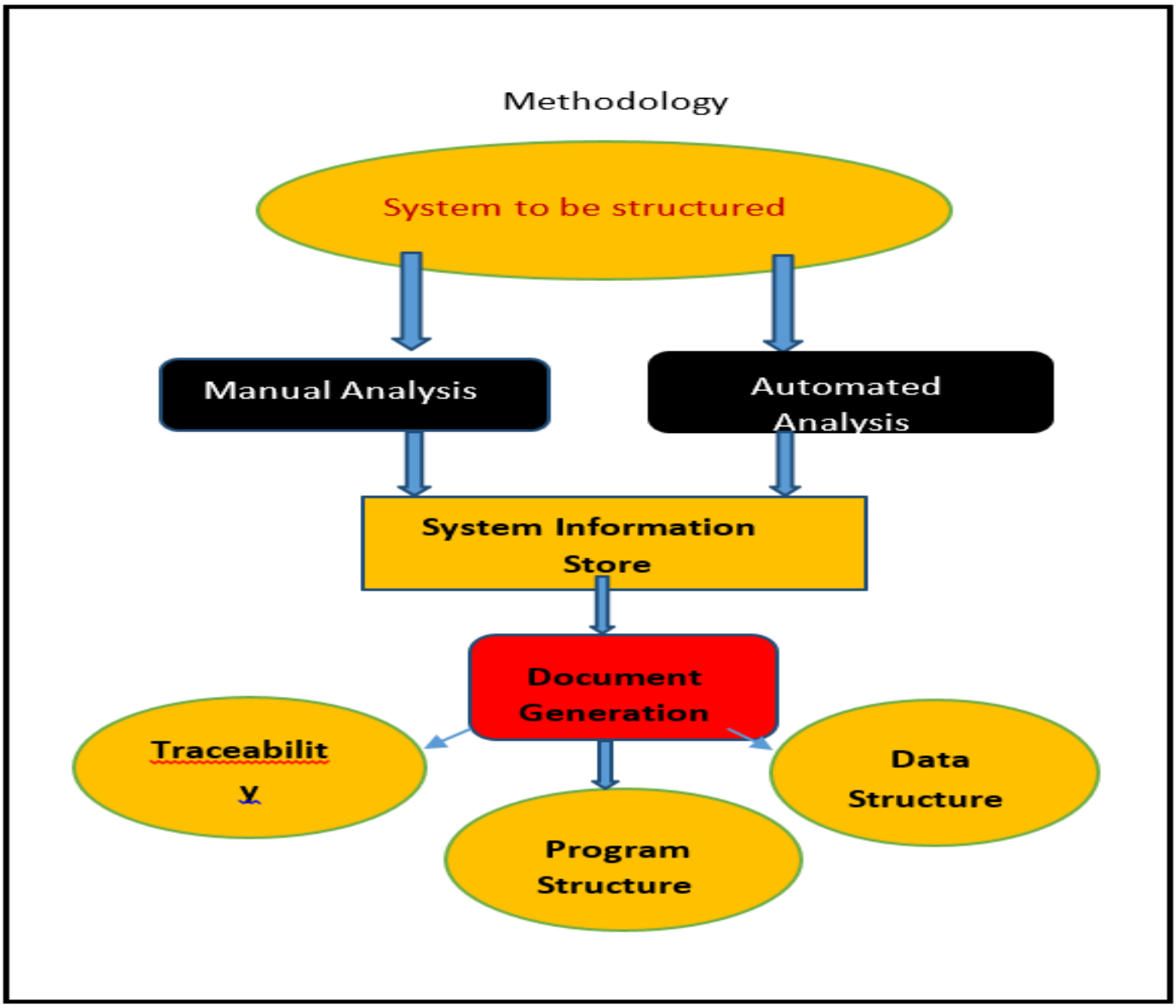


Fig 4 Reverser Engineering Process

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

IDA Pro disassembles binary programs to generate execution maps, displaying the instructions in assembly language. Since assembly code can be difficult to understand, advanced versions of IDA Pro incorporate techniques that improve readability and make the analysis more accessible [29].

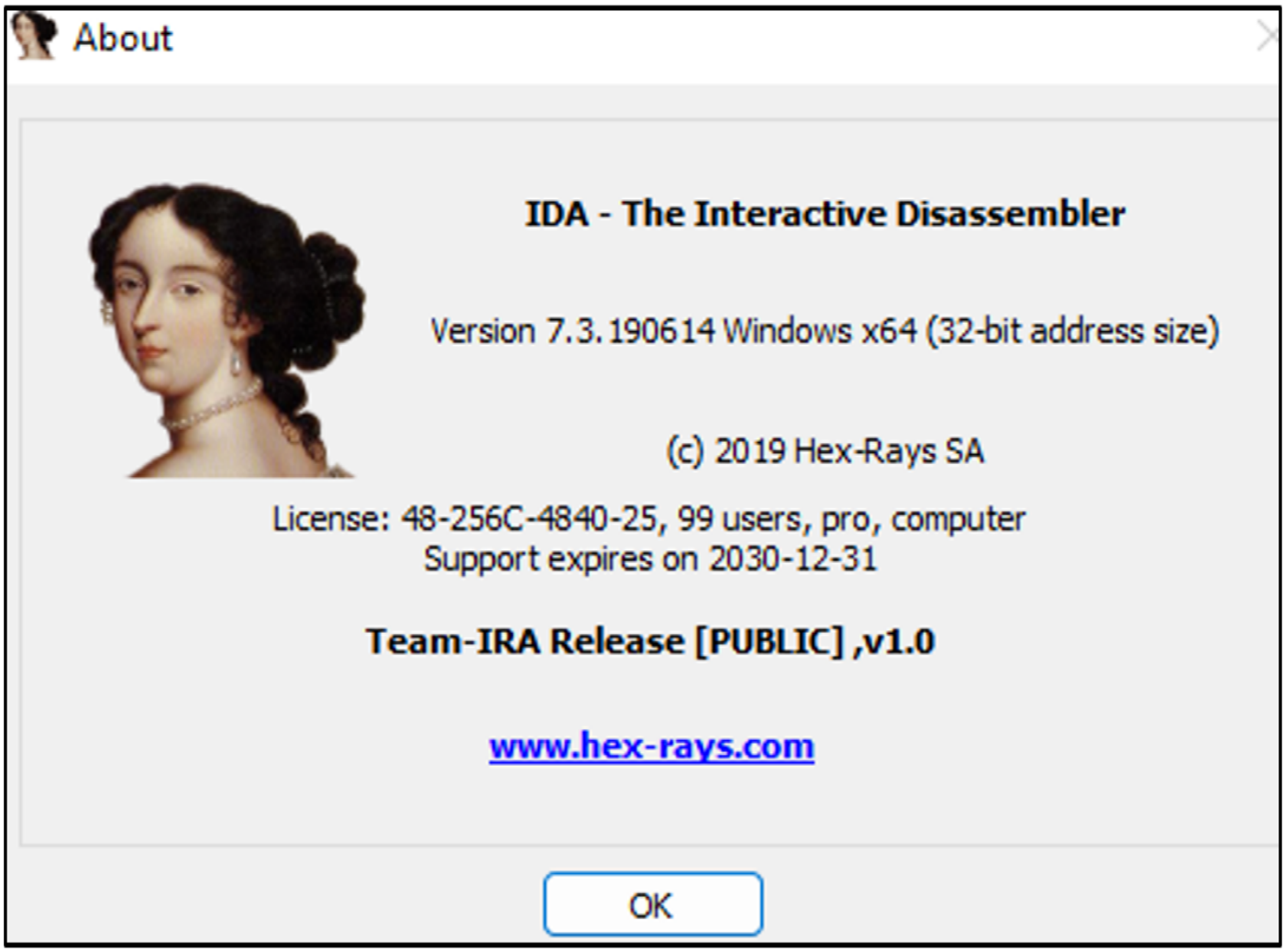


Fig 5 IDA Pro Reverser Engineering Tools

## VI. RESULTS AND DISCUSSION

The expected outcomes include both the source code and the binary executable files of the VeraCrypt encryption software. To achieve this, two primary analysis approaches are typically employed: static analysis and dynamic analysis.

**Static analysis** involves examining the program's code or binary files without executing them. This method allows the analyst to explore the structure, logic, and potential vulnerabilities of the software by inspecting the codebase, disassembled instructions, control flow graphs, and data structures. Tools like IDA Pro or Ghidra are often used in this phase to reverse engineer and understand the underlying implementation of VeraCrypt.

**Dynamic analysis**, on the other hand, involves observing the program's behavior during execution. This includes monitoring how the software interacts with the system, how it handles inputs, allocates memory, or responds to certain triggers. Techniques such as debugging, runtime tracing, and system call monitoring help provide insights into the real-time functioning of VeraCrypt, particularly the encryption and decryption processes.

By combining both static and dynamic analysis, a comprehensive understanding of the VeraCrypt encryption system can be developed, enabling the reconstruction of its source code, evaluation of its security mechanisms, and potential discovery of hidden or undocumented features.

Static analysis is a technique for learning as much as you can without actually executing a binary.[30]. To accomplish this, reverse engineering and disassembly techniques are employed. In addition, more specialized analysis methods may be used, such as string analysis, code obfuscation handling, restricted execution environments, unpacking, and other related techniques.

Static analysis involves breaking down the internal structure of the malware without running it. This typically includes loading the executable into tools like IDA Pro to

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

disassemble the code, followed by a thorough examination of the disassembled output and any available documentation. The goal is to understand the program's functionality, logic, and potentially malicious behavior through careful inspection of its code [31] . debugger is employed in dynamic analysis to look into the internal state of an active malicious attack [32] [31]. A debugger is a hardware or software tool utilized to examine and verify the operation of other programs. Dynamic analysis methods are implemented to gather comprehensive details about a program's execution and behavior. Three prominent debuggers used for botnet detection include [the text appears incomplete here]. Unlike alternative methodologies that present greater complexity and difficulty in understanding, this article presented a comprehensive reverse engineering approach in a clear and accessible format to analyze the internal mechanics and interpretive functions of software applications.The fig

Below shows a details module of VeraCrypt's IDRIX is a software development company that specialises in encryption and security solutions. VeraCrypt v1.24 is free and open-source encryption software for Windows, Mac, and Linux operating systems. It offers five encryption combinations, each of which uses a combination of two or three encryption algorithms in a fixed order to provide enhanced security. XTS mode is used for encryption, which provides enhanced security against certain types of attacks. The number of tiers refers to the number of encryption algorithms used in the combination, with 2-tier combinations using dual encryption and 3-tier combinations using triple encryption., In order to compare performance, hybrid models were developed using Visual Studio 2013, C++, Crypto, Windows 10 Home Single Language OS 64-bit architecture, 8 GB RAM, and Intel CoreTM i3-3217U CPU.

Table 2 Vera Crypts Algorithms Modules

| | | | |
|---|---|---|---|
| IDRIX VeraCrypt v1.24 Oct-2019 | AES-TWOFISH | 2 Tier | Dual encryption in fixed order.<br>Each block of 128-bit encrypted with Twofish then with AES in XTS Mode. |
| | AES-Serpent | 2 Tier | Dual encryption in fixed order.<br>Each block of 128-bit encrypted with AES then with Serpent in XTS Mode. |
| | Serpent-Twofish | 2 Tier | Dual encryption in fixed order.<br>Each block of 128-bit encrypted with Serpent then with Twofish in XTS Mode. |
| | AES-TwofishSerpent | 3 Tier | Triple encryption in fixed order.<br>Each block of 128-bit encrypted with AES, Twofish and Serpent in XTS Mode. |
| | Serpent-TwofishAES | 3 Tier | Triple encryption in fixed order.<br>Each block of 128-bit encrypted with Serpent, Twofish and AES in XTS Mode. |

```
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 A Collection of Recovered Source Code Segments Related to Encryption Algorithm

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

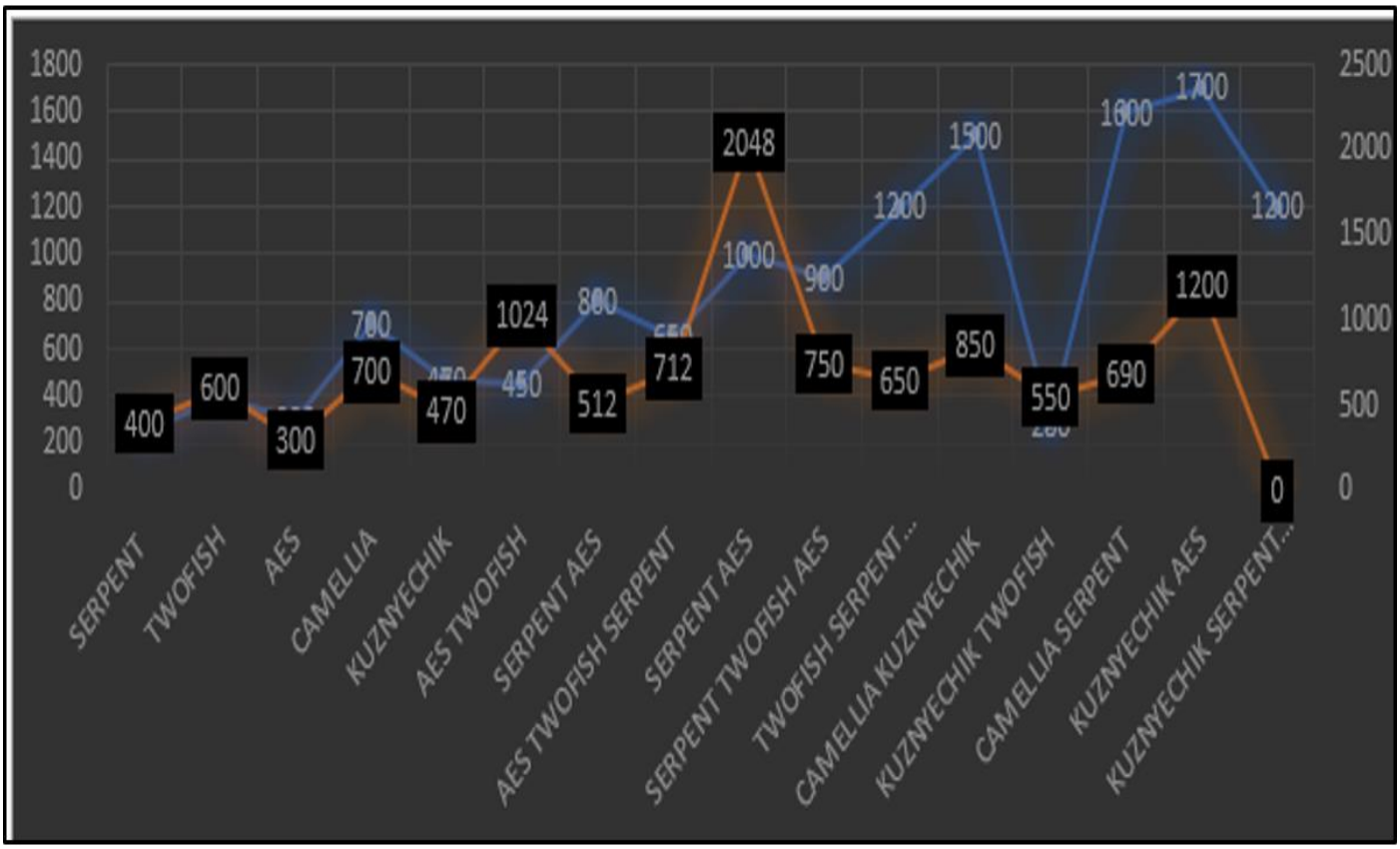


Fig 7 VeraCrypt's Encryption Algorithms

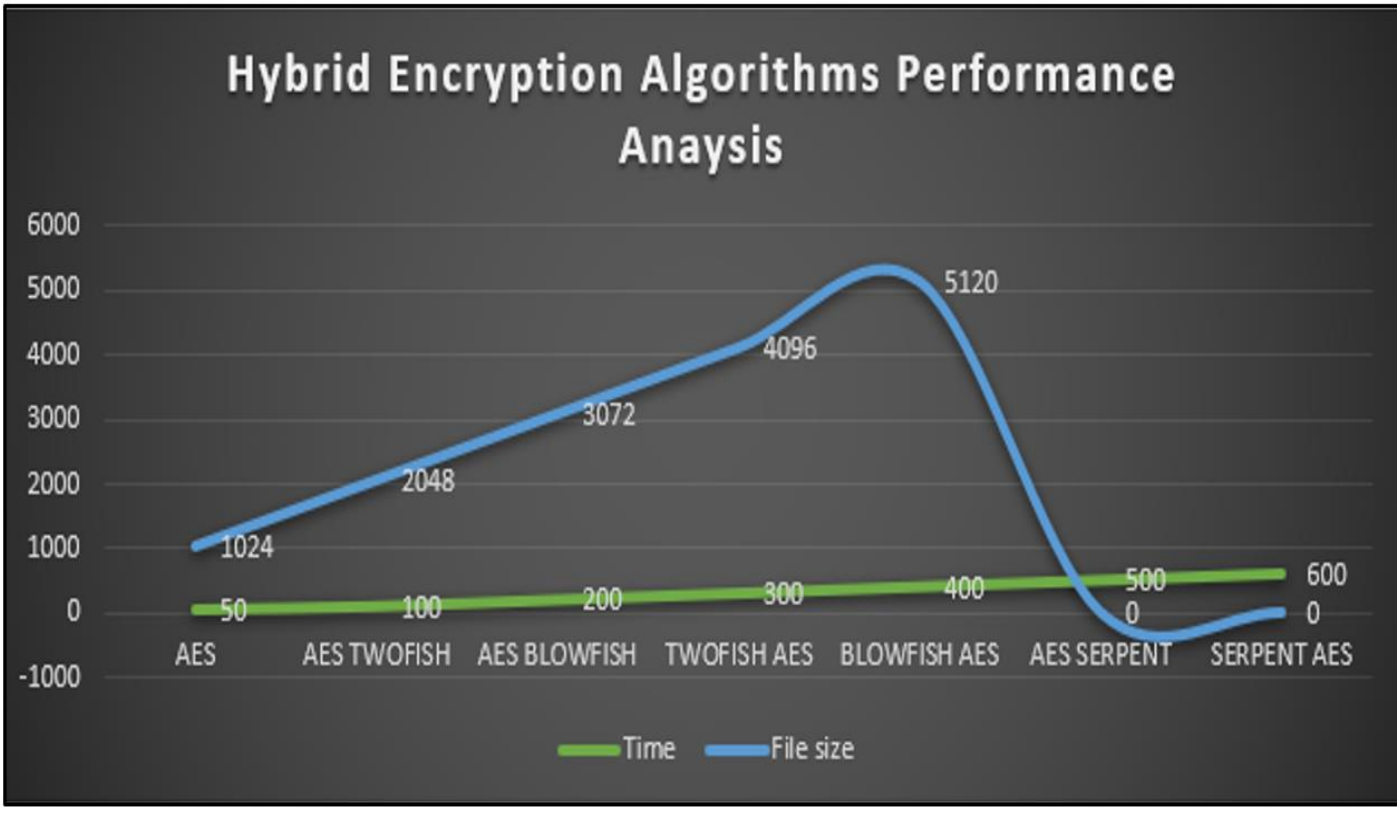


Fig 8 Vera Crypt's Encryption Algorithms Two Tier and Three Tier Performance Analysis

AES-TWOFISH or TWOFISH-AES model performance is superior in a hybrid cascaded 2-tier model, while AES-TWOFISH-SERPENT or SERPENTTWOFISH-AES model performance is nearly comparable in a hybrid cascaded 3-tier model.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

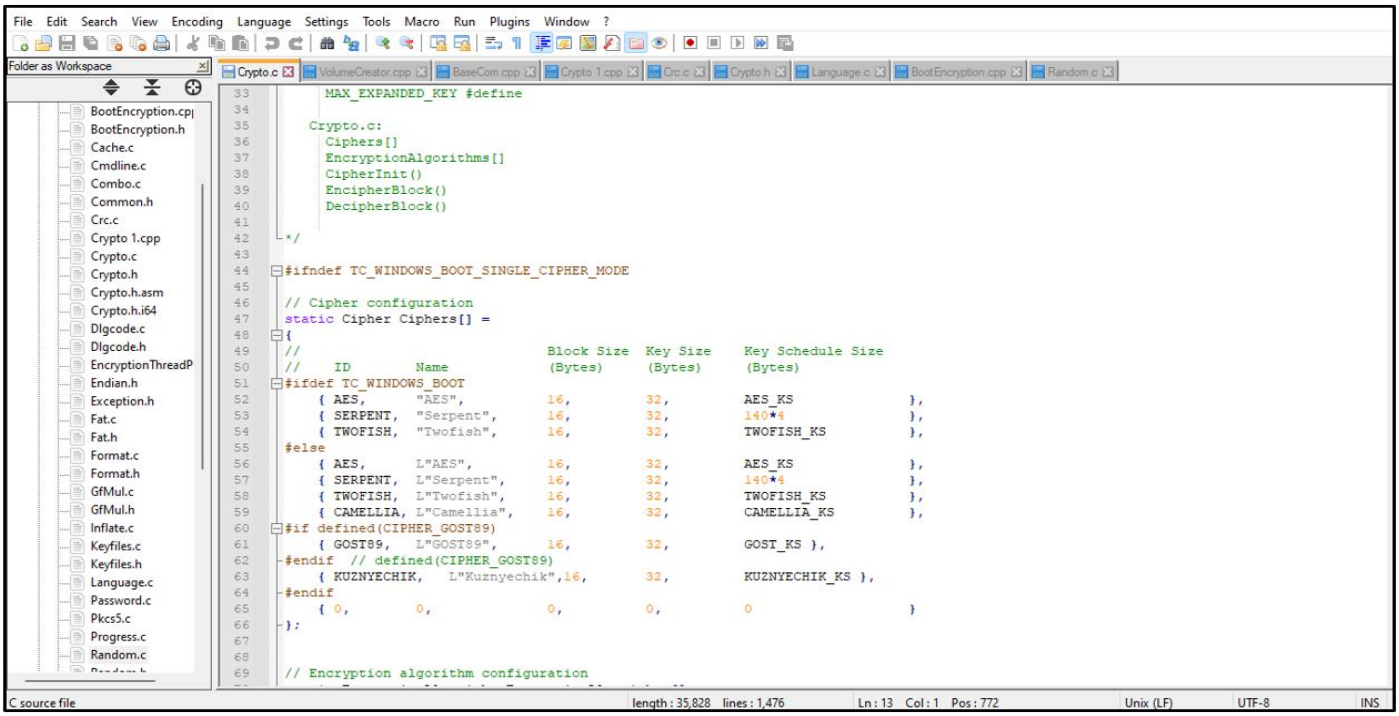


Fig 9 A Sample of Retrieved Source Code Snippets used in Encryption Algorithms

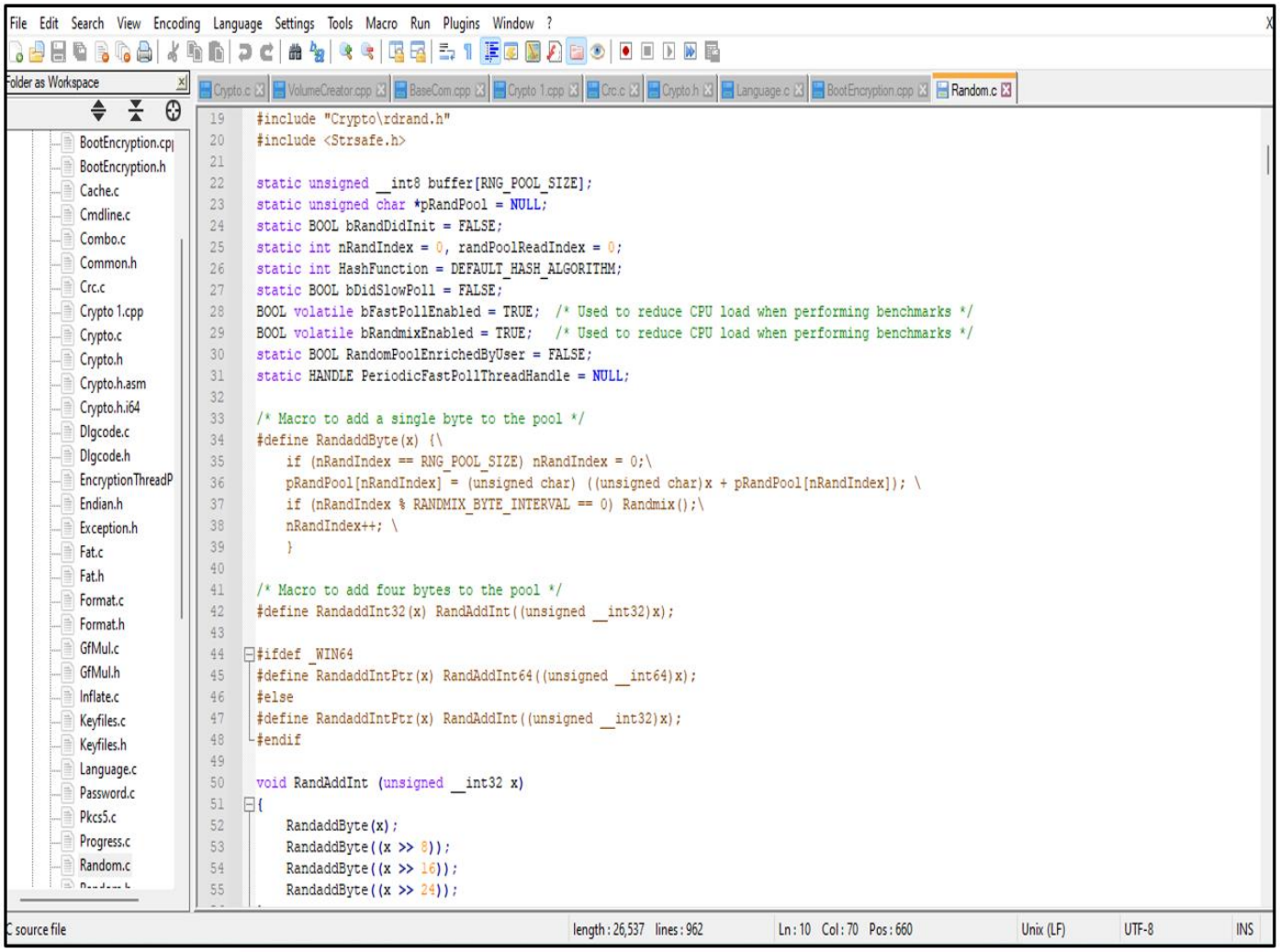


Fig 10 A selection of Random Pool Enriched Extracted Source Codes

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

## VII. CONCLUSION

This paper primarily concentrated on the reverse engineering analysis of VeraCrypt software. The application underwent thorough examination to identify source code within different modules, with its documentation systematically organized into distinct sections. The research presented a detailed methodology for developing a protocol applicable to pre-existing software systems. Through this approach, the programmers' architectural design was observed and documented. By leveraging the reconstructed design framework, which was preserved within a software development environment, it becomes possible to create a new version of the original program. The subsequent sections provide detailed accounts of the interactions and novel findings encountered during the protocol development process. The following statements present a consolidated overview of the identified challenges and issues discovered throughout the investigation. Hybrid cryptographic VeraCrypt models like AES-TWOFISH, AES-BLOWFISH, and SERPENT-TWOFISH-AES combine conventional cyphers to form a strong model that enhances data security. The performance of AES is the best, but its avalanche effect is less than that of other models.

## 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]. H. Majed, H. N. Noura, and A. Chehab, “Overview of Digital Forensics and Anti-Forensics Techniques,” 8th Int. Symp. Digit. Forensics Secur. ISDFS 2020, no. June, 2020, doi: 10.1109/ISDFS49300.2020.9116399.

[3]. M. Gül and E. Kugu, “A Survey On Anti-Forensics Techniques,” Int. Artif. Intell. Data Process. Symp., 2017.

[4]. 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.

[5]. E. Pimenidis, “Computer Anti-forensics Methods and Their Impact on Computer Forensic Investigation Computer Anti-forensics Methods and Their Impact on,” no. August 2009, pp. 145–155, 2016, doi: 10.1007/978-3-642-04062-7.

[6]. G. C. Kessler, “Anti-forensics and the digital investigator,” Proc. 5th Aust. Digit. Forensics Conf., pp. 1–7, 2007.

[7]. A. R. Mothukur, A. Balla, D. H. Taylor, S. T. Sirimalla, and K. Elleithy, “Investigation of Countermeasures to Anti-Forensic Methods,” 2019 IEEE Long Isl. Syst. Appl. Technol. Conf., pp. 1–6, 2019.

[8]. M. S. Bari and A. T. Siddique, “Study on different Cryptography Algorithm a Critical Review,” Int. J. Adv. Res. Comput. Eng. Technol. Vol. 6, Issue 2, Febr. 2017, ISSN 2278 – 1323 Study, vol. 6, no. 2, pp. 177–182, 2017.

[9]. F. Hou, N. Xiao, F. Liu, and H. He, “Secure disk with authenticated encryption and IV verification,” 5th Int. Conf. Inf. Assur. Secur. IAS 2009, vol. 2, pp. 41–44, 2009, doi: 10.1109/IAS.2009.48.

[10]. 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.

[11]. H. Gonzalez, A. A. Kadir, N. Stakhanova, N. Alzahrani, and A. A. Ghorbani, “Exploring reverse engineering symptoms in android apps,” Proc. 8th Eur. Work. Syst. Secur. EuroSec 2015, 2015, doi: 10.1145/2751323.2751330.

[12]. K. Lim, Y. Jeong, S. J. Cho, M. Park, and S. Han, “An android application protection scheme against dynamic reverse engineering attacks,” J. Wirel. Mob. Networks, Ubiquitous Comput. Dependable Appl., vol. 7, no. 3, pp. 40–52, 2016.

[13]. E. J. Schwartz and T. Avgerinos, “All you ever wanted to know about dynamic taint analysis forward symbolic execution (but might have been afraid to ask),” pp. 1–5, 2010.

[14]. J. Narayan, S. K. Shukla, and T. C. Clancy, “A survey of automatic protocol reverse engineering tools,” ACM Comput. Surv., vol. 48, no. 3, 2015, doi: 10.1145/2840724.

[15]. 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.

[16]. D. Cordes and M. Brown, “The Literate-Programming Paradigm,” Computer (Long. Beach. Calif)., vol. 24, no. 6, pp. 52–61, 1991, doi: 10.1109/2.86838.

[17]. T. Diamantopoulos, K. Thomopoulos, and A. Symeonidis, “Reusability-aware Recommendations of Source Code Components,” pp. 488–491, 2016, doi: 10.1145/2901739.2903492.

[18]. E. (2011). R. secrets of reverse engineering (1st ed. ). J. W. & S. Eilam, Reversing: Secrets of Reverse Engineering. Wilely Publishing, 2011.

[19]. B. David, E. Filiol, and K. Gallienne, “Structural analysis of binary executable headers for malware detection optimization,” J. Comput. Virol. Hacking Tech., vol. 13, no. 2, pp. 87–93, 2017, doi: 10.1007/s11416-016-0274-2.

[20]. S. She, R. Lotufo, T. Berger, A. Wąsowski, and K. Czarnecki, “Reverse engineering feature models,” Proc. - Int. Conf. Softw. Eng., pp. 461–470, 2011, doi: 10.1145/1985793.1985856.

[21]. N. Bibi, T. Rana, A. Maqbool, F. Afzal, A. Akgül, and M. De la Sen, “An Intelligent Platform for Software Component Mining and Retrieval,” Sensors (Basel)., vol. 23, no. 1, pp. 1–24, 2023, doi: 10.3390/s23010525.

[22]. R. Hoffmann, J. Fogarty, and D. S. Weld, “Assieme: Finding and leveraging implicit references in a web search interface for programmers,” UIST Proc. Annu. ACM Symp. User Interface Softaware Technol., pp.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25aug411

13–22, 2007, doi: 10.1145/1294211.1294216.
[23]. E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi, "Mining Internet-scale software repositories," Adv. Neural Inf. Process. Syst. 20 - Proc. 2007 Conf., no. January, 2008.
[24]. R. P. and S. C. S. Neelamadhab Padhy, "(PDF) Identifying the Reusable Components from Component-Based System_ Proposed Metrics and Model _ Rasmita Panigrahi - Academia," Adv. Intell. Syst. Comput., 2019.
[25]. Christnatalis, A. M. Husein, M. Harahap, A. Dharma, and A. M. Simarmata, "Hybrid-AES-Blowfish algorithm: Key exchange using neural network," 2019 Int. Conf. Comput. Sci. Inf. Technol. ICoSNIKOM 2019, pp. 4–7, 2019, doi: 10.1109/ICoSNIKOM48755.2019.9111500.
[26]. K. I. Santoso, M. A. Muin, and M. A. Mahmudi, "Implementation of AES cryptography and twofish hybrid algorithms for cloud," J. Phys. Conf. Ser., vol. 1517, no. 1, 2020, doi: 10.1088/1742-6596/1517/1/012099.
[27]. J. H. Chikofsky, E., & Cross, I., "Reverse Engineering and Recovery A Taxonomy," IEEE Softw., p. 7(1), 13-17, 1990.
[28]. D. Mutz, C. Kruegel, W. Robertson, G. Vigna, and R. a Kemmerer, "Reverse Engineering of Network Signatures," Proc. Auscert Asia Pacific Inf. Technol. Secur. Conf. Gold, no. i, pp. 1–86499, 2005.
[29]. E. Summary and I. D. a Pro, "Executive Summary : IDA Pro – at the cornerstone of IT security What is IDA Pro ? How is IDA Pro useful ? Who are IDA Pro users ?," PC Mag., 2009.
[30]. R. Sihwail, K. Omar, and K. A. Z. Ariffin, "A Survey on Malware Analysis Techniques : Static , Dynamic , Hybrid and Memory Analysis," no. September, 2018, doi: 10.18517/ijaseit.8.4-2.6827.
[31]. M. Sikorski and A. Honig, Practical malware analysis: the hands-on guide to dissecting malicious software. no starch press. 2012.
[32]. O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, "Dynamic malware analysis in the modern era—A state of the art survey," ACM Comput. Surv., vol. 52, no. 5, 2019, doi: 10.1145/3329786.